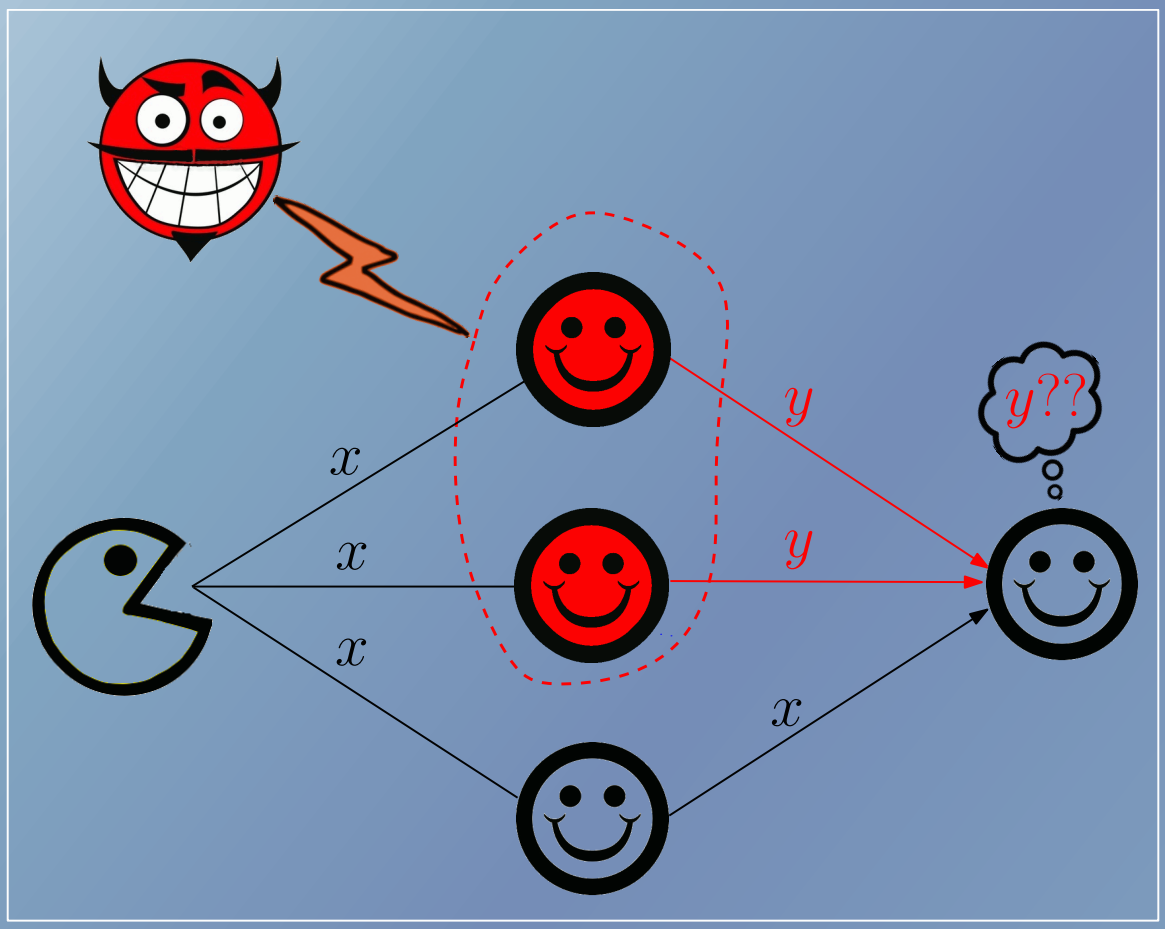
ΑΛΓΟΡΙΘΜΙΚΗ ΘΕΩΡΙΑ ΚΑΤΑΝΕΜΗΜΕΝΩΝ ΥΠΟΛΟΓΙΣΜΩΝ



Ευριπίδης Μάρκου
Ευάγγελος Κρανάκης
Άρης Παγουρτζής
Ντάννυ Κριζάνκ

Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα
www.kallipos.gr
HEALLINK
Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών
Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σχέδιο εξωφύλλου: Δημήτρης Σακαβάλας

ΕΥΡΙΠΙΔΗΣ ΜΑΡΚΟΥ
Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική
Πανεπιστήμιο Θεσσαλίας, Λαμία, Ελλάδα

ΕΥΑΓΓΕΛΟΣ ΚΡΑΝΑΚΗΣ
School of Computer Science
Carleton University, Ottawa, Ontario, Canada

ΑΡΗΣ ΠΑΓΟΥΡΤΖΗΣ
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα, Ελλάδα

ΝΤΑΝΝΥ ΚΡΙΖΑΝΚ
Department of Mathematics and Computer Science
Wesleyan University, Middletown, Connecticut, USA

Αλγοριθμική Θεωρία Κατανεμημένων Υπολογισμών

Αλγοριθμική Θεωρία Κατανεμημένων Υπολογισμών

Συγγραφή

Ευριπίδης Μάρκου (Κύριος Συγγραφέας)

Ευάγγελος Κρανάκης

Άρης Παγουρτζής

Ντάννυ Κριζάνκ

Κριτικός αναγνώστης

Σταύρος Νικολόπουλος

Συντελεστές έκδοσης

Γλωσσική Επιμέλεια: Ευριπίδης Μάρκου

Γραφιστική Επιμέλεια: Ευριπίδης Μάρκου

Τεχνική Επεξεργασία: Άρης Παγουρτζής

ISBN: 978-960-603-504-3

ΣΥΝΔΕΣΜΟΣ ΕΛΛΗΝΙΚΩΝ ΑΚΑΔΗΜΑΪΚΩΝ ΒΙΒΛΙΟΘΗΚΩΝ

Εθνικό Μετσόβιο Πολυτεχνείο

Ηρώων Πολυτεχνείου 9, 15780 Ζωγράφου

www.kallipos.gr

... στην Έλσα, στην Eda, στη Σωτηρία και στη Maura

Πρόλογος

Το βιβλίο αυτό απευθύνεται σε προπτυχιακούς και μεταπτυχιακούς φοιτητές τμημάτων Πληροφορικής και Μαθηματικών που διδάσκονται θέματα σχετικά με τη σχεδίαση και ανάλυση αλγορίθμων. Μπορεί να χρησιμοποιηθεί τόσο από μη εξοικειωμένους με την περιοχή, όσο και από προχωρημένους επιστήμονες στους τομείς της Θεωρητικής Πληροφορικής και ειδικότερα στους τομείς της Σχεδίασης Αλγορίθμων, της Θεωρίας Υπολογισμού αλλά και της Τεχνητής Νοημοσύνης και της Ρομποτικής. Οι αναγνώστες μπορούν να βρουν στο βιβλίο αυτό το απαραίτητο υλικό για την κατανόηση των εννοιών που σχετίζονται με:

- Κατανεμημένους αλγόριθμους
- Κινητούς πράκτορες
- Πολυπλοκότητα αγαθών στους κατανεμημένους υπολογισμούς
- Μοντέλα χρονισμού των κατανεμημένων μοντέλων
- Βασικά προβλήματα και κατανεμημένους αλγόριθμους
- Αναγωγές προβλημάτων
- Σχεδίαση αλγορίθμων με αποδείξεις ορθότητας και ανάλυση πολυπλοκότητας
- Προσεγγισιμότητα με εγγύηση ποιότητας της λύσης για δύσκολα (NP-hard) προβλήματα
- Ζητήματα ασφάλειας δικτύων και πρακτόρων

Δίνεται ιδιαίτερη έμφαση στην παρουσίαση και κατανόηση των θεωρητικών μοντέλων με βάση τα οποία όχι μόνο μπορούν να σχεδιαστούν βέλτιστοι κατανεμημένοι αλγόριθμοι που επιλύουν ρεαλιστικά προβλήματα, αλλά και να αναδειχθούν μέσα από αυτήν την αλγοριθμική θεωρία τα πλεονεκτήματα των κατανεμημένων μεθόδων σε σύγκριση με τους παράλληλους και τους σειριακούς αλγόριθμους.

Το μεγαλύτερο μέρος του βιβλίου εστιάζει στους κατανεμημένους αλγόριθμους που χρησιμοποιούνται από *κινητούς πράκτορες* οι οποίοι μοντελοποιούν λογισμικό που έχει την ικανότητα να αντιγράφει τον εαυτό του και να εκτελείται σε κόμβους του δικτύου. Αυτή είναι μια σύγχρονη προσέγγιση η οποία, χωρίς να είναι περιοριστική σε σχέση με την περισσότερο συμβατική προσέγγιση των κόμβων που ανταλλάσσουν μηνύματα, μελετάται ιδιαίτερα τα τελευταία 15 χρόνια στη διεθνή βιβλιογραφία, καθώς μέσα από αυτή μπορούν να αναλυθούν αποτελεσματικά, παραδοσιακά προβλήματα στα δίκτυα αλλά και νέα προβλήματα που παρουσιάζονται σε εφαρμογές όπως η συντήρηση ενός δικτύου, το ηλεκτρονικό εμπόριο, η ανακάλυψη πληροφορίας αλλά και η εξερεύνηση ενός άγνωστου περιβάλλοντος με τη χρήση ρομπότ. Η χρήση των κινητών πρακτόρων παρουσιάζει πλεονεκτήματα σε σχέση με την αποδοτικότητα, την ανοχή σε σφάλματα, την προσαρμοστικότητα και την απλότητα των μεθόδων.

Στο βιβλίο παρουσιάζονται και αναλύονται θεμελιώδη προβλήματα των κατανεμημένων υπολογισμών, ενώ γίνεται εκτενής αναφορά στις εφαρμογές τους. Οι αλγόριθμοι παρουσιάζονται σε ψευδογλώσσα ενώ οι αποδείξεις ορθότητας και πολυπλοκότητας των αλγορίθμων καθώς και οι αποδείξεις αρνητικών αποτελεσμάτων δίνονται με τυπικό τρόπο και έτσι ώστε να μπορούν να τις παρακολουθήσουν άνετα προπτυχιακοί φοιτητές. Στο τέλος των κεφαλαίων υπάρχουν σχόλια και βιβλιογραφικές αναφορές καθώς και ασκήσεις που βοηθούν στην καλύτερη κατανόηση των μεθόδων σχεδίασης κατανεμημένων αλγορίθμων και στην ανάλυσή τους.

Θα θέλαμε να εκφράσουμε τις ευχαριστίες μας σε πολλούς φίλους, συναδέλφους και συνεργάτες που συνέβαλαν με ευχάριστες συζητήσεις στα θέματα που παρουσιάζονται σε αυτό το βιβλίο. Σε αυτούς περιλαμβάνονται οι Jeremie Chalopin, Jurek Czyzowicz, Shantanu Das, Krzysztof Diks, Stefan Dobrev, Paola Flocchini, Pierre Fraigniaud, Leszek Gasieniec, Nicola Hanusse, Δημήτρης Καββαδίας, Dariusz Kowalski, Λευτέρης Κυρούσης, Ralf Klasing, Arnaud Labourel, Flaminia Luccio, Patrick Morin, Γιώργος Παναγιωτάκος, Andrzej Pelc, Paolo Penna, Giuseppe Prencipe, Sergio Rajsbaum, Tomasz Ratzik, Δημήτρης Σακαβάλας, Nicola Santoro, Fabiano Sarracco, Cindy Sawchuk, Γιάννης Σταματίου, Jorge Urrutia, και Peter Widmayer.

Ευχαριστούμε ιδιαίτερα τον Κριτικό Αναγνώστη Σταύρο Νικολόπουλο για τις εύστοχες παρατηρήσεις, τα σχόλια αλλά και τη συνεργασία κατά τη διάρκεια της συγγραφής του βιβλίου.

Μάρτιος, 2016

Ευριπίδης Μάρκου
Λαμία, Ελλάδα

Ευάγγελος Κρανάκης
Ottawa, ON, Canada

Άρης Παγουρτζής
Αθήνα, Ελλάδα

Ντάννυ Κριζάνκ
Middletown, CT, USA

Περιεχόμενα

Πίνακας συντομεύσεων - ακρωνυμίων

BFS	Αναζήτηση Πρώτα κατά Πλάτος (Breadth First Search), 3
DFS	Αναζήτηση Πρώτα κατά Βάθος (Depth First Search), 3
DM	μείνε ακίνητος (Do not Move), 37
IC($count$)	αύξησε κατά ένα τη μεταβλητή $count$ (Increment variable $count$), 37
MCC	μετακινήσου στο γειτονικό κόμβο αριστερόστροφα (Move one node CounterClockwise), 37
MCL	μετακινήσου στο γειτονικό κόμβο δεξιόστροφα (Move one node CLockwise), 37
PDA	Personal Digital Assistant, 5
PRAM	Παράλληλη Μηχανή Τυχαίας Προσπέλασης (Parallel Random Access Memory), 24
QMA	ανίχνευσε τον κόμβο για την ύπαρξη άλλου πράκτορα (Query host for another Mobile Agent), 37
QT	ανίχνευσε τον κόμβο για την ύπαρξη σημαδιού (Query host for Token), 37
RAM	Μηχανή Τυχαίας Προσπέλασης (Random Access Memory), 24
RD	Συνάντηση με ανίχνευση (Rendezvous with Detection), 43

RP	Συνάντηση χωρίς ανίχνευση (Rendezvous Problem), 43
RT	άφησε στον κόμβο το σημάδι (Release Token), 37

x Πίνακας συντομεύσεων - ακρωνυμίων

ΚΕΦΑΛΑΙΟ 1

Παράλληλοι και Κατανεμημένοι Υπολογισμοί

Σε αυτό το κεφάλαιο γίνεται μια ιστορική αναδρομή στα μοντέλα των παράλληλων υπολογισμών και μια εισαγωγή στους κατανεμημένους υπολογισμούς. Αναφέρονται διάφορα βασικά προβλήματα όπως ο πολλαπλασιασμός πινάκων και το πρόβλημα 'Graph Reachability' και γίνεται παρουσίαση και σύγκριση σειριακών και παράλληλων αλγόριθμων. Στη συνέχεια παρουσιάζονται μοντέλα κατανεμημένων υπολογισμών και γίνεται σύγκριση με τα μοντέλα των παράλληλων υπολογισμών. Αναφέρονται οι λόγοι της εισαγωγής και μελέτης κατανεμημένων αλγόριθμων και γίνεται αναφορά σε βασικά προβλήματα κατανεμημένων υπολογισμών όπως το πρόβλημα 'Broadcast', το πρόβλημα εκλογής αρχηγού, το πρόβλημα της συνάντησης πρακτόρων, το πρόβλημα της εξερεύνησης δικτύων και το πρόβλημα της ανακάλυψης εχθρικών κόμβων σε δίκτυο.

1.1 Εισαγωγή

Από το 1970 άρχισε να αναπτύσσεται η περιοχή της κατασκευής παράλληλων αλγόριθμων με στόχο κυρίως την πιο γρήγορη λύση ενός προβλήματος. Το πιο συνηθισμένο υπολογιστικό μοντέλο είναι ένα δίκτυο ανεξάρτητων επεξεργαστών όπου κάθε επεξεργαστής εκτελεί το δικό του πρόγραμμα. Οι επεξεργαστές μπορούν να επικοινωνούν μεταξύ τους στιγμιαία και συγχρονισμένα χρησιμοποιώντας ένα μεγάλο χώρο κοινής μνήμης. Πιο τυπικά οι επεξεργαστές:

- εκτελούν την πρώτη τους εντολή,
- μετά επικοινωνούν μεταξύ τους και ανταλλάσουν πληροφορίες,
- εκτελούν τη δεύτερή τους εντολή,
- ανταλλάσουν πληροφορίες, κοκ.

Το παραπάνω απλό μοντέλο για τον σχεδιασμό αλγόριθμων είναι όμως δύσκολο να επιτευχθεί τεχνικά. Επιπλέον, η κατασκευή αλγόριθμων για διαφορετικά προβλήματα συχνά απαιτούσε πολύ διαφορετικές τοπολογίες διασύνδεσης καθώς και διαφορετικό πλήθος επεξεργαστών.

1.2 Παράλληλοι Αλγόριθμοι

Αναφέρουμε κάποια προβλήματα παρουσιάζοντας παράλληλους αλγόριθμους για τη λύση τους σε αντιδιαστολή με σειριακούς αλγόριθμους.

1.2.1 Το πρόβλημα του Πολλαπλασιασμού Πινάκων

Το πρόβλημα του πολλαπλασιασμού δύο τετραγωνικών πινάκων μεγέθους n είναι ένα διάσημο πρόβλημα που έχει μελετηθεί εκτενώς στη βιβλιογραφία. Μπορεί να λυθεί με τον απλό βασικό σειριακό αλγόριθμο ο οποίος εκτελεί $O(n^3)$ πολλαπλασιασμούς ενώ το 1969 ο *Strassen* παρουσίασε έναν σειριακό αλγόριθμο που λύνει το πρόβλημα εκτελώντας $O(n^{\log_2 7}) \simeq O(n^{2.807})$ πολλαπλασιασμούς [Strassen, 1969]. Ο αλγόριθμος αυτός βασίστηκε στην ιδέα της εύρεσης του γινομένου μεταξύ δύο 2×2 πινάκων κάνοντας 7 αντί για 8 πολλαπλασιασμούς που απαιτεί ο βασικός αλγόριθμος. Το 1990 παρουσιάστηκε από τους *Coppersmith & Winograd* ένας σειριακός αλγόριθμος που εκτελεί $O(n^{2.376})$ πολλαπλασιασμούς και παρέμεινε για πάνω από 20 χρόνια ο καλύτερος αλγόριθμος (ως προς το πλήθος των πολλαπλασιασμών) [Coppersmith and Winograd, 1990]. Πρόσφατα παρουσιάστηκε ένας νέος αλγόριθμος [Williams, 2012] από την *Williams* που εκτελεί $O(n^{2.3727})$ πολλαπλασιασμούς. Είναι προφανές ότι η πολυπλοκότητα του προβλήματος είναι $\Omega(n^2)$ αφού χρειάζεται να προσπελαστούν $2n^2$ στοιχεία για την παραγωγή του τελικού αποτελέσματος.

Ένας παράλληλος αλγόριθμος που χρησιμοποιεί n^3 επεξεργαστές είναι ο εξής:

Αλγόριθμος 1 `Matrix-Multiplication1`

1: κάθε ένας από τους n^3 επεξεργαστές κάνει έναν πολλαπλασιασμό μεταξύ των στοιχείων a_{ij} και b_{jk}, όπου $1 \leq i, j, k \leq n$
2: κάθε ένας από τους n^2 επεξεργαστές κάνει σειριακά $n-1$ προσθέσεις των όρων $a_{i1}b_{1k}, a_{i2}b_{2k}, \ldots, a_{in}b_{nk}$

Οι πολλαπλασιασμοί στον Αλγόριθμο 1 μπορούν να εκτελεστούν ταυτόχρονα από τους n^3 επεξεργαστές και συνεπώς απαιτείται ένα βήμα για όλους τους πολλαπλασιασμούς. Το κάθε άθροισμα απαιτεί $n-1$ προσθέσεις, ενώ όλα τα n^2 αθροίσματα μπορούν να γίνουν παράλληλα από ισάριθμους επεξεργαστές. Συνεπώς η χρονική πολυπλοκότητα του Αλγόριθμου 1 είναι $\Theta(n)$.

Ένας καλύτερος παράλληλος αλγόριθμος που χρησιμοποιεί επίσης n^3 επεξεργαστές είναι ο Αλγόριθμος 2.

Ο Αλγόριθμος 2 διαφέρει από τον προηγούμενο στον τρόπο που υπολογίζει το άθροισμα των όρων $a_{i1}b_{1k}, a_{i2}b_{2k}, \ldots, a_{in}b_{nk}$. Κάθε ένα από αυτά τα αθροίσματα υπολογίζεται με τη βοήθεια $n/2$ επεξεργαστών ως εξής: Στο πρώτο βήμα ο l επεξεργαστής, όπου $1 \leq l \leq n/2$, υπολογίζει το άθροισμα των όρων $a_{i(2l-1)}b_{(2l-1)k}, a_{i(2l)}b_{(2l)k}$. Μετά το πρώτο βήμα έχουμε $n/2$ νέους όρους. Στο δεύτερο βήμα κάθε ένας από $n/4$ επεξεργαστές υπολογίζει με τον ίδιο τρόπο όπως και προηγου-

Αλγόριθμος 2 `Matrix-Multiplication2`

1: κάθε ένας από τους n^3 επεξεργαστές κάνει έναν πολλαπλασιασμό μεταξύ των στοιχείων a_{ij} και b_{jk}, όπου $1 \leq i, j, k \leq n$
2: $n/2$ επεξεργαστές κάνουν παράλληλα τις προσθέσεις που απαιτούνται για τον υπολογισμό καθενός από τα n^2 στοιχεία του τελικού πίνακα.

μένως το άθροισμα δύο όρων. Τα βήματα συνεχίζονται με τον ίδιο τρόπο, έως ότου να προκύψει το τελικό άθροισμα. Καθώς οι νέοι όροι μετά από κάθε βήμα υποδιπλασιάζονται, ο αριθμός των βημάτων x μέχρι τον υπολογισμό του τελικού αθροίσματος ισούται με το πόσες φορές πρέπει να διαιρέσει κάποιος το n για να πάρει έναν αριθμό που είναι μικρότερος ή ίσος της μονάδας. Δηλαδή το x είναι ο μικρότερος αριθμός για τον οποίον ισχύει $\frac{n}{2^x} \leq 1$. Συνεπώς $x = \Theta(\log n)$. Ο υπολογισμός των n^2 αυτών αθροισμάτων γίνεται παράλληλα ενώ συνολικά απαιτούνται $n^3/2$ επεξεργαστές. Η πολυπλοκότητα του Αλγόριθμου 2 είναι $\Theta(\log n)$.

1.2.2 Το πρόβλημα Graph Reachability

Το πρόβλημα *Graph Reachability* είναι το εξής: Δεδομένου ενός (κατευθυνόμενου) γραφήματος n κόμβων και δύο κόμβων u, v υπάρχει μονοπάτι που να συνδέει τους κόμβους u, v; Το πρόβλημα μπορεί να λυθεί με ένα σειριακό αλγόριθμο που κάνει επίσκεψη των κόμβων του γραφήματος (*DFS, BFS, ...*). Η πολυπλοκότητα του προβλήματος είναι $\Theta(n)$.

Ένας παράλληλος αλγόριθμος για το πρόβλημα είναι ο εξής:

Αλγόριθμος 3 `Graph-Reachability`

1: Έστω A ο πίνακας γειτνίασης (*adjacency matrix*) μεγέθους $n \times n$ του γραφήματος με n κόμβους στον οποίο τα στοιχεία της διαγωνίου $a_{ii} = 1$.
2: Υπολόγισε τους πίνακες $A^2, A^4, ..., A^n$.
3: **if** $a^n_{uv} = 1$ **then**
4: υπάρχει μονοπάτι από το u στο v
5: **else**
6: δεν υπάρχει μονοπάτι από το u στο v
7: **end if**

Η ορθότητα του Αλγόριθμου 3 προκύπτει από την εξής ιδιότητα: Ισχύει $a^k_{ij} = 1$ αν και μόνο αν υπάρχει μονοπάτι μήκους το πολύ k ακμών από τον κόμβο i στον j.

Για την υλοποίηση του Αλγόριθμου 3 χρειάζονται n^3 επεξεργαστές οι οποίοι υπολογίζουν τα γινόμενα των πινάκων εκτελώντας τον Αλγόριθμο 2 που περιγράφηκε στην προηγούμενη ενότητα. Όπως έχει αναφερθεί, ο υπολογισμός κάθε γινομένου χρησιμοποιώντας τον Αλγόριθμο 2, απαιτεί $\Theta(\log n)$ πράξεις, ενώ για τον υπολογισμό του πίνακα A^n χρειάζεται ο υπολογισμός $\log n$ γινομέ-

νων. Συνεπώς η χρονική πολυπλοκότητα του Αλγόριθμου 3 είναι $\Theta(\log^2 n)$. Μάλιστα με την ίδια πολυπλοκότητα μπορούμε να απαντήσουμε στις εξής ερωτήσεις (βλ. Άσκηση 1):

- υπάρχουν μονοπάτια από τον κόμβο u σε όλους τους κόμβους;
- υπάρχουν μονοπάτια από όλους τους κόμβους στον κόμβο v;

1.3 Κατανεμημένοι Υπολογισμοί

Ένα πιο πρόσφατο μοντέλο επίλυσης προβλημάτων είναι εκείνο των κατανεμημένων υπολογισμών το οποίο συχνά οδηγεί στη σχεδίαση αποδοτικότερων αλγόριθμων για μια σειρά προβλημάτων. Δίκτυα επικοινωνίας ή κατανεμημένες βάσεις δεδομένων αποτελούν συχνά το κατανεμημένο περιβάλλον στο οποίο θέλουμε να λύσουμε κάποιο πρόβλημα. Τα δεδομένα του προβλήματος και οι επεξεργαστές είναι συνήθως κατανεμημένοι σε ένα δίκτυο επικοινωνίας. Οι θέσεις τους αλλά και η τοπολογία του δικτύου μπορεί να αλλάζει δυναμικά. Δεν υπάρχει κοινή μνήμη την οποία μπορούν να διαβάζουν ή να γράφουν όλοι οι επεξεργαστές. Σε ένα τέτοιο μοντέλο συνήθως θεωρούμε έναν πεπερασμένο αριθμό από *πράκτορες* που μπορούν να κάνουν υπολογισμούς και να επικοινωνούν μεταξύ τους (ανταλλάσοντας μηνύματα, γράφοντας πληροφορίες ή αφήνοντας σημάδια σε κόμβους του δικτύου, κλπ.) με σκοπό την επίτευξη ενός κοινού στόχου όπως για παράδειγμα:

- να εκτελέσουν κάποια εργασία συντήρησης στο δίκτυο,
- να υπολογίσουν τη λύση σε ένα πρόβλημα,
- να ικανοποιήσουν το αίτημα είτε κάποιου χρήστη εκτός του περιβάλλοντος είτε κάποιου άλλου πράκτορα.

Κάθε πράκτορας έχει τη δυνατότητα να εκτελεί υπολογισμούς και η συνδυασμένη προσπάθεια όλων λύνει το πρόβλημα ή εκτελεί την εργασία συντήρησης στο δίκτυο. Για τη λύση του προβλήματος είναι αναγκαία η σχεδίαση ενός *κατανεμημένου αλγόριθμου* για τους πράκτορες, δηλαδή ενός συνόλου κανόνων που καθορίζουν τι πρέπει να κάνει κάθε πράκτορας. Οι πράκτορες εκτελούν αυτόνομα τους κανόνες. Φυσικά ο στόχος είναι η σχεδίαση *αποδοτικών* αλγόριθμων (δηλαδή με αποδεδειγμένα 'μικρό' υπολογιστικό κόστος) οι οποίοι συνοδεύονται από απόδειξη ορθότητας.

Μια πιο ειδική περιοχή έρευνας στους κατανεμημένους υπολογισμούς είναι η ανάπτυξη μοντέλων και η σχεδίαση αλγόριθμων για πράκτορες που έχουν την ικανότητα να μετακινούνται από κόμβο σε κόμβο του δικτύου. Η χρήση των κινητών πρακτόρων προσφέρει διάφορα πλεονεκτήματα έναντι των πρακτόρων που είναι ακίνητοι και μπορούν μόνο να ανταλλάσουν μηνύματα. Παρακάτω αναφέρουμε μερικά από αυτά τα πλεονεκτήματα:

- μετάβαση του πράκτορα σε 'καλύτερη' θέση στο δίκτυο,
- καλύτερη διαχείριση του φόρτου εργασίας στο δίκτυο (*Load Balancing*),
- μικρός αριθμός μηνυμάτων που ανταλλάσονται (*Latency*),

- δεν χρειάζεται προ-εγκατάσταση λογισμικού στους κόμβους του δικτύου,
- καλύπτει νέες ανάγκες των χρηστών,
- καλύτερη διαχείριση στη δυναμική αλλαγή της τοπολογίας του δικτύου,
- καλύτερη διαχείριση στις περιπτώσεις εμφάνισης σφαλμάτων στο δίκτυο,
- μεγαλύτερη αποδοτικότητα των αλγόριθμων,
- νέες εφαρμογές σε *ad-hoc* δίκτυα.

Το κυριότερο μειονέκτημα που παρουσιάζει η χρήση των κινητών πρακτόρων σε σχέση με τους ακίνητους έχει σχέση με την ασφάλεια του δικτύου. Στα μοντέλα με ακίνητους πράκτορες το θέμα της ασφάλειας συχνά επιλύεται πιο εύκολα και έχει μελετηθεί αρκετά. Η μελέτη προβλημάτων με κινητούς πράκτορες είναι πιο πρόσφατη ενώ εκεί υπάρχουν νέες προκλήσεις που αφορούν την ασφάλεια του συστήματος και δεν έχουν μελετηθεί επαρκώς.

Τα μοντέλα κινητών πρακτόρων έχουν πολλές εφαρμογές στο ηλεκτρονικό εμπόριο, τη συντήρηση ενός δικτύου, στα *Personal Digital Assistants (PDAs)*, κλπ.

Τα ζητήματα της ασφάλειας που πρέπει να αντιμετωπιστούν σε μοντέλα κινητών πρακτόρων είναι συνήθως τα εξής:

- επίθεση ενός πράκτορα σε κόμβο του δικτύου,
  - παροχή ψευδών στοιχείων ταυτότητας του πράκτορα (*masquerading agent*)
  - άρνηση παροχής υπηρεσιών από τον πράκτορα (*denial of service*)
  - μη-επιτρεπόμενη πρόσβαση του πράκτορα σε πληροφορίες που βρίσκονται αποθηκευμένες στον κόμβο του δικτύου (*unauthorized access to data*)
  - αλλοίωση των πληροφοριών που βρίσκονται αποθηκευμένες στον κόμβο του δικτύου (*alteration*)
- επίθεση ενός κόμβου του δικτύου σε πράκτορα,
  - παροχή ψευδών στοιχείων ταυτότητας του κόμβου (*masquerade*)
  - άρνηση παροχής υπηρεσιών από τον κόμβο (*denial of service*)
  - μη-επιτρεπόμενη πρόσβαση του κόμβου σε πληροφορίες που βρίσκονται στη μνήμη του πράκτορα (*unauthorized access*)
  - αλλοίωση των πληροφοριών που βρίσκονται αποθηκευμένες στη μνήμη του πράκτορα (*alteration*)
- επίθεση ενός πράκτορα σε άλλον πράκτορα.
  - παροχή ψευδών στοιχείων ταυτότητας του πράκτορα (*masquerading agent*)

- άρνηση παροχής υπηρεσιών από τον πράκτορα (*denial of service*)
- μη-επιτρεπόμενη πρόσβαση του πράκτορα σε πληροφορίες που βρίσκονται στη μνήμη του άλλου πράκτορα (*unauthorized access*)
- αλλοίωση των πληροφοριών που βρίσκονται αποθηκευμένες στη μνήμη του άλλου πράκτορα (*alteration*)

Οι πιο συνηθισμένες απαιτήσεις ασφάλειας είναι οι εξής:

- ιδιωτικότητα των πληροφοριών (*data privacy*),
- ιδιωτικότητα της επικοινωνίας (*communications privacy*),
- ιδιωτικότητα της θέσης (*location privacy*),
- τιμιότητα (*integrity*),
- υπευθυνότητα (*accountability*),
- διαθεσιμότητα (*availability*),
- ιδιωτικότητα των στοιχείων ταυτότητας (*anonymity*).

Μερικές από τις ενέργειες που γίνονται για την εγγύηση της ασφάλειας του συστήματος είναι οι εξής:

- εξασφαλίζεται πως η βάση του πράκτορα είναι ασφαλής,
- αποφυγή προγραμματιστικών λαθών στα πρωτόκολλα,
- προφύλαξη δεδομένων με τεχνικές της κρυπτογραφίας,
- καταγραφή της ιστορίας του πράκτορα,
- ζητείται από τους πράκτορες να ενημερώνουν τη βάση τους,
- παρακολούθηση των πρακτόρων από άλλους πράκτορες.

Έτσι λοιπόν ένα κρίσιμο ερώτημα που προκύπτει είναι αν μπορούμε να λύσουμε τα ζητήματα ασφάλειας που προκύπτουν όταν χρησιμοποιούμε κινητούς πράκτορες και ταυτόχρονα να διατηρήσουμε τα πλεονεκτήματα έναντι της χρήσης ακίνητων πρακτόρων με πιο σημαντικό ίσως το πλεονέκτημα της μεγαλύτερης αποδοτικότητας των αλγόριθμων.

1.3.1 Βασικές έννοιες

Ένας κινητός πράκτορας συνήθως μοντελοποιείται ως ένα ρομπότ με μνήμη το οποίο εκτελεί έναν ντετερμινιστικό αλγόριθμο. Συχνά το ρομπότ μπορεί να έχει κάποιες από τις εξής δυνατότητες:

- να έχει ανιχνευτές με τους οποίους παρατηρεί το περιβάλλον,
- να κινείται στο δίκτυο,
- να ανταλλάσει μηνύματα με άλλα ρομπότ,
- να διαβάζει και να γράφει πληροφορίες στους κόμβους,
- να μεταφέρει σημάδια (*tokens*) που μπορεί να αφήνει στους κόμβους, κλπ.

Τυπικά, ένας πράκτορας μοντελοποιείται σαν ένα πεπερασμένο αυτόματο (*Finite Moore Automaton*) $A = (X, Y, S, \delta, \lambda, S_0)$, όπου:

- $X \subseteq C_A \times C_E$:
  - C_A : ένα πεπερασμένο σύνολο που αποτελείται από όλα τα δυνατά σενάρια στα οποία μπορεί να βρεθεί ο πράκτορας,
  - C_E : ένα πεπερασμένο σύνολο που αποτελείται από όλα τα δυνατά σενάρια στα οποία μπορεί να βρεθεί το περιβάλλον όπως το παρατηρεί ο πράκτορας,
- Y : ένα πεπερασμένο σύνολο που αποτελείται από όλες τις πράξεις που μπορεί να κάνει ο πράκτορας,
- S : ένα πεπερασμένο σύνολο από καταστάσεις, μια από τις οποίες είναι η S_0 που καλείται αρχική κατάσταση,
- $\delta : S \times X \to S$: η συνάρτηση μετάβασης,
- $\lambda : S \to Y$: μια συνάρτηση που καθορίζει τί πράξη θα κάνει ο πράκτορας.

Η συμπεριφορά ενός πράκτορα έχει συνήθως ως εξής:

- Αρχικά ο πράκτορας A βρίσκεται σε κάποιον κόμβο u_0 του δικτύου στην αρχική κατάσταση S_0
- Ο πράκτορας A παρατηρεί το περιβάλλον και (σύμφωνα με τις συναρτήσεις δ, λ) μεταβαίνει σε μια άλλη κατάσταση $\sigma \in S$ και εκτελεί την πράξη $\lambda(S_0)$.
- Όταν ο A εισέρχεται σε έναν κόμβο v, παρατηρεί το περιβάλλον (π.χ., διαβάζει την ετικέτα της εισόδου/ακμής (*port*) μέσω της οποίας έφτασε στον v και την κατάσταση του κόμβου).

- Τα παραπάνω σε συνδυασμό με την τρέχουσα κατάσταση σ του πράκτορα, προκαλούν μια μετάβαση σε μια νέα κατάσταση σ' (σύμφωνα με την δ).
- Η νέα κατάσταση σ' καθορίζει μια πράξη $\lambda(\sigma')$, κοκ.

Συνήθως επιθυμούμε οι πράκτορες να σταματούν την εκτέλεση του αλγόριθμου όταν λύνεται το πρόβλημα. Αυτό δεν είναι πάντα εύκολο να επιτευχθεί και σε τέτοιες περιπτώσεις οι πράκτορες δεν ξέρουν εάν η αποστολή τους έχει τελειώσει. Συχνά επιθυμούμε οι πράκτορες να εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο.

Εάν δεν υπάρχουν σφάλματα στο δίκτυο τότε θεωρούμε ότι οι επικοινωνία μεταξύ δύο γειτονικών κόμβων ή η διάσχιση μιας ακμής από έναν πράκτορα μπορεί να καθυστερούν για πεπερασμένο χρόνο (αλλά συνήθως άγνωστο εκ των προτέρων (ασύγχρονα δίκτυα)).

Στα περισσότερα μοντέλα θεωρείται ότι ο πράκτορας μπορεί να ξεχωρίσει τις ακμές που οδηγούν στους γείτονές του και σε αυτήν την περίπτωση λέμε ότι έχει *τοπικό προσανατολισμό*. Δηλαδή υπάρχει μια τοπικά συνεπής επισήμανση των ακμών (*edge labelling ή port labelling*) που προσπίπτουν σε κάθε κόμβο η οποία όμως δεν είναι κατ' ανάγκη καθολικά συνεπής.

Το κόστος της επικοινωνίας σε ένα δίκτυο με V κόμβους και E ακμές, συνήθως εκτιμάται ποσοτικά ως εξής:

- με τον συνολικό αριθμό μηνυμάτων M που στάλθηκαν,
- με τον λόγο M/V,
- με τον λόγο M/E,
- με τον συνολικό αριθμό των *bits* πληροφορίας που ανταλλάσσεται.

Η χρονική πολυπλοκότητα ενός αλγόριθμου συνήθως ορίζεται ως ο χρόνος που μεσολαβεί από τη στιγμή που ο πρώτος πράκτορας ξεκινά την εκτέλεση του αλγόριθμου μέχρι τη στιγμή που ο τελευταίος πράκτορας τελειώνει.

1.3.2 Χαρακτηριστικά προβλήματα

Μερικά από τα χαρακτηριστικά προβλήματα σε κατανεμημένο περιβάλλον είναι τα εξής:

- Το πρόβλημα *Broadcast*. Στο πρόβλημα αυτό μια ομάδα πρακτόρων είναι τοποθετημένη στους κόμβους ενός δικτύου (ακριβώς ένας πράκτορας σε κάθε κόμβο). Ένας από τους πράκτορες γνωρίζει μια σημαντική πληροφορία την οποία θα ήθελε να μοιραστεί με όλους τους υπόλοιπους πράκτορες. Ο στόχος είναι η σχεδίαση ενός αλγόριθμου ο οποίος αν εκτελεστεί από τους πράκτορες θα έχει σαν αποτέλεσμα μετά από πεπερασμένο χρόνο όλοι οι πράκτορες να γνωρίζουν την πληροφορία.
- Το πρόβλημα της εκλογής αρχηγού (*Leader Election*). Στο πρόβλημα αυτό μια ομάδα πρακτόρων καλείται να εκλέξει έναν αρχηγό. Ο στόχος λοιπόν είναι να σχεδιαστεί ένας αλγόριθμος

που οδηγεί τους πράκτορες μέσα σε πεπερασμένο χρόνο στην εκλογή ενός αρχηγού. Πιο τυπικά, αρχικά όλοι οι πράκτορες βρίσκονται στην ίδια κατάσταση (*available*), διάσπαρτοι σε ένα δίκτυο. Ο στόχος είναι η σχεδίαση ενός αλγόριθμου ο οποίος οδηγεί όλους τους πράκτορες στην ίδια κατάσταση (*follower*) εκτός από έναν που βρίσκεται στην κατάσταση *leader*.

- Το πρόβλημα της συνάντησης (*Rendezvous*). Στο πρόβλημα αυτό ο στόχος είναι η σχεδίαση ενός αλγόριθμου που μετακινεί δύο ή περισσότερους *κινητούς* πράκτορες έτσι ώστε να συναντηθούν μετά από πεπερασμένο χρόνο σε κάποιο κόμβο του δικτύου. Ενδιαφέρουσες ερωτήσεις που μελετώνται σε αυτό το πρόβλημα είναι οι εξής: Ποιο είναι το πιο 'αδύναμο' σενάριο για το οποίο η συνάντηση είναι δυνατή μέσα σε πεπερασμένο χρόνο (ανεξάρτητα από τις αρχικές θέσεις των πρακτόρων ή το μέγεθος του δικτύου);
  - τι ικανότητες πρέπει να έχουν οι πράκτορες (π.χ., να μπορούν να αφήνουν μηνύματα στους κόμβους, να αφήνουν σημάδια (*tokens*) στους κόμβους),
  - πόση μνήμη χρειάζονται (π.χ., για να μετρούν τους κόμβους που επισκέπτονται),
  - ποια είναι η απαραίτητη αρχική γνώση που πρέπει να έχουν (π.χ., τον αριθμό των κόμβων ή την τοπολογία του δικτύου)
- Το πρόβλημα της εξερεύνησης ενός δικτύου. Στο πρόβλημα αυτό ένας οι περισσότεροι κινητοί πράκτορες καλούνται να εξερευνήσουν ένα δίκτυο για το οποίο αρχικά έχουν ελάχιστες πληροφορίες. Δηλαδή ο στόχος είναι η κατασκευή ενός αλγόριθμου για τους πράκτορες που θα οδηγήσει στην επίσκεψη κάθε κόμβου ή κάθε ακμής του δικτύου από τουλάχιστον έναν πράκτορα. Αυτό είναι ένα από τα πιο κλασσικά προβλήματα στην περιοχή των κατανεμημένων υπολογισμών και έχουν μελετηθεί πολλές παραλλαγές του, αναφορικά με τις δυνατότητες που έχουν οι πράκτορες και την αρχική τους γνώση για το δίκτυο.
- Το πρόβλημα της αναζήτησης μιας μαύρης τρύπας (*Black Hole*). Στο μοντέλο αυτό υπάρχει στο δίκτυο ένας εχθρικός κόμβος που καταστρέφει οποιονδήποτε πράκτορα τον επισκέπτεται χωρίς να αφήνει ίχνη. Μια ομάδα πρακτόρων καλείται να εξερευνήσει το δίκτυο με αποστολή την ανακάλυψη και αναφορά του εχθρικού κόμβου. Ενδιαφέρουσες ερωτήσεις εδώ είναι:
  - Πόσοι πράκτορες χρειάζονται για να ανακαλύψουν τη μαύρη τρύπα;
  - Τί γνώση χρειάζεται να έχουν οι πράκτορες;
  - Σε πόσο χρόνο μπορούν να ανακαλύψουν τη μαύρη τρύπα;
- Προβλήματα Ανοχής Σφάλματος (*Fault Tolerance*). Εδώ περιλαμβάνεται μια μεγάλη κατηγορία προβλημάτων σε δίκτυα για τα οποία θέλουμε να σχεδιάσουμε αλγόριθμους οι οποίοι λειτουργούν ακόμη και όταν κάποιες συνδέσεις του δικτύου αποτυγχάνουν. Σε ένα από τα προβλήματα αυτής της κατηγορίας, συνήθως δίνεται ένα γράφημα με ετικέτες (*labels*) στους κόμβους, του οποίου κάποιες ακμές μπορεί να κοπούν. Ένας πράκτορας που βρίσκεται σε

έναν κόμβο v του γραφήματος πρέπει να επισκεφτεί όλους τους κόμβους της συνεκτικής συνιστώσας του γραφήματος που περιλαμβάνει τον κόμβο v. Ο πράκτορας έχει στη διάθεσή του έναν πιστό χάρτη του γραφήματος αλλά δεν ξέρει που βρίσκονται οι κομμένες ακμές. Ο στόχος είναι η σχεδίαση ενός αλγόριθμου που να εγγυάται την επίσκεψη όλων των κόμβων της συνεκτικής συνιστώσας του δικτύου στον ελάχιστο χρόνο.

1.4 Ασκήσεις

1. Τροποποιήστε τον Αλγόριθμο 3 που παρουσιάστηκε στην ενότητα 1.2.2 για τη επίλυση του προβλήματος *Graph Reachability*, έτσι ώστε να δίνει απάντηση με την ίδια χρονική πολυπλοκότητα ($\Theta(\log^2 n)$) στις εξής δύο ερωτήσεις:

   - υπάρχουν μονοπάτια από τον κόμβο u σε όλους τους κόμβους;
   - υπάρχουν μονοπάτια από όλους τους κόμβους στον κόμβο v;

Βιβλιογραφία

[Coppersmith and Winograd, 1990] Coppersmith, D. and Winograd, S. (1990). Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280.

[Strassen, 1969] Strassen, V. (1969). Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356.

[Williams, 2012] Williams, V. V. (2012). Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, New York, NY, USA. ACM.

ΚΕΦΑΛΑΙΟ 2

Θεμελιώδη Προβλήματα και Αλγόριθμοι με Ακίνητους Πράκτορες

Σε αυτό το κεφάλαιο παρουσιάζονται και αναλύονται βασικές έννοιες και μοντέλα των κατανεμημένων υπολογισμών με ακίνητους πράκτορες και ανταλλαγή μηνυμάτων. Το πρόβλημα Broadcast. Παρουσίαση αλγορίθμων και ανάλυση πολυπλοκότητας. Η έννοια της κοινής γνώσης. Το πρόβλημα Wake-Up. Το πρόβλημα της εκλογής αρχηγού.

2.1 Το πρόβλημα Broadcast

Ας θεωρήσουμε ένα δίκτυο σε κάθε κόμβο του οποίου υπάρχει ακριβώς ένας πράκτορας. Ένας από τους πράκτορες έχει μια σημαντική πληροφορία που θα ήθελε να μοιραστεί με κάθε άλλο πράκτορα. Το πρόβλημα αυτό ονομάζεται *Broadcast*.

Το μοντέλο που χρησιμοποιούμε είναι το ακόλουθο:

- Οι πράκτορες που βρίσκονται σε γειτονικούς κόμβους του δικτύου μπορούν να ανταλλάσουν μηνύματα.
- Η αναπαράσταση του δικτύου γίνεται από ένα μη-κατευθυνόμενο, συνδεδεμένο γράφημα.
- Οι πράκτορες δεν έχουν καμμία αρχική γνώση για την τοπολογία του δικτύου ή το πλήθος των κόμβων του.
- Το δίκτυο δεν έχει σφάλματα.

Θέλουμε να σχεδιάσουμε έναν αλγόριθμο ο οποίος θα εκτελείται από κάθε πράκτορα και ανεξάρτητα από την τοπολογία του δικτύου ή το πλήθος των κόμβων του, θα μεταφέρει την πληροφορία που αρχικά γνωρίζει μόνο ένας από τους πράκτορες, σε όλους τους υπόλοιπους πράκτορες του δικτύου. Δηλαδή, ένας αλγόριθμος που επιλύει το πρόβλημα, θα πρέπει να λειτουργεί σε οποιαδήποτε τοπολογία δικτύου.

Έστω το μη-κατευθυνόμενο γράφημα $G(V, E)$ με $|V| = n$ που αναπαριστά το δίκτυο. Ο χρόνος που χρειάζεται για να ενημερωθούν όλοι οι κόμβοι είναι στη χειρότερη περίπτωση τουλάχιστον $\Omega(n)$ αφού η απόσταση του κόμβου που βρίσκεται μακρύτερα από εκείνον τον κόμβο που έχει αρχικά την πληροφορία μπορεί να είναι $n - 1$ ακμές. Αυτό συμβαίνει όταν το γράφημα είναι μια γραμμή. Σε αυτήν την περίπτωση χρειάζεται να σταλούν $n - 1$ μηνύματα: ένα από κάθε κόμβο (εκτός από τον τελευταίο), στον γειτονικό του στην κατεύθυνση προς τον κόμβο που βρίσκεται μακρύτερα από τον κόμβο που γνωρίζει αρχικά την πληροφορία.

Σχετικά με το ελάχιστο πλήθος μηνυμάτων, μπορεί να αποδειχθεί ότι στη χειρότερη περίπτωση χρειάζεται να σταλούν $\Omega(|E|)$ μηνύματα. Για να δούμε γιατί, ας υποθέσουμε ότι υπάρχει ένας αλγόριθμος $\mathcal{A}$ που λύνει το πρόβλημα στο (τυχαίο) γράφημα G, ενώ συνολικά ανταλλάσσονται το πολύ $|E| - 1$ μηνύματα. Αυτό σημαίνει ότι υπάρχει τουλάχιστον μία ακμή (u, v) την οποία δεν διατρέχει κανένα μήνυμα. Υπενθυμίζουμε, ότι ο αλγόριθμος αυτός θα πρέπει να λύνει το πρόβλημα σε οποιοδήποτε γράφημα. Έστω λοιπόν ένα νέο γράφημα $G'(V', E')$ το οποίο προκύπτει από το γράφημα G αν στη θέση της ακμής (u, v) τοποθετήσουμε τις ακμές (u, w) και (w, v), όπου w είναι ένας καινούριος κόμβος. Δηλαδή το γράφημα G' έχει έναν κόμβο και μία ακμή παραπάνω από το γράφημα G. Εφόσον κατά την εκτέλεση του αλγόριθμου $\mathcal{A}$ στο γράφημα G, κανένα μήνυμα δεν διέτρεχε την ακμή (u, v), η εκτέλεση του αλγόριθμου $\mathcal{A}$ στο γράφημα G' θα έχει σαν συνέπεια κανένα μήνυμα να μην διατρέχει τις ακμές (u, w) και (w, v). Συνεπώς ο κόμβος w δεν πρόκειται ποτέ να λάβει την πληροφορία και άρα ο αλγόριθμος $\mathcal{A}$ δεν επιλύει το πρόβλημα στο γράφημα G'.

Έτσι λοιπόν προκύπτουν τα εξής κάτω φράγματα για τη χρονική πολυπλοκότητα και το κόστος επικοινωνίας:

- $\Omega(n)$ χρόνος.
- $\Omega(|E|)$ μηνύματα.

Ένας απλός αλγόριθμος που λύνει το πρόβλημα είναι ο εξής:

Αλγόριθμος 4 `Broadcast1`

```
1: while γνωρίζεις την πληροφορία do
2:     στείλε την στους γείτονές σου
3: end while
```

Ο αλγόριθμος αρχίζει να εκτελείται από κάθε πράκτορα τη στιγμή που εκείνος μαθαίνει την πληροφορία. Δηλαδή κάθε πράκτορας βρίσκεται σε αναμονή μέχρι να λάβει την πληροφορία. Αμέσως μετά 'ξυπνά' και εκτελεί τον αλγόριθμο.

Όπως βλέπουμε ο παραπάνω αλγόριθμος δεν τερματίζει και στέλνει πολλά μηνύματα χωρίς να χρειάζεται. Δεν είναι δύσκολο όμως να δούμε ότι οι πράκτορες δεν χρειάζεται να στείλουν το μήνυμα περισσότερες από μία φορές τροποποιώντας τον αλγόριθμο ως εξής:

Παρατηρήστε ότι κάθε πράκτορας εκτελεί το νέο αλγόριθμο και τερματίζει αλλά κανείς από τους πράκτορες δεν ξέρει πότε έχουν τελειώσει όλοι οι πράκτορες. Επιπλέον, ένας πράκτορας μπορεί να τερματίσει τον αλγόριθμο πολύ πριν από τους άλλους.

Αλγόριθμος 5 `Broadcast2`

```
1: if έχεις την πληροφορία then
2:     στείλε την στους γείτονές σου
3: end if
```

Για τη χρονική πολυπλοκότητα του αλγορίθμου είναι εύκολο να δούμε ότι στη χειρότερη περίπτωση είναι ίση με το μήκος του μακρύτερου συντομότερου μονοπατιού στο γράφημα.

Ας υπολογίσουμε τώρα το πλήθος των μηνυμάτων που ανταλλάσσονται. Παρατηρήστε ότι κάθε πράκτορας u στέλνει ακριβώς μια φορά ένα μήνυμα σε κάθε του γείτονα μέσω όλων των ακμών που προσπίπτουν στον u. Άρα για κάθε ακμή (v_1, v_2) υπάρχει ένα μήνυμα που ταξιδεύει από τον κόμβο v_1 στον κόμβο v_2 και ένα μήνυμα που ταξιδεύει από τον v_2 στον v_1. Συνεπώς ο συνολικός αριθμός των μηνυμάτων που ανταλλάσσονται είναι $M = 2|E|$.

Μπορούμε να πετύχουμε μικρότερη πολυπλοκότητα μηνυμάτων αν κάνουμε την εξής αλλαγή στον Αλγόριθμο 5: Κάθε πράκτορας στέλνει την πληροφορία σε όλους τους γείτονές του εκτός από εκείνον από τον οποίον έμαθε την πληροφορία. Εάν δηλαδή ένας πράκτορας λάβει το μήνυμα μέσω του port k τότε δεν στέλνει μήνυμα μέσω αυτού του port. Έτσι ο συνολικός αριθμός των μηνυμάτων που ανταλλάσσονται τώρα είναι $M \leq 2|E| - n + 1$, καθώς ο κάθε ένας από τους $n - 1$ κόμβους που δεν γνωρίζουν αρχικά την πληροφορία, δεν θα στείλει μήνυμα σε τουλάχιστον έναν από τους γείτονές του (σε εκείνον από τον οποίον έμαθε την πληροφορία). Στη χειρότερη περίπτωση (δηλαδή όταν η τοπολογία του γραφήματος είναι τέτοια ώστε κάθε πράκτορας μαθαίνει την πληροφορία από ακριβώς έναν άλλον πράκτορα) έχουμε $M = 2|E| - n + 1$.

Αλγόριθμος 6 `Broadcast3`

```
1: if έχεις την πληροφορία then
2:     στείλε την στους γείτονές σου εκτός από εκείνον (ή εκείνους) από τους οποίους έμαθες την
       πληροφορία
3: end if
```

2.2 Η έννοια της Κοινής Γνώσης

Στον τελευταίο αλγόριθμο της προηγούμενης ενότητας, όταν όλοι οι πράκτορες έχουν τερματίσει την εκτέλεση, τότε ξέρουν την πληροφορία. Όμως δεν υπάρχει κάποια χρονική στιγμή κατά την οποία οι πράκτορες να γνωρίζουν ότι όλοι ξέρουν την πληροφορία.

Μια πληροφορία ρ είναι κοινή γνώση για μια ομάδα πρακτόρων w κατά τη χρονική στιγμή t αν και μόνο αν:

1. όλοι στην ομάδα w γνωρίζουν την πληροφορία ρ τη χρονική στιγμή t και
2. όλοι στην ομάδα w τη χρονική στιγμή t γνωρίζουν ότι η πρόταση 1 είναι αλήθεια και

3. όλοι στην ομάδα w τη χρονική στιγμή t γνωρίζουν ότι η πρόταση 2 είναι αλήθεια και...

Ένας ισοδύναμος αναδρομικός ορισμός είναι ο εξής: Η πληροφορία ρ είναι κοινή γνώση για την ομάδα w αν και μόνο αν:

1. όλοι στην ομάδα w γνωρίζουν την πληροφορία ρ και
2. όλοι στην ομάδα w γνωρίζουν ότι η ρ είναι κοινή γνώση

Στα περισσότερα προβλήματα κατανεμημένων υπολογισμών, απαιτείται από τους πράκτορες η *κοινή γνώση* κάποιας πληροφορίας. Οι πιο συνηθισμένες κατηγορίες πληροφοριών για τις οποίες συνήθως απαιτείται κοινή γνώση, είναι οι εξής:

- μετρική: π.χ., αριθμός κόμβων του δικτύου,
- τοπολογική: ιδιότητες γραφήματος (π.χ., τοπολογία δακτυλίου),
- τοπικοί ή καθολικοί χάρτες,
- άλλη: π.χ., οι ετικέτες των κόμβων του δικτύου.

Όπως όμως φαίνεται από τους παραπάνω ορισμούς, για να επιτευχθεί κοινή γνώση για κάποια πληροφορία, μπορεί να απαιτείται άπειρος αριθμός βημάτων. Αυτό φαίνεται στο παρακάτω παράδειγμα.

Υποθέστε ότι σε μια πολεμική σύρραξη οι αντίπαλοι στρατοί είναι παραταγμένοι ως εξής: Ο στρατός A βρίσκεται σε μια κοιλάδα η οποία περιβάλλεται από δύο βουνά. Στα δύο αυτά βουνά βρίσκονται οι στρατοί B και C οι οποίοι είναι σύμμαχοι και αγωνίζονται εναντίον του κοινού εχθρού A. Ο μόνος τρόπος επικοινωνίας των στρατών B και C είναι με τη μεταφορά μηνυμάτων (δηλαδή οι στρατοί δεν έχουν οπτική επαφή μεταξύ τους, ούτε μπορούν να επικοινωνήσουν ασύρματα). Επιπλέον, όλα τα μονοπάτια μέσω των οποίων θα μπορούσε να μεταφερθεί ένα μήνυμα από το στρατό B στο στρατό C (και από τον C στο B) περνούν από την κοιλάδα που βρίσκεται ο στρατός A. Ο στρατός A είναι πολύ μεγαλύτερος από τον B ή τον C αλλά μικρότερος από το άθροισμα των B και C. Έτσι λοιπόν οι B και C θα κερδίσουν τη μάχη αν και μόνο αν επιτεθούν ταυτόχρονα στον A. Ο στρατός B αποφασίζει να στείλει έναν αγγελιοφόρο στο στρατό C με το μήνυμα της ακριβούς ώρας επίθεσης. Επειδή όμως ο αγγελιοφόρος πρέπει να διασχίσει την κοιλάδα, είναι δυνατόν να συλληφθεί από το στρατό A και να μην καταφέρει να παραδώσει το μήνυμα στο στρατό C. Συνεπώς σε περίπτωση που ο στρατός C πάρει το μήνυμα θα πρέπει να στείλει ένα άλλο μήνυμα στο στρατό B για να ξέρει ο B ότι το αρχικό του μήνυμα παραδόθηκε. Αν όμως ο B πάρει το μήνυμα με την απάντηση του C θα πρέπει να στείλει ένα ακόμα μήνυμα στο C, ειδάλλως ο C δεν θα ξέρει ότι ο B γνωρίζει ότι το αρχικό μήνυμα παραδόθηκε. Ο C με τη σειρά του, αν λάβει το νέο μήνυμα θα πρέπει να στείλει ένα ακόμη μήνυμα, καθώς σε διαφορετική περίπτωση ο B δεν θα γνωρίζει ότι ο C έμαθε πως ο B είχε πάρει την απάντηση στο αρχικό του μήνυμα. Είναι εύκολο να δει κανείς ότι αυτή η διαδικασία θα πρέπει να συνεχίζεται επ' άπειρον και συνεπώς η πληροφορία της ακριβούς

ώρας επίθεσης δεν μπορεί να γίνει κοινή γνώση μετά από έναν πεπερασμένο αριθμό ανταλλαγής μηνυμάτων σε αυτό το μοντέλο.

Ευτυχώς, μερικές φορές η απόκτηση κοινής γνώσης για κάποια πληροφορία είναι δυνατόν να επιτευχθεί μετά από πεπερασμένο αριθμό βημάτων (βλ. Άσκηση 3).

2.3 Το πρόβλημα Wake-Up

Συχνά σε ένα κατανεμημένο περιβάλλον, για να γίνει μια εργασία απαιτείται η συμμετοχή όλων των πρακτόρων. Μερικές φορές όμως κάποιοι από τους πράκτορες είναι ενεργοί και έτοιμοι να εκτελέσουν έναν αλγόριθμο ενώ άλλοι πράκτορες είναι ανενεργοί και δεν γνωρίζουν ότι πρέπει να εκτελέσουν κάποιους υπολογισμούς. Σε ένα τέτοιο σενάριο, είναι αναγκαίο όλοι οι πράκτορες να είναι ενεργοί πριν αρχίσουν να εκτελούν τον αλγόριθμο. Συνεπώς θα πρέπει να προηγηθεί η εκτέλεση κάποιας διαδικασίας μετά το τέλος της οποίας όλοι οι πράκτορες θα είναι ενεργοί. Προφανώς αυτή τη διαδικασία μπορούν να την ξεκινήσουν μόνο οι πράκτορες που είναι ήδη ενεργοί. Κάθε ένας όμως από αυτούς τους πράκτορες πρέπει να δράσει αυτόνομα, καθώς δεν γνωρίζει ούτε αν υπάρχουν, ούτε ποιοί είναι οι άλλοι πράκτορες που είναι ενεργοί. Το πρόβλημα αυτό ονομάζεται *Wake-Up* και ορίζεται πιο τυπικά ως εξής: Κάποιοι από τους πράκτορες του δικτύου (ένας σε κάθε κόμβο) βρίσκονται στην κατάσταση *awake* και οι υπόλοιποι στην κατάσταση *asleep*. Ο σκοπός είναι η σχεδίαση ενός αλγορίθμου ο οποίος μετά την εκτέλεσή του έχει σαν αποτέλεσμα όλοι οι πράκτορες να βρίσκονται στην κατάσταση *awake*.

Το μοντέλο μας έχει ως εξής:

- Οι πράκτορες που βρίσκονται σε γειτονικούς κόμβους του δικτύου μπορούν να ανταλλάσουν μηνύματα.
- Η αναπαράσταση του δικτύου γίνεται από ένα μη-κατευθυνόμενο, συνδεδεμένο γράφημα.
- Οι πράκτορες δεν έχουν καμμία αρχική γνώση για την τοπολογία του δικτύου ή το πλήθος των κόμβων του.
- Το δίκτυο δεν έχει σφάλματα.

Δεν είναι δύσκολο να παρατηρήσει κανείς τη σχέση του προβλήματος *Wake-Up* με το πρόβλημα *Broadcast*. Το δεύτερο είναι ειδική περίπτωση του πρώτου στην οποία μόνος ένας πράκτορας είναι αρχικά ενεργός. Επιπλέον, μπορούμε να δούμε το πρόβλημα *Wake-Up* σαν να θέλουμε να λύσουμε το πρόβλημα *Broadcast* στην περίπτωση που υπάρχουν πάνω από ένας πράκτορες οι οποίοι γνωρίζουν αρχικά την πληροφορία που θέλουν να μοιραστούν με τους υπόλοιπους. Έτσι λοιπόν ένα λογικό βήμα είναι να δούμε αν μπορούμε να τροποποιήσουμε τους αλγόριθμους που παρουσιάσαμε στην Ενότητα 2.1 έτσι ώστε να οδηγούν στην επίλυση του προβλήματος *Wake-Up*. Πράγματι ο παρακάτω αλγόριθμος (που είναι σχεδόν ίδιος με τον Αλγόριθμο 6) λύνει το πρόβλημα *Wake-Up*.

Για τον συνολικό αριθμό M των μηνυμάτων που ανταλλάσσονται κατά την εκτέλεση του παραπάνω αλγόριθμου, σε ένα γράφημα $G(V, E)$ με $|V| = n$, μπορεί να αποδειχθεί εύκολα (βλ.

Αλγόριθμος 7 `Wake-Up`

```
1: if είσαι active then
2:     στείλε ένα μήνυμα στους γείτονές σου;
3: else
4:     if λάβεις ένα μήνυμα then
5:         γίνε active;
6:         στείλε ένα μήνυμα στους γείτονές σου εκτός από εκείνον (ή εκείνους) από τους οποίους
           έλαβες μήνυμα;
7:     end if
8: end if
```

Άσκηση 4) ότι ισχύει:

$$2|E| \geq M \geq 2|E| - n + 1$$

Όπως φαίνεται εύκολα, ο Αλγόριθμος 7 επιτυγχάνει τη μετάβαση όλων των πρακτόρων στην κατάσταση *active* μέσα σε το πολύ n βήματα. Τέλος τα κάτω φράγματα για την χρονική πολυπλοκότητα και το πλήθος των μηνυμάτων του προβλήματος είναι αντίστοιχα $\Omega(n)$ και $\Omega(|E|)$, αφού το πρόβλημα είναι ειδική περίπτωση του προβλήματος *Broadcast*, και συνεπώς ο Αλγόριθμος 7 είναι βέλτιστος (κατά την τάξη μεγέθους) σε χρόνο και αριθμό μηνυμάτων, ενώ η πολυπλοκότητα χρόνου και μηνυμάτων του προβλήματος είναι αντίστοιχα $\Theta(n)$ και $\Theta(|E|)$.

Αν η τοπολογία του γραφήματος είναι γνωστή τότε έχουμε τα εξής αποτελέσματα (βλ. Άσκηση 5):

- *Δέντρο*: Ο αλγόριθμος που παρουσιάσαμε στέλνει συνολικά $M = n + k_* - 2$ μηνύματα, όπου k_* είναι ο αριθμός των πρακτόρων που αρχικά βρίσκονται στην κατάσταση *awake*.
- *Πλήρες γράφημα*: Το κάτω φράγμα του αριθμού των μηνυμάτων για το πρόβλημα *Wake-Up* παραμένει $\Omega(n^2)$ σε αντίθεση με το πρόβλημα *Broadcast* που σε αυτήν την περίπτωση έχει πολυπλοκότητα $n - 1$.

2.4 Το πρόβλημα της Εκλογής Αρχηγού

Η επίλυση ενός προβλήματος σε ένα κατανεμημένο περιβάλλον, απαιτεί πολλές φορές από έναν πράκτορα να έχει προσωρινά τη θέση του κεντρικού διαχειριστή ή καθολικά υπεύθυνου για την εκτέλεση μιας εργασίας από τους υπόλοιπους πράκτορες. Σε κάποιες περιπτώσεις, η ανάγκη αυτή προκύπτει γιατί κάνει πιο απλή την εύρεση και σχεδίαση ενός αλγόριθμου για ένα πολύπλοκο πρόβλημα. Σε άλλες περιπτώσεις η εμφάνιση ενός και μόνου καθολικά υπεύθυνου μπορεί να υπαγορεύεται από τη φύση του προβλήματος. Το πρόβλημα της επιλογής ενός τέτοιου καθολικά υπεύθυνου ανάμεσα σε μια ομάδα αυτόνομων πρακτόρων καλείται πρόβλημα εκλογής αρχηγού (*Leader Election*) και πιο τυπικά ορίζεται ως εξής: Δίνεται ένα γράφημα, στον κάθε κόμβο του οποίου βρίσκεται ένας πράκτορας. Αρχικά όλοι οι πράκτορες βρίσκονται στην ίδια κατάσταση (*available*).

Σκοπός είναι η σχεδίαση ενός αλγόριθμου ο οποίος μετά την εκτέλεσή του έχει σαν αποτέλεσμα ότι όλοι οι πράκτορες βρίσκονται στην ίδια κατάσταση (*follower*) εκτός από έναν που βρίσκεται στην κατάσταση *leader*.

Σε πολλές περιπτώσεις η επίλυση του προβλήματος είναι αδύνατη. Για παράδειγμα θεωρήστε δύο πανομοιότυπους πράκτορες που βρίσκονται στην ίδια κατάσταση και μπορούν να επικοινωνούν μεταξύ τους. Ας υποθέσουμε ότι υπάρχει ένας αλγόριθμος που οδηγεί τους πράκτορες στην εκλογή ενός αρχηγού. Εφόσον οι πράκτορες αρχικά βρίσκονται στην ίδια κατάσταση και εκτελούν τον ίδιο αλγόριθμο είναι υποχρεωμένοι να κάνουν τις ίδιες ενέργειες. Έτσι λοιπόν αν στείλουν κάποιο μήνυμα ο ένας στον άλλον θα πρέπει να είναι το ίδιο μήνυμα και θα το στείλουν την ίδια στιγμή. Όταν ο ένας από αυτούς λάβει το μήνυμα θα το λάβει και ο άλλος. Θα εκτελούν πάντα ταυτόχρονα τους ίδιους υπολογισμούς και θα βρίσκονται πάντα στην ίδια κατάσταση. Συνεπώς αν ένας από αυτούς γίνει αρχηγός, ταυτόχρονα θα γίνει και ο άλλος. Άρα ο αλγόριθμος δεν μπορεί να επιλύει το πρόβλημα.

Από τα παραπάνω γίνεται φανερό ότι χρειάζεται να προσθέσουμε κάτι στο μοντέλο του κατανεμημένου συστήματος που να μπορεί να βοηθήσει τους πράκτορες να σπάσουν τη συμμετρία και να πάρουν διαφορετικές αποφάσεις προκειμένου να μπορούν να λύσουν το πρόβλημα. Ένα τέτοιο μοντέλο είναι το εξής:

- οι πράκτορες έχουν ταυτότητες (*labels*) ανά δύο διαφορετικές που ανήκουν σε ένα ολικά διατεταγμένο σύνολο (π.χ., φυσικοί αριθμοί),
- οι γειτονικοί πράκτορες μπορούν να ανταλλάσουν μηνύματα

Θεωρήστε τον Αλγόριθμο 8 που ακολουθεί.

Αλγόριθμος 8 `Leader-Election`

```
1: πήγαινε στην κατάσταση leader
2: στείλε σε όλους την ταυτότητά σου (broadcast)
3: if λάβεις κάποια ταυτότητα μικρότερη από τη δική σου then
4:     άλλαξε την κατάστασή σου σε follower
5: end if
```

Αν ο Αλγόριθμος 8 εκτελεστεί από πράκτορες που βρίσκονται σε ένα πλήρες δίκτυο, τότε μετά από έναν πεπερασμένο αριθμό βημάτων το πρόβλημα έχει λυθεί, όλοι οι πράκτορες έχουν τερματίσει τον αλγόριθμο, ενώ με μια μικρή τροποποίηση μπορούν όλοι να ξέρουν την ταυτότητα του αρχηγού (βλ. Άσκηση 7). Σε άλλες τοπολογίες δικτύου όμως αυτό δεν είναι τόσο εύκολο (βλ. Άσκηση 8).

2.5 Σχόλια και βιβλιογραφικές παρατηρήσεις

Ο κύριος στόχος των ερευνητών στην περιοχή της σχεδίασης και ανάλυσης κατανεμημένων αλγόριθμων είναι η αποδοτικότητα. Δηλαδή η σχεδίαση αλγόριθμων που πετυχαίνουν την επίλυση

του προβλήματος γρήγορα και χρησιμοποιούν όσο το δυνατόν λιγότερα αγαθά (όπως πλήθος πρακτόρων, μνήμη και άλλες δυνατότητες των πρακτόρων, αριθμό μηνυμάτων που ανταλλάσσονται, κλπ). Η απόδειξη της ορθότητας ενός αλγόριθμου είναι βέβαια υποχρεωτική και συνήθως επιτυγχάνεται χρησιμοποιώντας μαθηματικές τεχνικές (π.χ., [Lynch and Tuttle, 1987]). Τα μοντέλα των κατανεμημένων συστημάτων και των υπολογισμών που γίνονται σε αυτά, εστιάζουν (και διαφοροποιούνται) κυρίως σε παραμέτρους που σχετίζονται άμεσα ή έμμεσα με την αποδοτικότητα του υπολογισμού και την πολυπλοκότητα του προβλήματος, όπως είναι: η τοπολογία του δικτύου επικοινωνίας, ο τρόπος επικοινωνίας, το μέγεθος και είδος της πληροφορίας που είναι αρχικά διαθέσιμη στους πράκτορες, κλπ.

Η έννοια της *κοινής γνώσης*, είναι χρήσιμη στη σχεδίαση αλγορίθμων και στην απόδειξη ιδιοτήτων τους ([Halpern and Moses, 1990, Rosenschein, 1985]), ιδιαίτερα σε κατανεμημένα περιβάλλοντα στα οποία υπάρχουν σφάλματα.

Το μοντέλο του πράκτορα που συνήθως (αλλά όχι πάντα) χρησιμοποιείται, είναι ο πράκτορας που *αντιδρά* σε ερεθίσματα και γεγονότα (σε αντίθεση με τον πράκτορα που απλά *δρα*). Η θεμελιώδης διαφορά μεταξύ αυτών των δύο μοντέλων μπορεί να γίνει κατανοητή με το παράδειγμα που ακολουθεί. Θεωρήστε ένα μήνυμα που ταξιδεύει στο δίκτυο με προορισμό κάποιο πράκτορα. Στο μοντέλο του πράκτορα που *δρα*, ο πράκτορας που περιμένει το μήνυμα, θα προσπαθεί περιοδικά να το λάβει (ακόμη και όταν το μήνυμα δεν έχει φτάσει). Κάθε προσπάθεια λήψης του μηνύματος είναι ένα γεγονός που προκύπτει από μια πράξη του πράκτορα. Συνεπώς αυτό το απλό σενάριο μπορεί να προκαλέσει έναν απροσδιόριστο αριθμό από γεγονότα. Αντίθετα, στο μοντέλο του πράκτορα που *αντιδρά*, ο πράκτορας δεν εκτελεί καμμία λειτουργία μέχρι τη λήψη του μηνύματος. Σε αυτό το μοντέλο, η άφιξη του μηνύματος 'ξυπνά' τον πράκτορα και προκαλεί την αντίδρασή του. Τα δύο αυτά μοντέλα είναι ισοδύναμα, αναπαριστώντας διαφορετικούς τρόπους με τους οποίους μπορούμε να δούμε και να περιγράψουμε κάποιες διαδικασίες. Σε σχέση όμως με την περιγραφή και την ανάλυση των κατανεμημένων αλγορίθμων, το μοντέλο του πράκτορα που *αντιδρά* είναι απλούστερο στη χρήση.

2.6 Ασκήσεις

1. Να αποδείξετε ότι ο Αλγόριθμος 6 που παρουσιάστηκε στην Ενότητα 2.1 και λύνει το πρόβλημα *Broadcast* σε ένα οποιοδήποτε γράφημα $G(V, E)$ με $|V| = n$, έχει πολυπλοκότητα μηνυμάτων $M \leq 2|E| - n + 1$. Δώστε ένα γράφημα $G(V, E)$ στο οποίο η εκτέλεση του Αλγόριθμου 6 προκαλεί την ανταλλαγή ακριβώς $2|E| - n + 1$ μηνυμάτων.

2. Να αποδείξετε ότι το πρόβλημα *Broadcast* (βλ. ενότητα 2.1) έχει:

   - χρονική πολυπλοκότητα $n - 1$ και πολυπλοκότητα μηνυμάτων $n - 1$, όταν η τοπολογία του δικτύου είναι δέντρο n κόμβων.
   - χρονική πολυπλοκότητα 1 και πολυπλοκότητα μηνυμάτων $n - 1$, όταν η τοπολογία του δικτύου είναι πλήρες γράφημα n κόμβων.

3. Ένας αριθμός από n 'έξυπνους' μαθητές παίζουν στον κήπο του σχολείου και k από αυτούς λερώνονται με λάσπη στο μέτωπό τους χωρίς να το καταλάβουν. Μετά το παιχνίδι στον κήπο, όλοι οι μαθητές μπαίνουν στην τάξη. Όταν ο δάσκαλος μπαίνει στην τάξη τους λέει:

   - 'Τουλάχιστον ένας από εσάς έχει λάσπη στο μέτωπο. Εάν ακριβώς και μόνο οι k μαθητές που έχουν λάσπη καταφέρουν να το καταλάβουν και μου το πουν, τότε δεν θα τιμωρηθεί κανείς. Αλλιώς θα τιμωρηθείτε όλοι. Θα ξαναέρθω στην αίθουσα άλλες το πολύ $n-1$ φορές. Θα πρέπει τώρα ή κάποια από τις φορές που έρχομαι, ακριβώς και μόνο οι k μαθητές που είναι λερωμένοι να μου το πουν ταυτόχρονα όταν είμαι στην αίθουσα. Δεν επιτρέπεται καμμία επικοινωνία μεταξύ σας (π.χ. να μιλήσετε ο ένας στον άλλο, να δείξετε, κλπ). Δεν επιτρέπεται επίσης να κουνηθείτε. Επιτρέπεται μόνο να βλέπετε και να σκέφτεστε.'

   Πώς μπορούν να καταλάβουν (και μάλιστα ταυτόχρονα) οι k μαθητές ότι είναι λερωμένοι; Με άλλα λόγια, πώς μπορεί η τιμή του k να γίνει *κοινή γνώση* (βλ. Ενότητα 2.2); Σκιαγραφήστε τον αλγόριθμο που εκτελούν και εξηγήστε γιατί δουλεύει.

   Υπόδειξη: Αν ήταν μόνο ένας ο μαθητής που είχε λάσπη στο μέτωπο, τότε θα το καταλάβαινε αμέσως. Επαγωγή στην τιμή του k.

4. Να αποδείξετε ότι ο Αλγόριθμος 7 που παρουσιάστηκε στην Ενότητα 2.3 και λύνει το πρόβλημα *Wake-Up* σε ένα οποιοδήποτε γράφημα $G(V,E)$ με $|V|=n$, έχει πολυπλοκότητα μηνυμάτων M για την οποία ισχύει ότι $2|E| \geq M \geq 2|E|-n+1$.

   Δώστε δύο γραφήματα $G(V,E)$ στα οποία η εκτέλεση του Αλγόριθμου 7 προκαλεί την ανταλλαγή ακριβώς $2|E|-n+1$ και $2|E|$ μηνυμάτων αντίστοιχα.

5. Σχετικά με το πρόβλημα *Wake-Up* (βλ. ενότητα 2.3), να αποδείξετε ότι:

   - Ο Αλγόριθμος 7 που παρουσιάστηκε στην ενότητα 2.3, έχει πολυπλοκότητα μηνυμάτων $M = n + k_* - 2$ μηνύματα, όπου k_* είναι ο αριθμός των πρακτόρων που αρχικά βρίσκονται στην κατάσταση *awake*, όταν η τοπολογία του γραφήματος είναι ένα οποιοδήποτε δέντρο n κόμβων.
   - Η πολυπλοκότητα του αριθμού των μηνυμάτων για το πρόβλημα *Wake-Up* είναι $\Theta(n^2)$, όταν η τοπολογία του δικτύου είναι πλήρες γράφημα n κόμβων.

6. Τρεις άνθρωποι έχουν μπει σε σειρά (ο ένας πίσω από τον άλλον) κατά ύψος έτσι ώστε ο ψηλότερος βλέπει τους άλλους δύο, ο μεσαίος βλέπει τον κοντότερο και ο κοντότερος δεν βλέπει κανέναν. Υπάρχουν 3 μαύρα καπέλα και 2 κόκκινα. Κάποιος τέταρτος άνθρωπος φορά ένα καπέλο σε κάθε έναν από τους τρεις ανθρώπους. Ο καθένας από τους ανθρώπους δεν μπορεί να δει τι χρώμα καπέλο φορά ο ίδιος. Ο τέταρτος άνθρωπος θέτει την εξής ερώτηση σε καθέναν από τους τρεις ανθρώπους: 'Τι χρώμα καπέλο φοράς;' ξεκινώντας από τον ψηλότερο προς τον κοντότερο. Οι τρεις άνθρωποι είναι ειλικρινείς και θα απαντήσουν μόνο όταν πραγματικά ξέρουν, αλλιώς απαντούν 'δεν ξέρω'.

Να αποδείξετε ότι εάν όλοι μπορούν να ακούσουν τις απαντήσεις ο κοντότερος πάντα μπορεί να καταλάβει τι χρώμα καπέλο φορά.

7. Θεωρήστε το παρακάτω μοντέλο για το πρόβλημα της εκλογής αρχηγού (βλ. ενότητα 2.4): Υπάρχουν n πράκτορες που καταλαμβάνουν τους n κόμβους ενός δακτυλίου (ένας πράκτορας σε κάθε κόμβο). Οι πράκτορες είναι ακίνητοι, συγχρονισμένοι, μπορούν να ανταλλάσουν μηνύματα και έχουν ταυτότητες ανά δύο διαφορετικές που ανήκουν σε ένα ολικά διατεταγμένο σύνολο. Τα μηνύματα μπορούν να ταξιδεύουν στο δακτύλιο μόνο προς τη μία κατεύθυνση.

   - Να σχεδιάσετε έναν κατανεμημένο ντετερμινιστικό αλγόριθμο για την επίλυση του προβλήματος της εκλογής αρχηγού, έτσι ώστε όλοι οι πράκτορες να τερματίζουν τον αλγόριθμο και να γνωρίζουν την ταυτότητα του αρχηγού.
   - Ποιά είναι η χρονική πολυπλοκότητα του αλγορίθμου σας;
   - Ποιος είναι ο συνολικός αριθμός μηνυμάτων που ανταλλάσσονται κατά την εκτέλεση του αλγόριθμού σας;

8. Θεωρήστε το μοντέλο της προηγούμενης άσκησης για το πρόβλημα της εκλογής αρχηγού (βλ. ενότητα 2.4) με τη διαφορά ότι οι n πράκτορες καταλαμβάνουν τους n κόμβους ενός γενικού μη-κατευθυνόμενου γραφήματος του οποίου δεν γνωρίζουν την τοπολογία.

   - Να σχεδιάσετε έναν κατανεμημένο ντετερμινιστικό αλγόριθμο για την επίλυση του προβλήματος της εκλογής αρχηγού.
   - Ποιά είναι η χρονική πολυπλοκότητα του αλγορίθμου σας;
   - Ποιος είναι ο συνολικός αριθμός μηνυμάτων που ανταλλάσσονται κατά την εκτέλεση του αλγόριθμού σας;
   - Είναι δυνατόν ο αλγόριθμος να σχεδιαστεί έτσι ώστε όλοι οι πράκτορες να τερματίζουν την εκτέλεσή του και να γνωρίζουν την ταυτότητα του αρχηγού;

Βιβλιογραφία

[Halpern and Moses, 1990] Halpern, J. Y. and Moses, Y. (1990). Knowledge and common knowledge in a distributed environment. *Journal of the ACM (JACM)*, 37(3):549–587.

[Lynch and Tuttle, 1987] Lynch, N. A. and Tuttle, M. R. (1987). Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 137–151. ACM.

[Rosenschein, 1985] Rosenschein, S. J. (1985). Formal theories of knowledge in ai and robotics. *New Generation Computing*, 3(4):345–357.

ΚΕΦΑΛΑΙΟ 3

Κατανεμημένοι Υπολογισμοί με Κινητούς Πράκτορες

Σε αυτό το κεφάλαιο παρουσιάζονται και αναλύονται σε μεγαλύτερο βάθος, βασικές έννοιες και μοντέλα των κατανεμημένων υπολογισμών με κινητούς πράκτορες. Τί είναι οι κινητοί πράκτορες (mobile agents); Σε ποιες εφαρμογές μπορούν να χρησιμοποιηθούν και ποια είναι τα πλεονεκτήματά τους σε σχέση με τις περισσότερο συμβατικές προσεγγίσεις των κόμβων που ανταλλάσσουν μηνύματα; Ποια είναι τα μοντέλα επικοινωνίας των κινητών πρακτόρων; Ποια είναι τα μοντέλα χρονισμού στα κατανεμημένα δίκτυα; Πως ορίζεται η πολυπλοκότητα των κατανεμημένων αλγορίθμων και ποια είναι τα αγαθά (π.χ., χρόνος, μνήμη, γνώση των πρακτόρων για το δίκτυο, υπολογιστική δύναμη και ικανότητες των πρακτόρων) που ενδιαφερόμαστε να ελαχιστοποιήσουμε; Συζητάμε και αναλύουμε τις έννοιες της τοπικής γνώσης (local knowledge) και της καθολικής γνώσης (common knowledge) των ιδιοτήτων του δικτύου, καθώς και την ανταλλαγή μηνυμάτων και άλλους μηχανισμούς επικοινωνίας των πρακτόρων. Τέλος, αναφέρουμε μερικά βασικά προβλήματα με πράκτορες.

3.1 Εισαγωγή

Μια απλή και χρήσιμη έννοια που έχει εφαρμογές σε πολλές περιοχές της πληροφορικής είναι η έννοια του *πράκτορα* που εκτελεί μια εργασία για λογαριασμό μιας άλλης οντότητας. Αν προσθέσουμε τη δυνατότητα της *κίνησης* στον πράκτορα, τότε η επίλυση συγκεκριμένων προβλημάτων μπορεί να γίνει πιο απλά και αποδοτικά, όπως για παράδειγμα στις εξής εφαρμογές:

- *Συντήρηση Δικτύου.* Σε ένα ανομοιογενές δίκτυο είναι αναγκαία η συχνή ενημέρωση του λογισμικού των κόμβων του, η ανίχνευση πιθανών προβλημάτων ασφάλειας, κλπ. Μια απλή μέθοδος θα ήταν να χρησιμοποιούμε έναν πράκτορα ή μια ομάδα πρακτόρων που επισκέπτεται συχνά τους κόμβους του δικτύου για να πραγματοποιεί αυτή τη συντήρηση.
- *Ηλεκτρονικό Εμπόριο.* Σε κάποιες περιπτώσεις, η επιτυχία μιας συγκεκριμένης συνδιαλλαγής απαιτεί την σχεδόν ταυτόχρονη επιτυχία πολλαπλών συνδιαλλαγών. Για παράδειγμα, κατά

την προετοιμασία ενός ταξιδιού χρειάζεται η αγορά αεροπορικών εισιτηρίων, η κράτηση του ξενοδοχείου, κλπ. Ένας κινητός πράκτορας μπορεί να κινηθεί μεταξύ των κόμβων του δικτύου που χειρίζονται τις αντίστοιχες εφαρμογές και να εγγυηθεί ότι όλες οι συνδιαλλαγές μπορούν να πραγματοποιηθούν πριν ο χρήστης πραγματοποιήσει κάποια από αυτές.

- *Έξυπνη Αναζήτηση.* Κατά την αναζήτηση πληροφοριών από βάσεις δεδομένων που βρίσκονται διασκορπισμένες στους κόμβους ενός δικτύου, συχνά ο τρόπος που αναζητούμε την πληροφορία (queries) από έναν κόμβο πρέπει να διαμορφωθεί ανάλογα με τις απαντήσεις που λάβαμε από την αναζήτηση σε προηγούμενο κόμβο. Ένας πράκτορας με την ικανότητα να φιλτράρει τις πληροφορίες τοπικά και να διαφοροποιεί τη λειτουργία του ενώ κινείται από κόμβο σε κόμβο, θα μπορούσε να είναι πιο αποτελεσματικός από την πιο συμβατική μέθοδο κατά την οποία οι πληροφορίες επιστρέφουν κάθε φορά στο χρήστη για οδηγίες και φιλτράρισμα.

- *Εξερεύνηση με τη βοήθεια ρομπότ.* Ένα επικίνδυνο περιβάλλον είναι λογικό να εξερευνείται πρώτα από ρομπότ των οποίων τη ζημιά ή την ολική καταστροφή μπορούμε να ανεχτούμε. Μια απλή και ίσως πιο φθηνή λύση από ένα ακριβό ρομπότ που ελέγχεται από έναν άνθρωπο, είναι να έχουμε μια ομάδα από ρομπότ (πράκτορες) με αρκετά περιορισμένες δυνατότητες, που μπορούν να επικοινωνούν μεταξύ τους τα οποία συνεργάζονται για να εξερευνήσουν το περιβάλλον.

Σε αυτό και στα επόμενα κεφάλαια, εστιάζουμε στα μοντέλα πρακτόρων όπως έχουν εμφανιστεί στα κατανεμημένα συστήματα (βλ. [Milojičić et al., 1998]) αλλά τα περισσότερα από αυτά μπορούν να βρουν εφαρμογές και σε άλλες περιοχές όπως η τεχνητή νοημοσύνη (π.χ., έξυπνα συστήματα πολλών πρακτόρων [Wooldridge, 1999]), η ρομποτική (π.χ., αυτόνομοι κινητοί πράκτορες [Roy and Dudek, 2001]), υπολογιστικά οικονομικά (π.χ., οικονομικά μοντέλα βασισμένα σε πράκτορες [Tesfatsion, 2002]) και δίκτυα (π.χ., ενεργά δίκτυα [Tennenhouse et al., 1997]). Στη συνέχεια εξηγούμε την έννοια του κινητού πράκτορα και συζητούμε τα πλεονεκτήματα και μειονεκτήματα αυτής της προσέγγισης.

3.1.1 Η έννοια του κινητού πράκτορα

Ο *κινητός πράκτορας* είναι ένα αυτόνομο λογισμικό (προορισμένο να εκτελεί μια εργασία), το οποίο μπορεί να μεταγράφεται από ένα υπολογιστικό περιβάλλον σε άλλο παραμένοντας ενεργό καθώς μεταφέρεται. Αποτελεί ένα ισχυρό εργαλείο για τη σχεδίαση, ανάλυση και υλοποίηση κατανεμημένων αλγόριθμων σε δίκτυα υπολογιστών. Ακριβέστερα, θεωρούμε έναν κινητό πράκτορα σαν ένα λογισμικό με τις εξής ιδιότητες:

- *Αυτονομία.* Ο πράκτορας έχει αρκετή αυτονομία για να μπορεί να λειτουργεί και να παίρνει αρκετές αποφάσεις χωρίς να χρειάζεται να παίρνει συχνά οδηγίες από μια κεντρική αρχή (ή το χρήστη).

- *Ικανότητα κίνησης.* Ο πράκτορας μπορεί να κινείται από κόμβο σε κόμβο ενός δικτύου. Κατά τη διάρκεια αυτής της κίνησης θεωρούμε ότι η κατάσταση (μνήμη) του πράκτορα μεταφέρεται.

Εκτός από τις παραπάνω ιδιότητες, συχνά ένας πράκτορας έχει επίσης τα εξής χαρακτηριστικά:

- *Αλληλεπίδραση.* Ένας πράκτορας μπορεί να αλληλεπιδρά με το περιβάλλον του, ζητώντας πληροφορίες από τον κόμβο που επισκέπτεται, ενημερώνοντας τον κόμβο με πληροφορίες που μεταφέρει, κλπ. Σε πολλές εφαρμογές θεωρούμε ότι περισσότεροι από ένας πράκτορες βρίσκονται στο δίκτυο και αυτοί μπορούν να αλληλεπιδρούν μεταξύ τους ανταλλάσοντας πληροφορίες συνήθως όταν συναντιούνται σε έναν κόμβο. Στις περισσότερες εφαρμογές οι πράκτορες συνεργάζονται για να λύσουν κάποιο πρόβλημα ή να εκτελέσουν κάποιες εργασίες, αλλά είναι επίσης πιθανό να ανταγωνίζονται μεταξύ τους. Ο τρόπος αλληλεπίδρασης μεταξύ των πρακτόρων εξαρτάται από το μοντέλο αλλά συνήθως η επικοινωνία τους γίνεται είτε μέσω μηνυμάτων, είτε μέσω της πρόσβασης σε κάποιο χώρο που υπάρχει στους κόμβους του δικτύου.

- *Προσαρμοστικότητα.* Η χρησιμότητα ενός πράκτορα αυξάνει αρκετά, ανάλογα με την ικανότητά του να προσαρμόζεται σε νέες συνθήκες, να μαθαίνει από την εμπειρία του και να μοντελοποιεί σωστά τις απαιτήσεις του χρήστη και των πρακτόρων που συναντά.

Φυσικά για τη μελέτη και ανάλυση των παραπάνω χαρακτηριστικών, είναι αναγκαία η ανάπτυξη μιας κατάλληλης θεωρίας.

3.1.2 Τί προσφέρει η χρήση των κινητών πρακτόρων;

Τα προβλήματα που μπορούν να επιλυθούν με τη χρήση κινητών πρακτόρων, μπορούν επίσης να επιλυθούν και με πιο συμβατικές μεθόδους κατανεμημένων υπολογισμών (όπως για παράδειγμα η ανταλλαγή μηνυμάτων μεταξύ των κόμβων ενός δικτύου). Τί παραπάνω λοιπόν προσφέρει η χρήση των κινητών πρακτόρων; Μερικά από τα πλεονεκτήματα της χρήσης κινητών πρακτόρων αναφέρονται παρακάτω.

- *Αποδοτικότητα.* Υποθέτοντας ότι οι πράκτορες αναπαριστούν μικρά λογισμικά μπορούν να κάνουν πιο αποδοτική χρήση του δικτύου, χωρίς να το 'πλημμυρίζουν' με μηνύματα. Στην περίπτωση της επίσκεψης σειριακά n κόμβων του δικτύου, όπου οι πληροφορίες που συγκεντρώνονται από έναν κόμβο αποτελούν μέρος της εισόδου του λογισμικού που εκτελείται σε έναν άλλο κόμβο, ένας κινητός πράκτορας μπορεί να εκτελέσει αυτές τις εργασίες κινούμενος σε έναν κύκλο n ακμών και έτσι έχουμε ένα κόστος από n επικοινωνίες, ενώ η επικοινωνία ενός ακίνητου πράκτορα θα έδινε κόστος $2n$ για την ανταλλαγή μηνυμάτων από και προς κάθε κόμβο. Στην περίπτωση που είναι δυνατή η παράλληλη πρόσβαση σε κόμβους, μια ομάδα από κινητούς πράκτορες θα μπορούσε να επισκεφτεί τους κόμβους πιο γρήγορα από ότι ένας ακίνητος πράκτορας.

- *Ανεκτικότητα σε σφάλματα.* Σε περιπτώσεις όπου ο χρήστης έχει περιορισμένη πρόσβαση, ή μη-συνεχή πρόσβαση στο δίκτυο, ένας κινητός πράκτορας μπορεί να λειτουργεί για λογαριασμό του χρήστη ακόμη και όταν εκείνος δεν έχει πρόσβαση στο δίκτυο και να ενημερώνει τον χρήστη όταν η πρόσβαση αποκαθίσταται. Σε περιπτώσεις όπου οι κόμβοι βρίσκονται συχνά εκτός λειτουργίας χωρίς ενημέρωση για τις διακοπές, ένας κινητός πράκτορας μπορεί να μετακινείται σε κόμβους που λειτουργούν και να συνεχίζει τις εργασίες του.

- *Προσαρμοστικότητα.* Είναι εύκολο να προσθέσει κανείς δυνατότητες στους πράκτορες έτσι ώστε να μπορούν να προσαρμοστούν σε νέες συνθήκες. Πιο πολύπλοκοι πράκτορες μπορούν να είναι σχεδιασμένοι έτσι ώστε να μαθαίνουν από την προηγούμενή τους εμπειρία.

- *Ευκολία χρήσης.* Σε πολλές περιπτώσεις η ανάλυση ενός μοντέλου που χρησιμοποιεί αυτόνομους κινητούς πράκτορες, είναι πιο εύκολη και από τη σκοπιά του χρήστη και από τη σκοπιά του προγραμματιστή. Σε μερικές περιπτώσεις η εμπειρία του χρήστη από την εφαρμογή βελτιώνεται, ενώ η σχεδίαση και υλοποίηση του συστήματος μπορεί να γίνει πιο εύκολα.

Για να αξιολογηθούν (τουλάχιστον θεωρητικά) οι παραπάνω ισχυρισμοί, και ιδιαίτερα αυτοί της αποδοτικότητας και της ανεκτικότητας σε σφάλματα, είναι αναγκαία η κατασκευή ενός μοντέλου για την ανάλυση αλγορίθμων για κινητούς πράκτορες.

3.2 Αλγοριθμικά μοντέλα για προβλήματα με κινητούς πράκτορες

Για την σχεδίαση και ανάλυση αλγορίθμων με κινητούς πράκτορες, χρειαζόμαστε ένα υπολογιστικό μοντέλο. Για παράδειγμα ένα διάσημο μοντέλο στο οποίο μπορούν να σχεδιαστούν και να αναλυθούν σειριακοί αλγόριθμοι είναι η Μηχανή Τυχαίας Προσπέλασης (Random Access Memory Machine, RAM). Για την ανάλυση παράλληλων αλγόριθμων έχουν εμφανιστεί πολλά μοντέλα, όπως η Παράλληλη Μηχανή Τυχαίας Προσπέλασης (Parallel Random Access Memory, PRAM). Γενικά, σε ένα μοντέλο προσπαθούμε να αναπαραστήσουμε τις πιο σημαντικές λεπτομέρειες των υπολογιστικών διαδικασιών. Αποτελείται από την περιγραφή των επιτρεπόμενων διεργασιών ή μεταβάσεων που μπορούν να εκτελεστούν από μια υπολογιστική διαδικασία. Για παράδειγμα το μοντέλο RAM επιτρέπει διαδικασίες διαβάσματος ή γραψίματος, αριθμητικές πράξεις, κλπ. Μετά τον ορισμό ενός κατάλληλου μοντέλου ακολουθεί ο ορισμός των αγαθών που μας ενδιαφέρουν και μπορούμε να μετρήσουμε, όπως ο χρόνος (δηλαδή το πλήθος των βασικών πράξεων), ο χώρος (δηλαδή το πλήθος των καταχωρητών ή των δυνατών καταστάσεων ενός αυτομάτου), κλπ. Μετά από αυτούς τους ορισμούς μπορούμε να σχεδιάσουμε και να αναλύσουμε αλγόριθμους για συγκεκριμένα προβλήματα. Θεωρώντας ότι το μοντέλο περιγράφει τους δυνατούς υπολογισμούς με αρκετή ακρίβεια, μπορεί να χρησιμοποιηθεί για:

- την ανάλυση της πολυπλοκότητας (το ποσό ενός αγαθού που χρησιμοποιείται) διαφορετικών αλγόριθμων για ένα πρόβλημα με σκοπό να βρούμε τον πιο αποδοτικό αλγόριθμο, και

- την απόδειξη κάτω φραγμάτων στην πολυπλοκότητα ενός αλγόριθμου για κάποιο πρόβλημα όπως και στη σύγκριση της πολυπλοκότητας διαφορετικών προβλημάτων.

Για την περιγραφή αλγορίθμων με κινητούς πράκτορες είναι ανάγκη να μοντελοποιήσουμε και τους κινητούς πράκτορες αλλά και το δίκτυο στο οποίο λειτουργούν. Στη συνέχεια αυτού το κεφαλαίου, αλλά και στα επόμενα κεφάλαια παρουσιάζουμε μια κλάση από μοντέλα για το δίκτυο και τους πράκτορες. Η επιλογή ενός συγκεκριμένου μοντέλου, εξαρτάται φυσικά και από το πρόβλημα που θέλουμε να μελετήσουμε.

3.2.1 Μοντέλα κατανεμημένων δικτύων

Το μοντέλο του κατανεμημένου δικτύου είναι συνήθως ένα γράφημα του οποίου οι κόμβοι αναπαριστούν τους κόμβους του δικτύου και οι ακμές τις μεταξύ τους συνδέσεις (δες για παράδειγμα [Lynch, 1996] ή [Santoro, 2006]).

Οι κόμβοι μπορεί να έχουν ή να μην έχουν διακρίσιμες ταυτότητες. Αν δεν έχουν, τότε το δίκτυο καλείται ανώνυμο. Αυτό σημαίνει ότι ένας πράκτορας δεν μπορεί να ξεχωρίσει δύο διαφορετικούς κόμβους, εκτός αν διαφέρουν οι βαθμοί τους. Οι ακμές που προσπίπτουν σε έναν κόμβο συνήθως θεωρούνται διακρίσιμες, αλλά ένα σημαντικό θέμα είναι αν η ανάθεση ετικεττών στις ακμές έχει γίνει με ένα καθολικά συνεπή τρόπο ή όχι. Για παράδειγμα, στην περίπτωση ενός δακτυλίου, οι ακμές μπορεί να έχουν ετικέττες τέτοιες ώστε να υποδηλώνεται η (δεξιόστροφη ή αριστερόστροφη) κατεύθυνση σε κάθε κόμβο του δακτυλίου με καθολικά συνεπή τρόπο. Μπορεί όμως οι ετικέττες (π.χ., 1 και 2) να έχουν αποδοθεί έτσι ώστε ένας πράκτορας που ακολουθεί κάθε φορά την ετικέττα 1 για να φύγει από οποιονδήποτε κόμβο, δεν καταφέρνει να εκτελέσει μια πλήρη περιστροφή στον δακτύλιο. Αν η ανάθεση των ετικεττών στις ακμές του δικτύου ικανοποιεί κάποιες συγκεκριμένες ιδιότητες κωδικοποίησης, καλείται *sense of direction* [Flocchini et al., 1998], και σε αυτήν την περίπτωση ο πράκτορας μπορεί να χρησιμοποιήσει τις ετικέττες σαν πυξίδα κατά την πλοήγησή του στο δίκτυο. Η ιδιότητα αυτή του δικτύου μπορεί να είναι κρίσιμη για την επιλυσιμότητα ενός προβλήματος και την αποδοτικότητα των αλγορίθμων.

Άλλη σημαντική παράμετρος ενός δικτύου είναι ο χρονισμός του. Σε ένα συγχρονισμένο δίκτυο, υπάρχει ένα καθολικό ρολόι διαθέσιμο σε όλους τους κόμβους. Αυτό το ρολόι μπορεί να χρησιμοποιηθεί από τους πράκτορες για να συγχρονιστούν. Σε ένα τέτοιο δίκτυο, συνήθως θεωρούμε ότι σε ένα βήμα ένας πράκτορας μπορεί να διασχίσει μια ακμή φτάνοντας σε κάποιο κόμβο, να εκτελέσει κάποιους υπολογισμούς στον κόμβο και να αποχωρήσει, ενώ όλοι οι πράκτορες εκτελούν αυτές τις λειτουργίες συγχρονισμένα. Σε ένα ασύγχρονο δίκτυο δεν υπάρχει καθολικό ρολόι και ο χρόνος που χρειάζεται ένας πράκτορας για να κινηθεί από έναν κόμβο σε έναν άλλον κόμβο ή για να κάνει υπολογισμούς, είναι μεν πεπερασμένος αλλά δεν είναι γνωστός από την αρχή. Μάλιστα, ακόμη και αν ένας πράκτορας εκτελέσει την ίδια εργασία περισσότερες από μία φορές (π.χ., διάσχιση της ίδιας ακμής), η διάρκειά της μπορεί να είναι διαφορετική.

Άλλη παράμετρος είναι οι υπηρεσίες που παρέχονται από τους κόμβους στους πράκτορες. Όλοι οι κόμβοι παρέχουν αρκετό χώρο για την προσωρινή αποθήκευση κάποιου πράκτορα και αρ-

κετή υπολογιστική ισχύ για να εκτελέσει ο πράκτορας τους υπολογισμούς του. Σε κάποιες περιπτώσεις μπορεί να υπάρχουν εχθρικοί κόμβοι που αρνούνται την επίσκεψη των πρακτόρων ή, ακόμη χειρότερα, τους καταστρέφουν. Οι κόμβοι είναι επίσης υπεύθυνοι για τη μεταφορά των πρακτόρων σε γειτονικούς κόμβους όταν αυτό ζητηθεί. Εκτός από αυτές τις υπηρεσίες, ένας κόμβος μπορεί να παρέχει μόνιμο αποθηκευτικό χώρο στον οποίο ένας πράκτορας μπορεί να αποθηκεύσει πληροφορίες που παραμένουν εκεί ακόμα και μετά την αναχώρηση του πράκτορα από τον κόμβο. Έτσι αυτές οι πληροφορίες μπορούν να διαβαστούν από άλλους πράκτορες που επισκέπτονται τον κόμβο. Αυτός ο τύπος χώρου σε ένα κόμβο αναφέρεται συνήθως σαν πίνακας στον οποίο ένας πράκτορας μπορεί να αφήνει μηνύματα για τον εαυτό του και τους άλλους πράκτορες. Μια άλλη υπηρεσία που μπορεί να παρέχει ο κόμβος είναι ένας μηχανισμός ανταλλαγής μηνυμάτων μεταξύ δύο πρακτόρων που επισκέπτονται ταυτόχρονα τον κόμβο.

Τέλος, έχει φυσικά σημασία αν κάποιο στοιχείο του δικτύου μπορεί να αποτύχει, και ποιο μπορεί να είναι το είδος της αποτυχίας. Αν για παράδειγμα μπορεί να συμβεί αποτυχία κόμβων ή συνδέσεων, ή ακόμη και σφάλματα που επηρεάζουν μόνο τη λειτουργία των πρακτόρων και όχι τη συνδεσιμότητα του δικτύου, (π.χ., η απώλεια πληροφοριών στους κόμβους).

3.2.2 Μοντέλα κινητών πρακτόρων

Θέλουμε να μοντελοποιήσουμε ένα σύνολο από οντότητες λογισμικού οι οποίες δρουν αρκετά αυτόνομα και έχουν τη δυνατότητα να κινούνται από κόμβο σε κόμβο ενός δικτύου διατηρώντας την κατάστασή τους. Οι κόμβοι του δικτύου παρέχουν κάποιο χώρο για αποθήκευση (μόνιμη ή προσωρινή) πληροφοριών και την υπολογιστική δύναμη για την εκτέλεση του λογισμικού. Ο κινητός πράκτορας μοντελοποιείται συνήθως σαν ένα αυτόματο που αποτελείται από ένα σύνολο καταστάσεων (μνήμη του αυτομάτου) και μια συνάρτηση μετάβασης. Η συνάρτηση μετάβασης δέχεται σαν είσοδο την τρέχουσα κατάσταση του πράκτορα καθώς και πιθανόν την τρέχουσα κατάσταση του κόμβου στον οποίο βρίσκεται ο πράκτορας και επιστρέφει μια νέα κατάσταση για τον πράκτορα, μεταβάλλει την κατάσταση του κόμβου και πιθανόν κινεί τον πράκτορα σε έναν γειτονικό κόμβο.

Πιο τυπικά, ένας πράκτορας $\mathcal{A}$ συνήθως αναπαρίσταται ως ένα αυτόματο[1] $\mathcal{A} = (X, Y, \mathcal{S}, \delta, \lambda, S_0)$, όπου $X \subseteq \mathcal{D}_v \times \mathcal{C}_v$, $Y \subseteq \mathcal{D}_v \times \{\texttt{actions}\}$, $\mathcal{S}$ είναι ένα σύνολο από καταστάσεις μεταξύ των οποίων υπάρχει μια ιδιαίτερη κατάσταση S_0 που καλείται *αρχική* κατάσταση, $\delta : \mathcal{S} \times X \to \mathcal{S}$, και $\lambda : \mathcal{S} \to Y$. $\mathcal{D}_v$ είναι το σύνολο των ετικεττών των ακμών (port labels) που συνδέουν τον κόμβο v με όλους τους γειτονικούς του κόμβους στο δίκτυο. $\mathcal{C}_v$ είναι το σύνολο των πιθανών καταστάσεων στις οποίες μπορεί να βρίσκεται ο κόμβος v (π.χ., κωδικοποιεί την ύπαρξη άλλων πρακτόρων ή μηνυμάτων εκεί).

Αρχικά ο πράκτορας βρίσκεται σε κάποιον κόμβο u_0 στην αρχική κατάσταση $S_0 \in \mathcal{S}$. Η κατάσταση S_0 καθορίζει μια ενέργεια (π.χ., αλλαγή της κατάστασης του κόμβου) και μια κατεύθυνση

[1] Ο πρώτος γνωστός αλγόριθμος που σχεδιάστηκε για την εξερεύνηση ενός γραφήματος χρησιμοποιώντας ένα αυτόματο, παρουσιάστηκε από τον Shannon ([Shannon, 1951]) το 1951.

προς την οποία θα κινηθεί ο πράκτορας αφήνοντας τον κόμβο u_0, $\lambda(S_0) \in Y$. Όταν ο πράκτορας φτάνει σε έναν κόμβο v, συμπεριφέρεται ως εξής: διαβάζει την ετικέττα i (port label) της ακμής μέσω της οποίας έφτασε στον κόμβο v, και την κατάσταση $c_v \in \mathcal{C}_v$ του κόμβου v (δηλαδή, αν υπάρχουν άλλοι πράκτορες ή μηνύματα στον v). Το ζεύγος $(i, c_v) \in X$ είναι η είσοδος που προκαλεί τη μετάβαση του πράκτορα από την κατάσταση S στην κατάσταση $S' = \delta(S, (i, c_v))$. Η κατάσταση S' καθορίζει μια ενέργεια (π.χ., αλλαγή της κατάστασης του κόμβου) και μια κατεύθυνση $\lambda(S')$, προς την οποία θα κινηθεί ο πράκτορας αφήνοντας τον κόμβο v. Ο πράκτορας συνεχίζει να κινείται με αυτόν τον τρόπο, πιθανόν επ' άπειρον.

Σε μερικές περιπτώσεις θεωρούμε πιθανοτικά αυτόματα που έχουν τη δυνατότητα να χρησιμοποιούν κάποιες τυχαίες τιμές σαν μέρος της εισόδου τους. Τέτοιοι πράκτορες καλούνται συνήθως πιθανοτικοί.

Μια σημαντική ιδιότητα που μπορεί να έχουν οι πράκτορες είναι να είναι διακρίσιμοι, δηλαδή να έχουν διακρίσιμες ταυτότητες ή ετικέττες. Οι πράκτορες χωρίς διακρίσιμες ετικέττες καλούνται συνήθως ανώνυμοι πράκτορες. Οι ανώνυμοι πράκτορες είναι αναγκασμένοι να εκτελούν ακριβώς τον ίδιο αλγόριθμο, δηλαδή είναι πανομοιότυπα αυτόματα. Εφόσον η ταυτότητα ενός πράκτορα μπορεί να είναι μέρος της εισόδου της συνάρτησης μετάβασής του, πράκτορες με διαφορετικές ταυτότητες έχουν τη δυνατότητα να εκτελούν διαφορετικούς αλγόριθμους.

Η γνώση που έχει ένας πράκτορας σχετικά με το δίκτυο στο οποίο βρίσκεται και σχετικά με τους άλλους πράκτορες μπορεί να είναι κρίσιμη για την επιλυσιμότητα ενός προβλήματος και την αποδοτικότητα ενός αλγόριθμου. Για παράδειγμα, η γνώση του μεγέθους ή της τοπολογίας του δικτύου, του πλήθους ή/και των ταυτοτήτων των άλλων πρακτόρων μπορεί να χρησιμοποιηθεί στον αλγόριθμο που εκτελεί ο πράκτορας. Αν αυτές οι πληροφορίες είναι διαθέσιμες στους πράκτορες, θεωρούμε ότι αποτελούν μέρος της αρχικής τους κατάστασης. Εναλλακτικά μπορούμε να θεωρούμε ότι αυτές οι πληροφορίες μεταφέρονται από τους κόμβους στους πράκτορες και δεν είναι κατ' ανάγκη κωδικοποιημένες στους πράκτορες.

Μια σημαντική παράμετρος για την περίπτωση πολλών πρακτόρων, είναι ο τρόπος που επικοινωνούν. Για παράδειγμα, κατά πόσο είναι δυνατόν ένας πράκτορας που βρίσκεται σε κάποιον κόμβο να ανιχνεύσει αν βρίσκονται και άλλοι πράκτορες εκεί; Η μέθοδος με την οποία επικοινωνούν είναι ένα σημαντικό χαρακτηριστικό του μοντέλου. Για παράδειγμα, ένας πράκτορας μπορεί να έχει τη δυνατότητα να διαβάσει τις καταστάσεις στις οποίες βρίσκονται οι άλλοι πράκτορες που επισκέπτονται τον ίδιο κόμβο. Επίσης μπορεί να επιτρέπεται η επικοινωνία με ανταλλαγή μηνυμάτων ή μέσω ενός κοινού χώρου που βρίσκεται σε κάθε κόμβο του δικτύου στον οποίο χώρο οι πράκτορες μπορούν να διαβάζουν και να γράφουν πληροφορίες. Άλλα χαρακτηριστικά είναι η δυνατότητα των πρακτόρων να δημιουργούν αντίγραφα του εαυτού τους και η δυνατότητα να ενώνονται όταν συναντιούνται και να λειτουργούν σαν ένας πράκτορας για το υπόλοιπο του αλγόριθμου.

3.2.3 Ποσοτική εκτίμηση αγαθών

Για ένα δεδομένο μοντέλο πρακτόρων και δικτύου, υπάρχουν διάφορα αγαθά των οποίων ενδιαφερόμαστε να υπολογίσουμε την πολυπλοκότητα. Σε συγχρονισμένα μοντέλα, η μέτρηση του χρόνου γίνεται χρησιμοποιώντας το καθολικό ρολόι. Σε ασύγχρονα μοντέλα η αξιολόγηση του χρόνου γίνεται τις περισσότερες φορές μετρώντας τον αριθμό των βημάτων που χρειάζονται στην χειρότερη περίπτωση. Το μέγεθος της μνήμης ενός πράκτορα εξαρτάται από το πλήθος των bits που απαιτούνται για την κωδικοποίηση των καταστάσεών του, δηλαδή είναι ανάλογο με το λογάριθμο (με βάση το δύο) του πλήθους των διαφορετικών καταστάσεων. Αν ο πράκτορας μπορεί να στέλνει μηνύματα, τότε το πλήθος και το μέγεθος αυτών των μηνυμάτων είναι επίσης σημαντικό. Άλλες ποσότητες που ενδιαφέρουν είναι το μέγεθος της κοινής μνήμης που παρέχει κάθε κόμβος, το πλήθος των τυχαίων ψηφίων που χρησιμοποιούνται από κάποιον πιθανοτικό πράκτορα και το πλήθος και είδος των σφαλμάτων που μπορούν να παρουσιαστούν στο μοντέλο και πρέπει να ανέχεται ο αλγόριθμος.

3.3 Βασικά προβλήματα με κινητούς πράκτορες

3.3.1 Το πρόβλημα της συνάντησης κινητών πρακτόρων

Ένα από τα βασικά προβλήματα στην εξερεύνηση δικτύων χρησιμοποιώντας κινητούς πράκτορες είναι το *πρόβλημα της συνάντησης*. Όπως και η *εκλογή αρχηγού*, έτσι και η συνάντηση κινητών πρακτόρων είναι μια βασική διαδικασία χρήσιμη σε αλγόριθμους που απαιτούν το συγχρονισμό των πρακτόρων, την ανταλλαγή πληροφοριών, και την κατανομή αρμοδιοτήτων. Η έρευνα σε αυτό το πρόβλημα έχει επικεντρωθεί κυρίως στις ικανότητες, την μνήμη και την αρχική γνώση για την τοπολογία του δικτύου που πρέπει να έχουν οι πράκτορες για να συναντηθούν σε κάποιον κόμβο. Ένα σημαντικό ερώτημα που έχει μελετηθεί είναι το εξής: ποιες είναι οι αναγκαίες προϋποθέσεις για να είναι εφικτή η συνάντηση;

Ανάλογα με τις ικανότητες, τη μνήμη και την αρχική γνώση των πρακτόρων, το πρόβλημα της συνάντησης μπορεί να λυθεί εύκολα σε δίκτυα που έχουν ασυμμετρίες (π.χ., όταν υπάρχει ένας κόμβος που διαφέρει από τους υπόλοιπους) ακόμη και σε ασύγχρονα μοντέλα αφού σε αυτές τις περιπτώσεις οι πράκτορες μπορούν να πάρουν οδηγίες να συναντηθούν στον μοναδικό διαφορετικό κόμβο. Όταν όμως τα δίκτυα δεν έχουν τέτοιες ασυμμετρίες τότε η επίλυση του προβλήματος της συνάντησης είναι δύσκολη και συχνά αδύνατη.

3.3.2 Το πρόβλημα της ανακάλυψης ενός εχθρικού κόμβου σε δίκτυο

Στα κατανεμημένα περιβάλλοντα με κινητούς πράκτορες ένα από τα πιο σημαντικά ζητήματα είναι η ασφάλεια. Δύο είναι οι πιο σημαντικές απειλές: ένας εχθρικός πράκτορας που μπορεί να κινείται στο δίκτυο και μια στατική εχθρική διαδικασία που βρίσκεται σε κάποιο κόμβο του δι-

κτύου. Ένα από τα πιο μελετημένα μοντέλα για στατικές εχθρικές διαδικασίες είναι η μαύρη τρύπα (*black hole*) την οποία εισήγαγαν οι S. Dobrev, P. Flocchini, G. Prencipe και N. Santoro το 2001 ([Dobrev et al., 2001]). Η μαύρη τρύπα είναι ένας εχθρικός κόμβος του δικτύου που καταστρέφει οποιοδήποτε κινητό πράκτορα επισκέπτεται τον κόμβο, χωρίς να αφήνει κανένα ίχνος. Ο σκοπός του προβλήματος *Black Hole Search* είναι η εύρεση αυτού του κόμβου χωρίς την απώλεια πολλών πρακτόρων και η προσπάθεια είναι να βρεθούν οι ελάχιστες συνθήκες κάτω από τις οποίες το πρόβλημα μπορεί να λυθεί και μάλιστα όσο το δυνατόν γρηγορότερα.

3.3.3 Το πρόβλημα του καθαρισμού ενός μολυσμένου δικτύου

Η περίπτωση των εχθρικών κινητών πρακτόρων (που αναπαριστούν ιούς που μπορούν να κινούνται και να μολύνουν όποιον κόμβο του δικτύου επισκέπτονται) έχει μελετηθεί. Ένα σημαντικό πρόβλημα είναι ο καθαρισμός ενός μολυσμένου δικτύου με τη βοήθεια κινητών πρακτόρων που έχουν την ικανότητα να καθαρίζουν μολυσμένους κόμβους και να εμποδίζουν την επαναμόλυνση καθαρών περιοχών. Το πρόβλημα αυτό είναι ισοδύναμο με το πρόβλημα της εύρεσης ενός εισβολέα που κινείται στο δίκτυο.

3.4 Σχόλια και βιβλιογραφικές παρατηρήσεις

Υπάρχει μια πληθώρα άρθρων στα οποία γίνεται χρήση κινητών πρακτόρων σε κατανεμημένα συστήματα. Τα βιβλία [Milojičić et al., 1999] και [Weiss, 1999] αποτελούν μια καλή αρχή.

Ένα άλλο σχετικό και πολύ ενδιαφέρον πρόβλημα, είναι το πρόβλημα της εξερεύνησης στο οποίο ο σκοπός είναι η σχεδίαση ενός αλγόριθμου που επιτρέπει σε έναν πράκτορα να επισκεφτεί όλους τους κόμβους ή/και τις ακμές ενός γραφήματος [Kranakis et al., 2006, Diks et al., 2004, Deng and Papadimitriou, 1999].

Οι ομάδες κινητών πρακτόρων έχουν επίσης χρησιμοποιηθεί για την επίλυση προβλημάτων όπως η εξαγωγή αποφάσεων για την αποτελεσματική εξερεύνηση ενός άγνωστου περιβάλλοντος (π.χ., βλέπε άρθρο [Rajnarayan and Ghose, 2003]) και για την εξερεύνηση με τη βοήθεια χάρτη (π.χ., βλέπε άρθρο [Das et al., 2007]).

Δεν είναι εύκολο να καταλήξει κανείς σε έναν ικανοποιητικό και ακριβή ορισμό για τον κινητό πράκτορα. Σύμφωνα με τον Shoham [Shoham, 1997], στην τεχνολογία λογισμικού, ένας κινητός πράκτορας είναι μια οντότητα λογισμικού που λειτουργεί συνεχώς και αυτόνομα σε ένα περιβάλλον που συχνά περιλαμβάνει και άλλους πράκτορες και διεργασίες. Σύμφωνα με το άρθρο [Bradshaw, 1997], ένας πράκτορας που βρίσκεται σε ένα περιβάλλον με άλλους πράκτορες και διεργασίες, αναμένεται να μπορεί να επικοινωνήσει και να συνεργαστεί και ίσως να μπορεί να κινηθεί. Εναλλακτικά, μπορούμε να ορίσουμε τους κινητούς πράκτορες αναθέτοντάς τους ιδιότητες και χαρακτηρίζοντας τις επιθυμητές ικανότητες που πρέπει να έχουν για να μπορούν να λειτουργήσουν στο περιβάλλον που βρίσκονται (βλέπε για παράδειγμα τα άρθρα [Wooldridge, 1999, Wooldridge, 2002]).

Βιβλιογραφία

[Bradshaw, 1997] Bradshaw, J. (1997). *Software Agents*. The MIT Press.

[Das et al., 2007] Das, S., Flocchini, P., Kutten, S., Nayak, A., and Santoro, N. (2007). Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1-3):34–48.

[Deng and Papadimitriou, 1999] Deng, X. and Papadimitriou, C. H. (1999). Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297.

[Diks et al., 2004] Diks, K., Fraigniaud, P., Kranakis, E., and Pelc, A. (2004). Tree exploration with little memory. *Journal of Algorithms*, 51:38–63.

[Dobrev et al., 2001] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. (2001). Mobile agents search for a black-hole in an anonymous ring. In *Proceedings of 15th International Symposium on Distributed Computing*, pages 166–179.

[Flocchini et al., 1998] Flocchini, P., Mans, B., and Santoro, N. (1998). Sense of direction: Definitions, properties, and classes. *Networks*, 32(3):165–180.

[Kranakis et al., 2006] Kranakis, E., Krizanc, D., and Rajsbaum, S. (2006). Mobile agent rendezvous. In *Proceedings of 13th International Colloquium on Structural Information and Communication Complexity*, pages 1–9.

[Lynch, 1996] Lynch, N. (1996). *Distributed Algorithms*. Morgan Kaufman.

[Milojičić et al., 1999] Milojičić, D., Chauhan, D., and LaForge, W. (1999). *Mobility: Processes, Computers and Agents*. ACM Press.

[Milojičić et al., 1998] Milojičić, D. S., LaForge, W., and Chauhan, D. (1998). Mobile objects and agents (MOA). In *Proc. of 4th Conf. on USENIX Conf. on Object-Oriented Technologies and Systems-Volume 4*, pages 1–14.

[Rajnarayan and Ghose, 2003] Rajnarayan, D. G. and Ghose, D. (2003). Multiple agent team theoretic decision-making for searching unknown environments. In *Proc. 42nd IEEE Conference on Decision and Control*, pages 2543–2548.

[Roy and Dudek, 2001] Roy, N. and Dudek, G. (2001). Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11:117–136.

[Santoro, 2006] Santoro, N. (2006). *Design and analysis of distributed algorithms*. Wiley-Blackwell.

[Shannon, 1951] Shannon, C. E. (1951). Presentation of a maze-solving machine. In *Proceedings of 8th Conference of the Josiah Macy Jr. Found. (Cybernetics)*, pages 173–180.

[Shoham, 1997] Shoham, Y. (1997). An overview of agent-oriented programming. In Bradshaw, J. M., editor, *Software Agents*, chapter 13, pages 271–290. AAAI Press / The MIT Press.

[Tennenhouse et al., 1997] Tennenhouse, D. L., Smith, J. M., Sincoskie, W. D., Wetherall, D. J., and Minden, G. J. (1997). A survey of active network research. *IEEE communications Magazine*, 35(1):80–86.

[Tesfatsion, 2002] Tesfatsion, L. (2002). Agent-based computational economics: Growing economies from the bottom up. *Artificial Life*, 8(1):55–82.

[Weiss, 1999] Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press.

[Wooldridge, 1999] Wooldridge, M. (1999). Intelligent agents. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 27–77. The MIT Press.

[Wooldridge, 2002] Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. Wiley.

ΚΕΦΑΛΑΙΟ 4

Το Πρόβλημα της Συνάντησης δύο Κινητών Πρακτόρων

Σε αυτό το κεφάλαιο γίνεται εκτενής αναφορά στην επίλυση του προβλήματος της συνάντησης δύο κινητών πρακτόρων σε διαφορετικές τοπολογίες δικτύων, όπως δακτύλιους και τορικά (tori) δίκτυα. Παρουσιάζονται αρνητικά αποτελέσματα (μοντέλα στα οποία το πρόβλημα της συνάντησης είναι μη-επιλύσιμο) και αναλύονται ντετερμινιστικοί αλγόριθμοι σε συγχρονισμένα δίκτυα για πράκτορες που έχουν μοντελοποιηθεί με μηχανές Turing και για πεπερασμένα αυτόματα χωρίς μνήμη. Παρουσιάζονται επίσης αλγόριθμοι για πράκτορες που μπορούν να αφήσουν μηνύματα πάνω στους κόμβους του δικτύου.

4.1 Εισαγωγή

Ένα βασικό πρόβλημα για κάθε σύστημα με πολλούς πράκτορες είναι το πρόβλημα της συνάντησης. Η συνάντηση μεταξύ πρακτόρων (με σκοπό το συγχρονισμό τους, την ανταλλαγή πληροφοριών, το μοίρασμα καθηκόντων, κλπ) είναι συνήθως μια απαραίτητη βασική διαδικασία, που αποτελεί προϋπόθεση για την επιτυχία πιο πολύπλοκων αλγόριθμων για εφαρμογές όπως η αναζήτηση πληροφορίας σε ένα δίκτυο, η εξερεύνηση του δικτύου, κλπ. Με δεδομένο ένα μοντέλο για τους πράκτορες και το δίκτυο, οι πράκτορες είναι αρχικά τοποθετημένοι με έναν τυχαίο τρόπο σε κόμβους του δικτύου. Οι πράκτορες συναντιούνται αν μετά την εκτέλεση των αλγορίθμων τους για πεπερασμένο χρόνο βρίσκονται στον ίδιο κόμβο κατά την ίδια χρονική στιγμή. Όπως συμβαίνει συχνά, ενδιαφερόμαστε για την αναγνώριση και ανάλυση των περιπτώσεων εκείνων για τις οποίες το πρόβλημα είναι δύσκολο. Ιδιαίτερο ενδιαφέρον έχει η επίλυση του προβλήματος στις περιπτώσεις συμμετρίας ανώνυμων πρακτόρων σε ένα ανώνυμο δίκτυο.

Στις περιπτώσεις που το πρόβλημα της συνάντησης λύνεται, ενδιαφερόμαστε για τη σύγκριση της απόδοσης διαφορετικών αλγόριθμων. Ιδιαίτερα μας ενδιαφέρει ο αριθμός των κινήσεων που απαιτούνται για τη συνάντηση. Σε κάποιες περιπτώσεις παίζει ρόλο αν οι πράκτορες ξεκινούν την εκτέλεση του αλγόριθμου ταυτόχρονα ή υπάρχει καθυστέρηση. Επίσης μας ενδιαφέρει το μέγεθος της μνήμης που απαιτείται να έχουν οι πράκτορες. Στην καλύτερη περίπτωση, θα θέλαμε να σχεδιά-

σουμε έναν αλγόριθμο για πράκτορες με σταθερή μνήμη (ανεξάρτητη από το μέγεθος του δικτύου) ο οποίος λύνει το πρόβλημα μετά από γραμμικό αριθμό κινήσεων. Η επίτευξη και των δύο αυτών σκοπών δεν είναι πάντα δυνατή.

Στο μοντέλο των ντετερμινιστικών, συγχρονισμένων και ανώνυμων πρακτόρων, απαιτείται κάτι για να σπάσει τις συμμετρίες μεταξύ των πρακτόρων, αλλιώς δύο πράκτορες που ξεκινούν στην ίδια κατάσταση θα εκτελούν συγχρονισμένα τις ίδιες οδηγίες και έτσι δεν θα μπορέσουν να συναντηθούν ποτέ. Ένας μηχανισμός για το σπάσιμο των συμμετριών, είναι η χρήση σημαδιών (βλέπε [Baston and Gal, 2001]) για το μαρκάρισμα των κόμβων του δικτύου. Ανάλογα με το μοντέλο, τα σημάδια μπορεί να είναι στατικά ή μετακινήσιμα, δηλαδή να μπορούν να χρησιμοποιηθούν για το μαρκάρισμα διαφορετικών κόμβων σε διαφορετικές στιγμές ή από τη στιγμή που τοποθετούνται σε ένα έναν κόμβο παραμένουν μόνιμα εκεί. Τα σημάδια συνήθως είναι μη-διακρίσιμα, δηλαδή τα σημάδια που τοποθετήθηκαν από διαφορετικούς πράκτορες ή από τον ίδιο πράκτορα σε διαφορετικές στιγμές δεν είναι διακρίσιμα. Το πλήθος των σημαδιών που διαθέτει ένας πράκτορας εξαρτάται από τη μνήμη που παρέχουν οι κόμβοι και είναι άλλο ένα αγαθό του οποίου την ποσότητα μας ενδιαφέρει να μελετήσουμε.

4.2 Συνάντηση με χρήση σημαδιών σε δακτύλιο

Ίσως η πιο απλή ενδιαφέρουσα περίπτωση είναι εκείνη κατά την οποία δύο πράκτορες προσπαθούν να συναντηθούν σε ένα δακτύλιο. Περιγράφουμε παρακάτω ένα βασικό μοντέλο σε έναν συγχρονισμένο, ανώνυμο και προσανατολισμένο δακτύλιο (βλέπε για παράδειγμα [Attiya et al., 1988]), όπου:

1. οι κόμβοι δεν έχουν διακρίσιμες ταυτότητες, δηλαδή οι πράκτορες δεν μπορούν να ξεχωρίσουν δύο διαφορετικούς κόμβους,

2. οι υπολογισμοί και οι διασχίσεις ακμών γίνονται συγχρονισμένα,

3. οι ακμές που προσπίπτουν σε κάθε κόμβο έχουν ετικέττες `left` και `right` με ένα καθολικά συνεπή τρόπο.

Μοντελοποιούμε τους πράκτορες ως ντετερμινιστικά πεπερασμένα αυτόματα $A =< S, \delta, s_0 >$, όπου S είναι το σύνολο των καταστάσεων του αυτομάτου που περιλαμβάνει την αρχική κατάσταση s_0 και την ειδική κατάσταση `halt`, και $\delta : S \times P \to S \times M$ είναι η συνάρτηση μετάβασης, όπου $P = \{\texttt{present}, \texttt{notpresent}\}$ είναι ένα κατηγόρημα που κωδικοποιεί την παρουσία άλλων πρακτόρων ή όχι σε έναν κόμβο, ενώ $M = \{\texttt{left}, \texttt{right}\}$ αναπαριστά τις δυνατές κινήσεις ενός πράκτορα. Κατά τη διάρκεια ενός συγχρονισμένου βήματος, ανάλογα με την κατάσταση και την παρουσία ή όχι άλλων πρακτόρων στον ίδιο κόμβο, ο πράκτορας χρησιμοποιεί τη συνάρτηση δ για να αλλάξει την κατάστασή του και να κινηθεί είτε προς την ακμή με ετικέττα `left` είτε προς εκείνη με ετικέττα `right`. Οι πράκτορες σταματούν την εκτέλεση του αλγόριθμου αμέσως μόλις ανιχνεύσουν την παρουσία άλλου πράκτορα στον κόμβο.

Η πρώτη ερώτηση στην οποία καλείται κανείς να απαντήσει σε αυτό το στιγμιότυπο του προβλήματος είναι αν η επίτευξη της συνάντησης είναι πάντα δυνατή. Είναι αρκετά εύκολο να δει κανείς ότι αν οι πράκτορες έχουν αρχικά περιττή απόσταση σε έναν δακτύλιο άρτιου μεγέθους, τότε δεν μπορούν να συναντηθούν στο παραπάνω μοντέλο αφού είναι υποχρεωμένοι να κινούνται σε κάθε βήμα και συνεπώς θα διατηρούν την μεταξύ τους περιττή απόσταση για πάντα. Υπάρχουν διάφοροι τρόποι να ξεπεράσουμε αυτό και ίσως ο ευκολότερος είναι να προσθέσουμε στους πράκτορες τη δυνατότητα να παραμένουν στον ίδιο κόμβο (`stay`). Ένας εναλλακτικός τρόπος είναι να επιτρέψουμε στους πράκτορες να συναντιούνται στις ακμές. Μερικές φορές θεωρούμε ότι οι πράκτορες είναι αρχικά τοποθετημένοι σε άρτια απόσταση σε ένα δακτύλιο άρτιου μεγέθους. Ακόμη όμως και με αυτές τις συνθήκες η επίτευξη της συνάντησης δεν είναι πάντα εφικτή όπως θα δούμε σε επόμενα κεφάλαια.

Σε αυτή την ενότητα μελετούμε το πρόβλημα της συνάντησης δύο πανομοιότυπων κινητών πρακτόρων σε έναν ανώνυμο, συγχρονισμένο και πιθανά προσανατολισμένο δακτύλιο που αποτελείται από n κόμβους. Οι κινητοί πράκτορες δεν χρησιμοποιούν πιθανοτικούς αλγόριθμους ή διαφορετικούς ντετερμινιστικούς αλγόριθμους για να σπάσουν τις συμμετρίες. Εκτελούν τον **ίδιο** ντετερμινιστικό αλγόριθμο και χρησιμοποιούν πανομοιότυπα σημάδια τα οποία μπορούν να τοποθετούν σε κόμβους του δακτυλίου. Αναλύουμε επίσης το πρόβλημα ανάλογα με το αν οι πράκτορες μπορούν να μετακινήσουν τα σημάδια ή όχι.

Αρχικά μελετούμε πότε τα πανομοιότυπα μη-μετακινήσιμα σημάδια δεν μπορούν να χρησιμοποιηθούν για να λυθεί το πρόβλημα. Αναλύουμε το πρόβλημα με βάση τις εξής δύο παραμέτρους:

1. την αρχική απόσταση d μεταξύ των δύο πρακτόρων, και

2. το μέγεθος (πλήθος κόμβων) n του δακτυλίου.

Θεωρούμε ότι τα d και n είναι άρτιοι αριθμοί. Αποδεικνύεται ότι η επίτευξη της συνάντησης εξαρτάται από τη δυνατότητα εύρεσης ασυμμετρίας η οποία με τη σειρά της εξαρτάται από το αν $d < \frac{n}{2}$ ή όχι. Στη συνέχεια μελετούμε το πρόβλημα της συνάντησης με δεδομένο ότι $d < \frac{n}{2}$. Κάθε πράκτορας τοποθετεί το μη-μετακινήσιμο σημάδι του στην αντίστοιχη εναρκτήρια κορυφή στον δακτύλιο και μετακινείται με ταχύτητα μιας κορυφής ανά μονάδα χρόνου. Θεωρούμε ότι οι πράκτορες ξέρουν ότι δρουν σε δακτύλιο και γνωρίζουν ότι $d < \frac{n}{2}$, αλλά ίσως να μην γνωρίζουν τα d, n, καθώς και τον προσανατολισμό του δακτυλίου.

Ο Πίνακας 4.1 απεικονίζει πάνω και κάτω φράγματα της χρονικής πολυπλοκότητας της συνάντησης δύο κινητών πρακτόρων που χρησιμοποιούν πανομοιότυπα μη-μετακινήσιμα σημάδια και ξέρουν ότι $d < \frac{n}{2}$. Τα κάτω φράγματα ισχύουν ακόμη και όταν οι πράκτορες έχουν απεριόριστη μνήμη, ενώ τα άνω φράγματα ισχύουν όταν το μέγεθος της μνήμης των πρακτόρων είναι όπως φαίνεται στην τελευταία στήλη του πίνακα.

Εάν οι πράκτορες ξέρουν τουλάχιστον κάποιο από τα d, n, τότε το κάτω φράγμα της χρονικής πολυπλοκότητας είναι $\frac{3n}{4}$. Εάν τα d, n είναι άγνωστα, τότε το κάτω φράγμα της χρονικής πολυπλοκότητας αυξάνει σε $\frac{5n}{4}$.

Πίνακας 4.1: Φράγματα στη χρονική πολυπλοκότητα του προβλήματος της συνάντησης δύο κινητών πρακτόρων σε απόσταση $d < \frac{n}{2}$ σε έναν ανώνυμο συγχρονισμένο δακτύλιο n κόμβων

Αρχική Γνώση			Κάτω	Πάνω	Απαιτήσεις
n	d	Προσανατολισμός	Φράγμα	Φράγμα	Μνήμης
Ναι	Ναι	Ναι	$3n/4$	$3n/4$	$O(\log d)$
Ναι	Ναι	Όχι	$3n/4$	$5n/6$	$O(\log d)$
Ναι	Όχι	Ναι	$3n/4$	$3n/4$	$O(\log n)$
Ναι	Όχι	Όχι	$3n/4$	$3n/4$	$O(\log n)$
Όχι	Ναι	Ναι	$3n/4$	$3n/4$	$O(\log d)$
Όχι	Ναι	Όχι	$3n/4$	$5n/6$	$O(\log d)$
Όχι	Όχι	Ναι	$5n/4$	$5n/4$	$O(\log n)$
Όχι	Όχι	Όχι	$5n/4$	$5n/4$	$O(\log n)$

Για να πάρουμε τα πάνω φράγματα της χρονικής πολυπλοκότητας, παρουσιάζουμε τέσσερις λύσεις στο πρόβλημα με δεδομένο ότι $d < \frac{n}{2}$, χρησιμοποιώντας πανομοιότυπα μη-μετακινήσιμα σημάδια. Αυτές οι λύσεις, με μία εξαίρεση, αποδεικνύουν ότι τα κάτω φράγματα στον Πίνακα 4.1 είναι σφιχτά. Η εξαίρεση εμφανίζεται όταν το d είναι γνωστό αλλά ο προσανατολισμός του δακτυλίου είναι άγνωστος. Σε αυτές τις περιπτώσεις, το κάτω φράγμα στη χρονική πολυπλοκότητα είναι $\frac{3n}{4}$ αλλά το αντίστοιχο άνω φράγμα είναι $\frac{5n}{6}$.

Ενώ τα αποτελέσματα στον Πίνακα 4.1 θεωρούν ότι οι πράκτορες γνωρίζουν ότι $d < \frac{n}{2}$, οι λύσεις μπορούν εύκολα να μετατραπούν ώστε οι πράκτορες να σταματούν αν ισχύει $d = \frac{n}{2}$. Χρειάζεται απλά κάθε πράκτορας να έχει μνήμη μεγέθους $O(\log n)$. Η τελευταία γραμμή στον Πίνακα 4.1 αντιστοιχεί στην περίπτωση που οι πράκτορες δεν γνωρίζουν τα n, d, ούτε τον προσανατολισμό του δακτυλίου. Η λύση που μας έδωσε το άνω φράγμα απαιτεί $O(\log n)$ μνήμη και $\frac{5n}{4}$ χρόνο. Αργότερα παρουσιάζουμε μια ακόμη λύση για την περίπτωση που οι πράκτορες δεν γνωρίζουν τα n, d, ούτε τον προσανατολισμό του δακτυλίου, αλλά αυτή η λύση απαιτεί $O(\log \log n)$ μνήμη και $O(\frac{n \log n}{\log \log n})$ χρόνο. Ενώ αυτός ο τελευταίος αλγόριθμος μας δείχνει ότι η μνήμη των πρακτόρων μπορεί να μειωθεί με κόστος την αύξηση της χρονικής πολυπλοκότητας, αποδεικνύουμε ότι αυτό δεν μπορεί να γίνει απεριόριστα. Αν οι πράκτορες έχουν $O(1)$ μνήμη και δεν γνωρίζουν τα n, d, ούτε τον προσανατολισμό του δακτυλίου τότε δείχνουμε ότι είναι αδύνατη η κατασκευή ενός αλγόριθμου με βάση τον οποίον οι πράκτορες να σταματούν αν η συνάντηση είναι αδύνατη, ή αλλιώς να συναντιούνται.

Αυτό το τελευταίο αποτέλεσμα σε συνδυασμό με τα άνω φράγματα που παρουσιάζονται νωρίτερα μας δείχνουν ότι για να λυθεί το πρόβλημα της συνάντησης όταν οι πράκτορες έχουν μόνο $O(1)$ μνήμη απαιτείται μια αλλαγή στο μοντέλο. Έστω λοιπόν ότι τα σημάδια είναι μετακινήσιμα, δηλαδή κάθε φορά που ένας πράκτορας συναντά ένα σημάδι μπορεί να το μετακινήσει σε κάποιον άλλο κόμβο. Σε αυτήν την περίπτωση μπορούμε να σχεδιάσουμε έναν αλγόριθμο που απαιτεί μόνο $O(1)$ μνήμη, και λύνει το πρόβλημα χρησιμοποιώντας πανομοιότυπα αλλά μετακινήσιμα σημάδια.

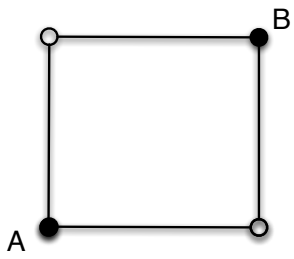
4.2.1 Ένα μη-μετακινήσιμο σημάδι

Αρχικά σχεδιάζουμε και αναλύουμε αλγόριθμους στους οποίους κάθε πράκτορας χρησιμοποιεί ένα μη-μετακινήσιμο σημάδι. Υπενθυμίζουμε ότι τα σημάδια είναι πανομοιότυπα.

4.2.1.1 Η επιλυσιμότητα του προβλήματος της συνάντησης

Πριν δώσουμε αλγόριθμους που λύνουν το πρόβλημα χρησιμοποιώντας πανομοιότυπα μη-μετακινήσιμα σημάδια είναι ανάγκη να αναγνωρίσουμε τις συνθήκες κάτω από τις οποίες είναι δυνατόν να γίνει αυτό. Στα παραδείγματα που ακολουθούν γίνεται φανερή η εξάρτηση της επιλυσιμότητας του προβλήματος από το άρτιο ή περιττό πλήθος κόμβων και την ύπαρξη σημαδιών.

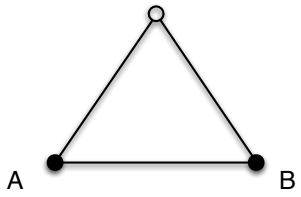
Παράδειγμα 4.1 Έστω ένας ανώνυμος, συγχρονισμένος και προσανατολισμένος δακτύλιος με τέσσερις κόμβους ($n = 4$) και δύο πανομοιότυποι πράκτορες A, B, οι οποίοι αρχικά βρίσκονται σε δύο διαφορετικούς κόμβους που απέχουν απόσταση δύο (βλέπε Σχήμα 4.1). Εάν οι πράκτορες



Σχήμα 4.1: Δύο κινητοί πράκτορες σε απόσταση δύο σε έναν δακτύλιο τεσσάρων κόμβων.

είναι συγχρονισμένοι και εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο, δεν θα καταφέρουν ποτέ να συναντηθούν αφού κινούνται ταυτόχρονα στην ίδια κατεύθυνση με την ίδια ταχύτητα. Ακόμη και αν οι πράκτορες χρησιμοποιούν πανομοιότυπα μετακινήσιμα σημάδια η επίλυση του προβλήματος είναι αδύνατη.

Παράδειγμα 4.2 Θεωρήστε δύο κινητούς πράκτορες σε έναν προσανατολισμένο δακτύλιο τριών κόμβων (βλέπε Σχήμα 4.2). Εάν δεν υπάρχουν διαθέσιμα σημάδια, ο μόνος τρόπος για να σπάσει η συμμετρία με την οποία κινούνται οι πράκτορες είναι να κινηθούν σε αντίθετες κατευθύνσεις. Αυτό όμως δεν είναι δυνατό, δεδομένου ότι οι πράκτορες είναι πανομοιότυποι και συνεπώς θα επιλέγουν πάντα την ίδια κατεύθυνση για να κινηθούν. Η συνάντηση μπορεί να επιτευχθεί όμως στην περίπτωση που οι πράκτορες έχουν από ένα μη-μετακινήσιμο σημάδι το οποίο αφήνουν στον κόμβο που βρίσκονται αρχικά. Ο αλγόριθμος συνεχίζει ζητώντας από κάθε πράκτορα να κινηθεί μέχρι να συναντήσει ένα σημάδι, να αλλάξει κατεύθυνση και να κινηθεί μέχρι να συναντήσει τον άλλο πράκτορα. Υπενθυμίζουμε ότι οι πράκτορες ανιχνεύουν αμέσως την παρουσία κάποιου άλλου



Σχήμα 4.2: Δύο κινητοί πράκτορες σε απόσταση ένα, σε έναν δακτύλιο τριών κόμβων.

πράκτορα στον κόμβο που βρίσκονται. Έτσι λοιπόν η συνάντηση σε αυτήν την περίπτωση θα συμβεί μετά από μόλις δύο βήματα: ένας από τους πράκτορες θα συναντήσει ένα σημάδι μετά από δύο βήματα ενώ ο άλλος μετά από ένα βήμα.

Το πρώτο μας αποτέλεσμα είναι μια γενίκευση του πρώτου παραδείγματος, δηλαδή αν το n είναι άρτιος αριθμός και $d = \frac{n}{2}$, τότε είναι αδύνατη η κατασκευή ενός αλγόριθμου, για τη λύση του προβλήματος, που χρησιμοποιεί πανομοιότυπα μη-μετακινήσιμα σημάδια.

Για να παρουσιάσουμε μια τυπική απόδειξη του παραπάνω ισχυρισμού είναι ανάγκη να ορίσουμε επακριβώς τις δυνατότητες των πρακτόρων. Οι ατομικές λειτουργίες που μπορεί να εκτελέσει ένας πράκτορας ακολουθώντας έναν αλγόριθμο είναι:

RT**:** άφησε στον κόμβο το σημάδι

QT**:** ανίχνευσε τον κόμβο για την ύπαρξη σημαδιού

QMA**:** ανίχνευσε τον κόμβο για την ύπαρξη άλλου πράκτορα

MCL**:** μετακινήσου στο γειτονικό κόμβο δεξιόστροφα

MCC**:** μετακινήσου στο γειτονικό κόμβο αριστερόστροφα

DM**:** μείνε ακίνητος

$IC(count)$**:** αύξησε κατά ένα τη μεταβλητή $count$ (αν επαρκεί η μνήμη)

Επιπλέον των ατομικών λειτουργιών που περιγράφτηκαν παραπάνω, ένας αλγόριθμος μπορεί να περιέχει τις δομές:

$do - until$ **loop:** ο πράκτορας επαναλαμβάνει μια πεπερασμένη ακολουθία λειτουργιών μέχρι να συμβεί ένα δεδομένο γεγονός.

$if - then$ **statement:** εάν ισχύσει μια δεδομένη συνθήκη, ο πράκτορας θα εκτελέσει μια πεπερασμένη ακολουθία λειτουργιών.

Χρησιμοποιώντας τους παραπάνω ορισμούς μπορούμε να περιγράψουμε ακριβώς τί μπορούν να κάνουν οι πράκτορες. Τα δύο λήμματα που ακολουθούν θα μας χρειαστούν στην απόδειξη του βασικού αποτελέσματος (Θεώρημα 4.5).

Λήμμα 4.3 Θεωρήστε ότι οι πράκτορες βρίσκονται αρχικά σε μια απόσταση $d > 0$ στο δακτύλιο. Εάν οι πράκτορες εκτελέσουν τον ίδιο ντετερμινιστικό αλγόριθμο ξεκινώντας την ίδια στιγμή και ο αλγόριθμος περιορίζεται σε μια πεπερασμένη ακολουθία από ατομικές λειτουργίες, δηλαδή δεν υπάρχουν $do - until$ loops ή $if - then$ statements, τότε η απόσταση d μεταξύ των πρακτόρων δεν θα αλλάξει.

Απόδειξη. Έστω ένας αλγόριθμος που αποτελείται από μία μόνο ατομική λειτουργία. Οποιαδήποτε από τις ατομικές λειτουργίες $\{RT, QT, QMA, IC(c), DM, MCL, MCC\}$ δεν μπορεί να αλλάξει την απόσταση d, καθώς είτε οι πράκτορες παραμένουν ακίνητοι σε διαφορετικούς κόμβους είτε κινούνται ταυτόχρονα σε κάποιον γειτονικό κόμβο προς την ίδια κατεύθυνση. Έστω πως αυτό ισχύει για έναν οποιοδήποτε αλγόριθμο που περιλαμβάνει μια ακολουθία από k ατομικές λειτουργίες.

Θεωρήστε έναν αλγόριθμο ο οποίος αποτελείται από μια ακολουθία $k + 1$ ατομικών λειτουργιών. Αφού η απόσταση d παραμένει αμετάβλητη μετά από k ατομικές λειτουργίες, θα παραμείνει αμετάβλητη αν η $k + 1$-στή λειτουργία είναι μια από τις $\{RT, QT, QMA, IC(c), DM\}$ αφού οι πράκτορες παραμένουν ακίνητοι μετά από αυτές τις ατομικές λειτουργίες. Επίσης η τιμή του d δεν αλλάζει αν η $k + 1$-στή λειτουργία είναι μια από τις MCL, MCC αφού οι πράκτορες μετακινούνται στους γειτονικούς τους κόμβους προς την ίδια κατεύθυνση. Συνεπώς η απόσταση d παραμένει αμετάβλητη μετά από $k + 1$ ατομικές λειτουργίες και έτσι ολοκληρώνεται η απόδειξη. ■

Το Λήμμα 4.3 δείχνει ότι αν οι δύο πράκτορες βρίσκονται αρχικά σε απόσταση $d = n/2$ και εκτελέσουν έναν αλγόριθμο ο οποίος περιλαμβάνει μόνο ατομικές λειτουργίες τότε οι πράκτορες δεν μπορούν να συναντηθούν. Έτσι λοιπόν δεν μπορεί να προκύψει συνάντηση αν δεν χρησιμοποιηθούν $do - until$ loops ή $if - then$ statements. Η χρήση όμως τέτοιων δομών είναι αναγκαία αλλά όχι ικανή για τη λύση του προβλήματος της συνάντησης.

Λήμμα 4.4 Έστω ότι οι πράκτορες βρίσκονται αρχικά σε απόσταση $d = \frac{n}{2}$ κόμβων στον δακτύλιο έτσι ώστε *CLToToken* $=$ *CCToToken* $= 0$ και για τους δύο πράκτορες, όπου τα *CLToToken* και *CCToToken* αναπαριστούν για έναν δεδομένο πράκτορα την απόστασή του από τον πλησιέστερο κόμβο με σημάδι δεξιόστροφα και αριστερόστροφα αντίστοιχα. Αν t^* είναι η πρώτη χρονική στιγμή κατά τη διάρκεια εκτέλεσης του αλγόριθμου όπου η απόσταση d είναι διαφορετική από $n/2$, τότε t^* είναι επίσης η πρώτη χρονική στιγμή που οι αντίστοιχες τιμές των *CLToToken* (και συνεπώς *CCToToken*) για τους δύο πράκτορες διαφέρουν.

Απόδειξη. Για κάθε $t < t^*$, ισχύει $d = n/2$ έτσι ώστε είτε και οι δύο πράκτορες βρίσκονται σε διαφορετικούς κόμβους με σημάδια είτε κανείς πράκτορας δεν βρίσκεται σε κόμβο με σημάδι. Αν και οι δύο πράκτορες βρίσκονται σε διαφορετικούς κόμβους με σημάδια ισχύει *CLToToken* $=$ *CCToToken* $= 0$ και για τους δύο πράκτορες. Έστω ότι τη χρονική στιγμή t^*, κανείς πράκτορας δεν βρίσκεται σε

κόμβο με σημάδι. Επιπλέον, έστω ότι υπάρχει μια χρονική στιγμή $t' < t^*$ έτσι ώστε t' είναι η πρώτη χρονική στιγμή κατά την οποίαν οι τιμές των *CLToToken* των δύο πρακτόρων διαφέρουν, όπως και οι τιμές των αντίστοιχων *CCToToken*. Έστω $\{CLToToken_1, CCToToken_1\}$ και $\{CLToToken_2, CCToToken_2\}$ είναι οι αντίστοιχες τιμές. Έστω, χωρίς βλάβη της γενικότητας ότι $CLToToken_1 < CCToToken_2$. Αν και οι δύο πράκτορες μετακινηθούν $CLToToken_1$ κόμβους δεξιόστροφα, τότε ο ένας πράκτορας θα συναντήσει σημάδι ενώ ό άλλος όχι. Συνεπώς η απόσταση μεταξύ των πρακτόρων δεν ήταν $d = n/2$ το οποίο είναι άτοπο και έτσι ολοκληρώνεται η απόδειξη. ■

Θεωρήστε τις συνθήκες που βρίσκονται στις δομές $do - until$ και $if - then$. Οι συνθήκες αυτές καθορίζουν αντίστοιχα πόσες επαναλήψεις θα γίνουν ή αν το σώμα της δομής $if - then$ θα εκτελεστεί. Σε οποιαδήποτε περίπτωση η αποτίμηση μιας συνθήκης τη χρονική στιγμή t είναι μια συνάρτηση του γνωστού ιστορικού για τον δεδομένο πράκτορα τη χρονική στιγμή $t - 1$. Δύο πράκτορες δεν μπορούν να αποτιμήσουν την ίδια συνθήκη στο χρόνο t και να λάβουν διαφορετικές τιμές εκτός αν τα γνωστά ιστορικά τους διαφέρουν τη στιγμή $t - 1$.

Θεώρημα 4.5 ([Sawchuk, 2004]) Έστω δύο πανομοιότυποι πράκτορες σε έναν ανώνυμο δακτύλιο n κόμβων (όπου το n είναι άρτιος αριθμός). Οι πράκτορες βρίσκονται αρχικά σε απόσταση $d = \frac{n}{2}$ στο δακτύλιο. Κάθε πράκτορας ξεκινά τοποθετώντας ένα σημάδι στον εναρκτήριό του κόμβο και αυτά τα πανομοιότυπα σημάδια παραμένουν εκεί για όλη τη διάρκεια του αλγόριθμου. Οι δύο πράκτορες δεν θα συναντηθούν αν χρησιμοποιούν τον ίδιο ντετερμινιστικό αλγόριθμο.

Απόδειξη. Η κατάσταση ενός πράκτορα περιλαμβάνει τις τιμές των *CLToToken* και *CCToToken*. Έστω το s_0^i αναπαριστά την κατάσταση του i-οστού πράκτορα τη στιγμή 0. Το ιστορικό του πράκτορα i την χρονική στιγμή t ορίζεται από την ακολουθία $H_t^i = s_0^i, s_1^i, ..., s_t^i$.

Έστω ότι οι δύο πράκτορες συναντιούνται τη στιγμή t. Αυτό σημαίνει ότι τη στιγμή t, ισχύει $d = 0$. Θεωρήστε την πρώτη χρονική στιγμή $t^* < t$, κατά την οποία η απόσταση d δεν είναι ίση με $n/2$. Το Λήμμα 4.4 δείχνει ότι αυτή είναι επίσης η πρώτη χρονική στιγμή κατά την οποία τα αντίστοιχα *CLToToken* και *CWToToken* των δύο πρακτόρων διαφέρουν. Την χρονική στιγμή t^*, η απόσταση d (και συνεπώς τα *CLToToken* και *CCToToken*) αλλάζουν διότι είτε:

1. ο ένας πράκτορας εκτέλεσε τη λειτουργία DM και ο άλλος πράκτορας τη λειτουργία MCL ή MCC έτσι ώστε η απόσταση d μεταβλήθηκε κατά ένα, είτε,

2. ο ένας πράκτορας εκτέλεσε τη λειτουργία MCL και ο άλλος πράκτορας τη λειτουργία MCC έτσι ώστε η απόσταση d μεταβλήθηκε κατά δύο.

Αφού η απόσταση d μεταβλήθηκε τη στιγμή t^*, οι πράκτορες αποτίμησαν κάποια συνθήκη βρίσκοντας διαφορετικές τιμές. Από αυτό προκύπτει ότι τα ιστορικά των πρακτόρων διέφεραν τη στιγμή $t^* - 1$. Έστω $t' \leq t^*$ η πρώτη στιγμή κατά την οποίαν τα ιστορικά των δύο πρακτόρων διαφέρουν. Όπως προκύπτει από το Λήμμα 4.3 οι καταστάσεις των δύο πρακτόρων διαφέρουν τη στιγμή t' μόνο αν μια συνθήκη που αποτιμήθηκε εκείνη τη στιγμή έδωσε διαφορετικές τιμές για τους δύο πράκτορες το οποίο σημαίνει ότι τα ιστορικά των δύο πρακτόρων διέφεραν τη χρονική

στιγμή $t' - 1$. Αυτό αντίκειται στο γεγονός ότι η t' ήταν η πρώτη χρονική στιγμή που τα ιστορικά διέφεραν. ■

4.2.1.2 Η χρονική πολυπλοκότητα του προβλήματος της συνάντησης

Ενώ από το Θεώρημα 4.5 αποδεικνύεται ότι τα μη-μετακινήσιμα σημάδια δεν μπορούν να χρησιμοποιηθούν για να σπάσουν τις συμμετρίες όταν η αρχική απόσταση είναι $d = \frac{n}{2}$, τα Θεωρήματα 4.6 και 4.7 παρακάτω μας δίνουν πάνω και κάτω φράγματα για τη χρονική πολυπλοκότητα του προβλήματος χρησιμοποιώντας πανομοιότυπα μη-μετακινήσιμα σημάδια όταν $d < \frac{n}{2}$.

Θεώρημα 4.6 ([Kranakis et al., 2003]) Θεωρήστε δύο πανομοιότυπους πράκτορες τοποθετημένους σε απόσταση $d < \frac{n}{2}$ σε έναν ανώνυμο συγχρονισμένο δακτύλιο που αποτελείται από n κόμβους. Οι πράκτορες γνωρίζουν ότι το δίκτυο είναι δακτύλιος και ότι ισχύει $d < \frac{n}{2}$. Αν οι πράκτορες έχουν απεριόριστη μνήμη και εκτελούν ταυτόχρονα τον ίδιο ντετερμινιστικό αλγόριθμο για το πρόβλημα της συνάντησης τότε τα κάτω φράγματα χρονικής πολυπλοκότητας φαίνονται στον Πίνακα 4.1.

Απόδειξη. Θεωρήστε την περίπτωση στην οποία οι πράκτορες γνωρίζουν τη μεταξύ τους απόσταση d. Μετά από τη μετακίνηση ενός πράκτορα κατά d κόμβους σε κάποια κατεύθυνση, ο πράκτορας έχει αποκτήσει νέα δεδομένα αφού είτε βρίσκει ένα σημάδι ή μαθαίνει ότι το σημάδι βρίσκεται προς την άλλη κατεύθυνση από τον κόμβο που ξεκίνησε. Οι πράκτορες βρίσκονται ακόμη σε απόσταση d, και έτσι η συνάντηση απαιτεί από τους πράκτορες να μετακινηθούν για τουλάχιστον άλλους $\frac{d}{2}$ κόμβους. Συνεπώς, εφόσον οι πράκτορες πρέπει να κάνουν τουλάχιστον $\frac{3d}{2}$ βήματα για να συναντηθούν και το κάθε βήμα απαιτεί μία χρονική μονάδα, η επίτευξη της συνάντησης απαιτεί τουλάχιστον $\frac{3d}{2}$ χρονικές μονάδες. Η χειρότερη περίπτωση προκύπτει όταν $d = \frac{n}{2} - 1$, και η συνάντηση απαιτεί τουλάχιστον $\frac{3n}{4} - O(1)$ χρόνο. Έτσι προκύπτει το κάτω φράγμα για όλες τις περιπτώσεις όταν η απόσταση d είναι γνωστή.

Θεωρήστε τώρα ότι οι πράκτορες δεν γνωρίζουν την απόσταση d αλλά γνωρίζουν τον αριθμό κόμβων n του δακτυλίου. Ένας από τους πράκτορες μπορεί να βρεί ένα σημάδι αφού κινηθεί σε απόσταση d σε κάποια κατεύθυνση. Ο άλλος πράκτορας όμως μπορεί να μην έχει κάποια νέα πληροφορία μέχρι να φτάσει σε απόσταση $\frac{n}{2}$ κόμβων από τον κόμβο που ξεκίνησε. Σε όλη αυτήν τη διάρκεια, οι πράκτορες συνεχίζουν να βρίσκονται σε απόσταση d μεταξύ τους. Με δεδομένο ότι οι πράκτορες πρέπει να ταξιδέψουν μια απόσταση από τουλάχιστον $\frac{n+d}{2}$ κόμβους για να συναντηθούν, και η μετακίνηση από κόμβο σε κόμβο απαιτεί μια χρονική μονάδα, η επίτευξη της συνάντησης απαιτεί τουλάχιστον $\frac{n+d}{2}$ χρόνο. Έτσι παίρνοντας $d = \frac{n}{2} - 1$, η συνάντηση απαιτεί τουλάχιστον $\frac{3n}{4} - O(1)$ χρόνο το οποίο μας δίνει το κάτω φράγμα για όλες τις περιπτώσεις όπου το n είναι γνωστό.

Οι περιπτώσεις που απομένουν είναι εκείνες στις οποίες οι πράκτορες δεν γνωρίζουν ούτε το n ούτε το d. Ένας πράκτορας μπορεί να μετρήσει τον αριθμό m των κόμβων από την αφετηρία του έως το πρώτο σημάδι που συναντά. Δεν μπορεί όμως να αποφασίσει αν $m = d$ ή $m = n - d$ μέχρι να ταξιδέψει σε ολόκληρο τον δακτύλιο μετρώντας το n. Όταν οι πράκτορες επιστρέφουν

στους αντίστοιχους κόμβους από τους οποίους ξεκίνησαν βρίσκονται ακόμη σε απόσταση d. Με δεδομένα ότι οι πράκτορες πρέπει να ταξιδέψουν μια απόσταση τουλάχιστον $n + \frac{d}{2}$ κόμβων για να συναντηθούν και η μετακίνηση από κόμβο σε κόμβο χρειάζεται μια χρονική μονάδα, ο χρόνος που απαιτείται για τη συνάντηση είναι τουλάχιστον $n + \frac{d}{2}$ όταν τα n και d είναι άγνωστα. Παίρνοντας $d = \frac{n}{2} - 1$, βρίσκουμε ότι η συνάντηση απαιτεί τουλάχιστον $\frac{5n}{4} - O(1)$ χρόνο. Έτσι προκύπτει το κάτω φράγμα για τις περιπτώσεις που τα n και d είναι άγνωστα. ■

Τα πάνω φράγματα της χρονικής πολυπλοκότητας που φαίνονται στον Πίνακα 4.1 αποδεικνύονται από το Θεώρημα 4.7 παρακάτω. Για την απόδειξή τους σχεδιάζουμε αλγόριθμους που επιτυγχάνουν τη συνάντηση κάτω από δεδομένες αρχικές συνθήκες. Για παράδειγμα ο Αλγόριθμος 9 οδηγεί τους πράκτορες σε συνάντηση όταν το d και ο προσανατολισμός του δακτυλίου είναι γνωστά αλλά το n όχι. Ενώ για τον υπολογισμό των κάτω φραγμάτων στο Θεώρημα 4.6 θεωρήσαμε ότι οι πράκτορες έχουν απεριόριστη μνήμη, οι απαιτήσεις σε μνήμη για τον υπολογισμό των άνω φραγμάτων εξαρτώνται από την περίπτωση. Για παράδειγμα ο Αλγόριθμος 9 απαιτεί $O(\log d)$ μνήμη για κάθε πράκτορα.

Πριν δώσουμε την απόδειξη του Θεωρήματος 4.7 αναφέρουμε τα φράγματα που προκύπτουν από τους αλγόριθμους οι οποίοι παρουσιάζονται στην απόδειξη. Ειδικότερα, με την εξαίρεση της περίπτωσης που το d είναι γνωστό και ο δακτύλιος δεν είναι προσανατολισμένος, τα φράγματα του Θεωρήματος 4.6 είναι σφιχτά. Οι Αλγόριθμοι 9, 10, και 11 δείχνουν ότι, με μια εξαίρεση, αν οι πράκτορες γνωρίζουν είτε το d είτε το n, το πάνω φράγμα της χρονικής πολυπλοκότητας είναι $\frac{3n}{4}$. Η εξαίρεση εμφανίζεται όταν το d είναι γνωστό αλλά ο δακτύλιος δεν είναι προσανατολισμένος. Σε αυτήν την περίπτωση το πάνω φράγμα προκύπτει από τον Αλγόριθμο 10 και είναι $\frac{5n}{6}$.

Θεώρημα 4.7 ([Kranakis et al., 2003]) Θεωρήστε δύο πανομοιότυπους πράκτορες σε απόσταση $d < \frac{n}{2}$ μεταξύ τους σε έναν ανώνυμο συγχρονισμένο δακτύλιο από n κόμβους. Οι πράκτορες γνωρίζουν ότι βρίσκονται σε τοπολογία δακτυλίου καθώς και ότι ισχύει $d < \frac{n}{2}$. Αν οι πράκτορες εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο για το πρόβλημα της συνάντησης τα πάνω φράγματα της χρονικής πολυπλοκότητας είναι αυτά που φαίνονται στον Πίνακα 4.1.

Απόδειξη. Η απόδειξη των άνω φραγμάτων βασίζεται σε τέσσερις απλούς αλγόριθμους που παρουσιάζονται στη συνέχεια. Αυτοί οι αλγόριθμοι αφορούν τέσσερις περιπτώσεις σε σχέση με τις τιμές των παραμέτρων d, n καθώς και με την αρχική γνώση που έχουν οι πράκτορες για αυτές. Στις παρακάτω περιπτώσεις οι πράκτορες γνωρίζουν ότι $d < \frac{n}{2}$ ακόμη και αν οι τιμές των d και n είναι άγνωστες.

Ο Αλγόριθμος 9 λειτουργεί όταν οι πράκτορες έχουν $O(\log d)$ μνήμη, γνωρίζουν το d και συμφωνούν για τον προσανατολισμό του δακτυλίου.

Με το d και τον προσανατολισμό του δακτυλίου γνωστά, ο ένας από τους πράκτορες θα συναντήσει ένα σημάδι πριν κινηθεί d βήματα και θα συνεχίσει να κινείται στην ίδια κατεύθυνση. Ο άλλος πράκτορας θα κινηθεί για d βήματα και μετά θα κινηθεί στην αντίθετη κατεύθυνση. Οι πράκτορες θα συναντηθούν μετά από χρόνο $\frac{3d}{2}$. Άρα για $d < \frac{n}{2}$, η χειρότερη περίπτωση για τη χρονική πολυπλοκότητα είναι $\frac{3n}{4}$.

Αλγόριθμος 9 (Συνάντηση δύο πρακτόρων σε προσανατολισμένο δακτύλιο όταν είναι γνωστή η ελάχιστη μεταξύ τους απόσταση d)

1: Άφησε το σημάδι.
2: Κινήσου στο δακτύλιο σε αριστερόστροφη κατεύθυνση.
3: **Αν** συναντήσεις σημάδι μέσα σε d βήματα, συνέχισε να ταξιδεύεις στην ίδια κατεύθυνση.
4: Αλλιώς, **αν** δεν συναντήσεις σημάδι μέσα σε d βήματα, κινήσου στην αντίθετη κατεύθυνση.

Ο Αλγόριθμος 10 λειτουργεί όταν οι πράκτορες έχουν $O(\log d)$ μνήμη, το d είναι γνωστό αλλά ο δακτύλιος δεν είναι προσανατολισμένος.

Αλγόριθμος 10 (Συνάντηση δύο πρακτόρων σε μη-προσανατολισμένο δακτύλιο όταν είναι γνωστή η ελάχιστη μεταξύ τους απόσταση d)

1: Άφησε το σημάδι.
2: Διάλεξε μια κατεύθυνση και κινήσου.
3: **Αν** συναντήσεις σημάδι μέσα σε d βήματα, συνέχισε να ταξιδεύεις στην ίδια κατεύθυνση.
4: Αλλιώς, **αν** δεν συναντήσεις σημάδι μέσα σε d βήματα, κινήσου στην αντίθετη κατεύθυνση.

Η χειρότερη περίπτωση εμφανίζεται όταν οι πράκτορες διαλέγουν αρχικά διαφορετικές κατευθύνσεις και κινούνται για d βήματα χωρίς να συναντήσουν κάποιο σημάδι. Αν ισχύει $d < \frac{n}{3}$, οι πράκτορες αλλάζουν κατεύθυνση μετά από d βήματα και συνεχίζουν να κινούνται. Μετά από συνολικά $2d$ βήματα, κάθε πράκτορας βρίσκεται πίσω στην αρχική του θέση και έτσι η συνάντηση θα επιτευχθεί μετά από ακόμη $\frac{d}{2}$ βήματα και συνεπώς ο συνολικός αριθμός βημάτων είναι $\frac{5d}{2}$. Αφού $d < \frac{n}{3}$, η συνάντηση επιτυγχάνεται μετά από το πολύ $\frac{5n}{6}$ βήματα. Αν $\frac{n}{3} \leq d < \frac{n}{2}$, οι πράκτορες θα συναντηθούν μετά από το πολύ $\frac{d}{2}$ ή $\frac{n-d}{2}$ βήματα το οποίο οδηγεί σε πολυπλοκότητα το πολύ $\frac{n}{3}$ βημάτων. Έτσι, τελικά για $d < \frac{n}{2}$, η χειρότερη περίπτωση είναι $\frac{5n}{6}$ βήματα.

Ο Αλγόριθμος 11 λειτουργεί όταν οι πράκτορες έχουν $O(\log n)$ μνήμη, και μόνο το n είναι γνωστό.

Αλγόριθμος 11 (Συνάντηση δύο πρακτόρων σε μη-προσανατολισμένο δακτύλιο όταν είναι γνωστό το πλήθος n των κόμβων του δακτυλίου)

1: Άφησε το σημάδι.
2: Διάλεξε μια κατεύθυνση και κινήσου.
3: **Αν** συναντήσεις σημάδι μέσα σε $\frac{n}{2}$ βήματα, συνέχισε να ταξιδεύεις στην ίδια κατεύθυνση.
4: Αλλιώς, **αν** δεν συναντήσεις σημάδι μέσα σε $\frac{n}{2}$ βήματα, κινήσου στην αντίθετη κατεύθυνση.

Στην χειρότερη περίπτωση οι δύο πράκτορες κινούνται προς την ίδια κατεύθυνση. Ο ένας πράκτορας κινείται για d βήματα, συναντά ένα σημάδι, και συνεχίζει να κινείται προς την ίδια κατεύθυνση. Ο άλλος πράκτορας κινείται για $n/2$ βήματα, δεν συναντά σημάδι, και συνεπώς αλλάζει

κατεύθυνση και συνεχίζει να κινείται. Έτσι η συνάντηση θα συμβεί μετά από $\frac{n+d}{2}$ βήματα το οποίο με $d < \frac{n}{2}$, μας δίνει το πολύ $\frac{3n}{4}$ βήματα.

Ο Αλγόριθμος 12 λειτουργεί όταν οι πράκτορες έχουν $O(\log n)$ μνήμη, ενώ τα d και n είναι άγνωστα.

Αλγόριθμος 12 (Συνάντηση δύο πρακτόρων σε μη-προσανατολισμένο δακτύλιο

1: Άφησε το σημάδι.
2: Διάλεξε μια κατεύθυνση και κινήσου.
3: Μέτρησε τον αριθμό δ_1 των βημάτων μέχρι το πρώτο σημάδι και συνέχισε να κινείσαι.
4: Μέτρησε τον αριθμό δ_2 των βημάτων μέχρι να συναντήσεις το δεύτερο σημάδι.
/* Ο πράκτορας έχει επιστρέψει στον κόμβο από τον οποίο ξεκίνησε.*/
5: **Αν** $\delta_1 < \delta_2$, συνέχισε να κινείσαι προς την ίδια κατεύθυνση.
6: Αλλιώς κινήσου στην αντίθετη κατεύθυνση.

Οι δύο πράκτορες κινούνται στην ίδια κατεύθυνση και έτσι η συνάντηση θα προκύψει σε $n + \frac{d}{2}$ βήματα το οποίο με $d < \frac{n}{2}$, μας δίνει το πολύ $\frac{5n}{4}$ βήματα. ■

Οι παραπάνω αλγόριθμοι θεωρούν ότι οι πράκτορες γνωρίζουν ότι $d < \frac{n}{2}$. Παρατηρήστε ότι αν οποιοσδήποτε από αυτούς τους αλγόριθμους εκτελεστεί σε δακτύλιο με άρτιο πλήθος κόμβων και $d = \frac{n}{2}$ το αποτέλεσμα θα είναι ότι ο αλγόριθμος θα τρέχει για πάντα. Το πρόβλημα αυτό μπορεί εύκολα να αντιμετωπιστεί. Έστω πως σε όλες τις περιπτώσεις οι πράκτορες έχουν $O(\log n)$ μνήμη. Οι πράκτορες μπορούν τώρα να μετρήσουν τον αριθμό των βημάτων που κάνουν αφότου πήραν την απόφαση να διατηρήσουν ή να αλλάξουν κατεύθυνση. Για παράδειγμα στον Αλγόριθμο 10, οι πράκτορες παίρνουν την απόφαση να αλλάξουν ή όχι κατεύθυνση στο βήμα 3 ή 4 του αλγορίθμου. Αν η συνάντηση δεν συμβεί μέσα στα επόμενα $\frac{3d}{2} \leq \frac{3n}{4}$ βήματα, τότε οι πράκτορες καταλαβαίνουν ότι είναι αδύνατο να συναντηθούν και σταματούν. Έτσι λοιπόν αν οι πράκτορες έχουν $O(\log n)$ μνήμη, δεν χρειάζεται να γνωρίζουν αν ισχύει $d < \frac{n}{2}$.

Αυτό μας οδηγεί στην ακόλουθη διάκριση: *συνάντηση χωρίς ανίχνευση* (ή απλά συνάντηση *rendezvous*, $\mathcal{RP}$) και *συνάντηση με ανίχνευση* ($\mathcal{RD}$). Στην πρώτη περίπτωση οι πράκτορες γνωρίζουν ότι το πρόβλημα λύνεται (είτε λόγω της αρχικής τοποθέτησης είτε ανεξάρτητα από αυτήν) και απλά πρέπει να συναντηθούν όσο γίνεται πιο γρήγορα σε έναν κόμβο του δακτυλίου (για παράδειγμα σε ένα δακτύλιο με περιττό πλήθος κόμβων το πρόβλημα λύνεται πάντα). Στη δεύτερη περίπτωση ενδιαφερόμαστε επίσης για το πρόβλημα απόφασης το οποίο απαιτεί επίσης μια λύση του προβλήματος τερματισμού του αλγόριθμου. Δηλαδή χρειαζόμαστε έναν αλγόριθμο ο οποίος ανιχνεύει την επιλυσιμότητα του προβλήματος της συνάντησης για οποιονδήποτε αρχικό σχηματισμό μετά από *πεπερασμένο* αριθμό βημάτων (ο οποίος συνήθως εξαρτάται είτε από την απόσταση των πρακτόρων είτε από το μέγεθος του δακτυλίου). Έτσι αν η συνάντηση είναι εφικτή επιτυγχάνεται και αν όχι οι πράκτορες σταματούν και γνωρίζουν ότι η συνάντηση είναι αδύνατη.

4.2.1.3 Ισοζύγιο μνήμης για συνάντηση με ανίχνευση

Με $O(\log n)$ μνήμη, οι Αλγόριθμοι 9 με 12 μπορούν να ανιχνεύσουν αν $d = \frac{n}{2}$ και να δράσουν αναλόγως, δηλαδή να σταματήσουν αν $d = \frac{n}{2}$ και να οδηγήσουν σε συνάντηση σε αντίθετη περίπτωση. Στον Αλγόριθμο 12, όπου οι πράκτορες δεν γνωρίζουν τα n, d, ή τον προσανατολισμό του δακτυλίου, η χρονική πολυπλοκότητα είναι $\frac{5n}{4}$. Αν η αρχική γνώση των πρακτόρων είναι η ίδια αλλά η μνήμη τους είναι μόνο $O(\log \log n)$, ο αλγόριθμος που ακολουθεί μπορεί να ανιχνεύσει αν $d = \frac{n}{2}$ και να δράσει αναλόγως. Η χρονική πολυπλοκότητα του αλγορίθμου είναι

$$O\left(\frac{n \log n}{\log \log n}\right),$$

το οποίο μας δείχνει ότι η μνήμη των πρακτόρων μπορεί να μειωθεί με κόστος τη χρονική πολυπλοκότητα. Στον παρακάτω αλγόριθμο θεωρούμε ότι οι αριθμοί $p_1, \ldots, p_k$ αναπαριστούν τους k μικρότερους πρώτους αριθμούς έτσι ώστε να ισχύει $\prod_{i=1}^{k} p_i > n$.

Αλγόριθμος 13 (Συνάντηση δύο πρακτόρων με μνήμη $O(\log \log n)$ σε μη-προσανατολισμένο δακτύλιο n κόμβων)

1: Άφησε το σημάδι.
2: Θέσε $m = p_1$.
3: Διάλεξε μια κατεύθυνση και κινήσου.
4: Μέτρησε τον αριθμό δ_1 των βημάτων $\mod m$, μέχρι το πρώτο σημάδι και συνέχισε να κινείσαι.
5: Μέτρησε τον αριθμό δ_2 των βημάτων $\mod m$, μέχρι το δεύτερο σημάδι.
/* Ο πράκτορας έχει επιστρέψει στον κόμβο από τον οποίο ξεκίνησε. */
6: **Αν** $\delta_1 \mod m = \delta_2 \mod m$,
Αν $m = p_k$, σταμάτα.
/* Η συνάντηση είναι αδύνατη. */
Αν $m < p_k$, θέσε $m = p_{i+1}$ και επανέλαβε από το βήμα 4.
7: **Αν** $\delta_1 \mod m < \delta_2 \mod m$, συνέχισε να κινείσαι προς την ίδια κατεύθυνση.
8: **Αλλιώς** άλλαξε κατεύθυνση και συνέχισε να κινείσαι.
/* Αν εκτελεστεί το βήμα 7 ή το 8 η συνάντηση συμβαίνει σε επιπλέον $\frac{d}{2}$ βήματα.*/

Θεώρημα 4.8 ([Kranakis et al., 2003]) Θεωρήστε δύο πανομοιότυπους πράκτορες με μνήμη $O(\log \log n)$ που βρίσκονται σε απόσταση d ο ένας από τον άλλον σε έναν ανώνυμο συγχρονισμένο δακτύλιο από n κόμβους. Οι πράκτορες δεν γνωρίζουν τα n, d, ή τον προσανατολισμό του δακτυλίου καθώς και ούτε αν $d = \frac{n}{2}$. Οι πράκτορες εκτελούν ταυτόχρονα τον ίδιο ντετερμινιστικό αλγόριθμο. Ο Αλγόριθμος 13 ανιχνεύει αν $d = \frac{n}{2}$ και σταματά αν $d = \frac{n}{2}$, αλλιώς οδηγεί τους πράκτορες σε συνάντηση. Η χρονική πολυπλοκότητα του αλγορίθμου είναι $O\left(\frac{n \log n}{\log \log n}\right)$.

Απόδειξη. Από το Κινέζικο Θεώρημα Υπολοίπων προκύπτει πως αν $d \equiv (n-d) \bmod p_i$ για κάθε $p_i, i = 1, \ldots, k$, τότε

$$d \equiv (n-d) \bmod \prod_{i=1}^{k} p_i.$$

Ο αλγόριθμος ελέγχει κάθε p_i για να δει αν $d \equiv (n-d) \bmod p_i$. Αν η ιδιότητα είναι αλήθεια για κάθε p_i, τότε $d = n - d = \frac{n}{2}$ και ο αλγόριθμος σταματά αφού η συνάντηση είναι αδύνατη. Αν

$$d \not\equiv (n-d) \bmod \prod_{i=1}^{k} p_i,$$

για κάποιο p_i, τότε $d < \frac{n}{2}$. Την πρώτη στιγμή που ο αλγόριθμος ανακαλύπτει ένα τέτοιο p_i, ένας από τους πράκτορες αλλάζει κατεύθυνση και η συνάντηση προκύπτει $\frac{d}{2}$ βήματα αργότερα.

Η χειρότερη περίπτωση εμφανίζεται όταν $d = \frac{n}{2}$ αφού όλα τα k των p_i πρέπει να ελεγχθούν. Ο χρόνος εκτέλεσης που προκύπτει είναι $O(kn)$, αλλά πρέπει να βρούμε την τιμή του k. Έστω το μικρότερο k έτσι ώστε $\prod_{i=1}^{k} p_i > n$. Αυτό σημαίνει ότι $\prod_{i=1}^{k-1} p_i \leq n$ και συνεπώς

$$\prod_{i=1}^{k} p_i \leq n * p_k \leq n^2.$$

Έστω $\Pi(m)$ το πλήθος των πρώτων αριθμών που είναι μικρότεροι ή ίσοι του m. Στο άρθρο [Apostol, 1997] βρίσκουμε ότι για κάθε ακέραιο m,

$$\frac{m}{6 \log m} \leq \Pi(m) \leq \frac{8m}{\log m}.$$

Αφού ο αριθμός p_i είναι πρώτος, $\Pi(p_i) = i$ για κάθε i έτσι ώστε

$$\frac{p_i}{6 \log p_i} \leq \Pi(p_i) = i \leq \frac{8p_i}{\log p_i}$$

και συνεπώς $p_i \geq \frac{i \log p_i}{8}$. Παίρνοντας το γινόμενο για όλα τα i και χρησιμοποιώντας την προσέγγιση του Stirling [Cormen et al., 2001] προκύπτει ότι

$$\begin{aligned} n^2 \geq \prod_{i=1}^{k} p_i &\geq \prod_{i=1}^{k} \frac{i \log p_i}{8} \\ &= k! 8^{-k} \prod_{i=1}^{k} \log p_i \\ &\geq k! 8^{-k} \\ &\geq 2^{\Omega(k \log k)}. \end{aligned}$$

Παίρνοντας τους λογαρίθμους στην παραπάνω εξίσωση έχουμε ότι $2\log n \geq k\log k$, έτσι μια νόμιμη τιμή για το k πρέπει να ικανοποιεί την ιδιότητα $k\log k \leq 2\log n$ και άρα παίρνοντας

$$k \in O\left(\frac{\log n}{\log\log n}\right)$$

είναι αρκετό. Αφού η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(kn)$ παίρνουμε το φράγμα $O(\frac{n\log n}{\log\log n})$. ■

4.2.1.4 Όρια στο ισοζύγιο μνήμης

Παρά το γεγονός ότι τα αποτελέσματα και η συζήτηση στην ενότητα 4.2.1.3 μας δείχνουν ότι η μνήμη των πρακτόρων θα μπορούσε να μειωθεί με κόστος την αύξηση της χρονικής πολυπλοκότητας, στο θεώρημα που ακολουθεί αποδεικνύεται ότι η συνάντηση δεν μπορεί να επιτευχθεί όταν οι πράκτορες έχουν σταθερή μνήμη.

Θεώρημα 4.9 ([Kranakis et al., 2003]) Θεωρήστε δύο πανομοιότυπους κινητούς πράκτορες με σταθερή μνήμη οι οποίοι είναι τοποθετημένοι με απόσταση d μεταξύ τους σε έναν ανώνυμο και συγχρονισμένο δακτύλιο από n κόμβους. Οι πράκτορες δεν γνωρίζουν τα n, d, ούτε τον προσανατολισμό του δακτυλίου καθώς και ούτε αν ισχύει $d = \frac{n}{2}$. Οι πράκτορες εκτελούν ταυτόχρονα τον ίδιο ντετερμινιστικό αλγόριθμο. Οι πράκτορες έχουν στη διάθεσή τους από ένα μη-μετακινήσιμο σημάδι το οποίο μπορούν να τοποθετήσουν σε κάποιο κόμβο του δακτυλίου. Δεν υπάρχει αλγόριθμος που μπορεί να ανιχνεύσει αν ισχύει $d = \frac{n}{2}$ και αν ναι να σταματήσει ενώ αν όχι να οδηγήσει τους πράκτορες σε συνάντηση.

Απόδειξη. Έστω ότι υπάρχει ένας αλγόριθμος με βάση τον οποίον οι πράκτορες μπορούν να ανιχνεύσουν αν βρίσκονται σε απόσταση $d = \frac{n}{2}$ και αν όχι να συναντηθούν. Θεωρήστε έναν δακτύλιο μιας κατεύθυνσης που αποτελείται από n κόμβους και έστω η απόσταση $d = \frac{n}{2}$. Λόγω της σταθερής μνήμης κάθε πράκτορας μπορεί να αποθηκεύσει το πολύ έναν σταθερό αριθμό από k bits σε μια χρονική στιγμή. Αυτά τα k bits αναπαριστούν την κατάσταση του πράκτορα εκείνη τη στιγμή και η επόμενη πράξη του πράκτορα εξαρτάται από αυτήν την κατάσταση. Ας συμβολίσουμε με MA_1 τον έναν από τους δύο πράκτορες. Κατασκευάζουμε την εξής ακολουθία από bits για κάθε κόμβο του δακτυλίου. Τα πρώτα k bits στην ακολουθία είναι εκείνα που είναι αποθηκευμένα στη μνήμη του MA_1 όταν εκείνος επισκέπτεται για πρώτη φορά τον δεδομένο κόμβο του δακτυλίου. Σε κάθε επόμενη επίσκεψη του πράκτορα στον κόμβο, τα k bits που είναι αποθηκευμένα στη μνήμη του MA_1 τη στιγμή της επίσκεψης εισάγονται στο τέλος της ακολουθίας. Έστω R οι φορές που έχει ταξιδέψει ο πράκτορας MA_1 τον δακτύλιο. Η ακολουθία που προκύπτει σε κάθε κόμβο θα περιέχει τουλάχιστον Rk bits. Αυτά τα Rk bits κωδικοποιούν το ιστορικό, δηλαδή την ακολουθία των καταστάσεων του που είχε ο MA_1 κάθε φορά που επισκέφτηκε τον δεδομένο κόμβο. Σε έναν δακτύλιο με n κόμβους όπου το n είναι αρκετά μεγάλο, δηλαδή $n >> Rk$, δύο τέτοιες ακολουθίες θα είναι ίδιες, δηλαδή δύο διαφορετικοί κόμβοι θα έχουν την ίδια ακολουθία από Rk bits. Αν το n είναι αρκετά μεγάλο ένα από τα μονοπάτια μεταξύ αυτών των δύο κόμβων δεν θα περιέχει κόμβο με σημάδι.

Έστω $node_a$ και $node_b$ αυτοί οι δύο κόμβοι όπου ο $node_a$ προηγείται του $node_b$ στον δακτύλιο μιας κατεύθυνσης. Από την κατασκευή των ακολουθιών προκύπτει πως η κατάσταση στην οποία βρίσκεται ο πράκτορας MA_1 όταν επισκέπτεται τον κόμβο $node_a$ είναι η ίδια με εκείνη που βρίσκεται όταν επισκέπτεται τον κόμβο $node_b$ *για κάθε* μία από τις R επισκέψεις. Μπορούμε πάντα να αφαιρέσουμε κόμβους που βρίσκονται ανάμεσα στους $node_a$ και $node_b$ έτσι ώστε η συμπεριφορά του MA_1 σε όλους τους κόμβους να παραμένει η ίδια. Δηλαδή οι δύο πράκτορες έχουν την ίδια συμπεριφορά σε δακτύλιο για τον οποίον $d = \frac{n}{2}$ και σε εκείνον για τον οποίον $d < \frac{n}{2}$. Αυτό αντίκειται στην υπόθεση ότι ο αλγόριθμος σταματά στην πρώτη περίπτωση και επιτυγχάνει συνάντηση στη δεύτερη. Άρα δεν μπορεί να υπάρχει αλγόριθμος για πράκτορες με σταθερή μνήμη ο οποίος ανιχνεύει αν $d = \frac{n}{2}$ και $d < \frac{n}{2}$, και σταματά ή οδηγεί τους πράκτορες σε συνάντηση αντίστοιχα. ■

4.2.2 Μετακινήσιμα σημάδια

Το Θεώρημα 4.9 μας δείχνει ότι αν οι πράκτορες έχουν μόνο σταθερή μνήμη και δεν γνωρίζουν τα n, d, ή τον προσανατολισμό του δακτυλίου, τότε το πρόβλημα της συνάντησης δεν μπορεί να λυθεί στο δεδομένο μοντέλο. Ακόμη και αν οι πράκτορες γνωρίζουν ότι $d < \frac{n}{2}$, ο Αλγόριθμος 12 όπως φαίνεται στο Θεώρημα 4.7 απαιτεί από τους πράκτορες να μπορούν να μετρήσουν μέχρι το $\frac{n}{2}$ και συνεπώς χρειάζονται $O(\log n)$ μνήμη. Προσθέτουμε λοιπόν στο μοντέλο την εξής δυνατότητα. Όταν ένας πράκτορας συναντήσει κάποιο σημάδι μπορεί να το μετακινήσει σε κάποιον άλλο κόμβο. Αποδεικνύουμε στο Θεώρημα 4.10 παρακάτω, ότι ο Αλγόριθμος 14 ο οποίος χρησιμοποιεί μετακινήσιμα σημάδια μπορεί να λύσει το πρόβλημα της συνάντησης χρησιμοποιώντας σταθερή μνήμη.

Ο παρακάτω αλγόριθμος λειτουργεί με σταθερή μνήμη χωρίς να είναι γνωστά τα n, d, ή ο προσανατολισμός του δακτυλίου.

Αλγόριθμος 14 (Συνάντηση δύο πρακτόρων με σταθερή μνήμη σε μη-προσανατολισμένο δακτύλιο)

1: Άφησε το σημάδι και διάλεξε μια κατεύθυνση.
2: Κινήσου μέχρι να βρεις ένα σημάδι.
3: Μόλις βρεις ένα σημάδι άλλαξε κατεύθυνση.
4: Μετακίνησε το σημάδι στον διπλανό κόμβο προς τη νέα κατεύθυνση.
5: **Αν** υπάρχουν δύο σημάδια στον κόμβο σταμάτα.
6: **Αλλιώς** επανέλαβε από το βήμα 2.

Θεώρημα 4.10 ([Kranakis et al., 2003]) Θεωρήστε δύο πανομοιότυπους πράκτορες με σταθερή μνήμη σε απόσταση $d < \frac{n}{2}$ μεταξύ τους σε έναν ανώνυμο συγχρονισμένο δακτύλιο από n κόμβους. Οι πράκτορες δεν γνωρίζουν τα n, d, ή τον προσανατολισμό του δακτυλίου αλλά έχουν στη διάθεσή τους από ένα μετακινήσιμο σημάδι. Τότε ο Αλγόριθμος 14 οδηγεί τους πράκτορες σε συνάντηση μετά από το πολύ χρόνο $T \in O\left(d^2\left(\frac{d^2}{n-2d} + n\right)\right)$. Συνεπώς αν το d τείνει στο $\frac{n}{2}$, το T τείνει στο $O(n^4)$.

Απόδειξη. Στη χειρότερη περίπτωση οι πράκτορες θα επιλέξουν την ίδια κατεύθυνση στο πρώτο βήμα του αλγορίθμου. Παρατηρήστε ότι επειδή $d < n - d$, η απόσταση μεταξύ των δύο σημαδιών δεν θα αυξηθεί κατά τη διάρκεια του αλγορίθμου. Θα μελετήσουμε τώρα πότε ακριβώς μειώνεται η απόσταση μεταξύ των σημαδιών.

Έστω ότι ο πράκτορας MA_1 έχει κινηθεί μπρος και πίσω ανάμεσα στα σημάδια i φορές και έχει επιστρέψει στον κόμβο από τον οποίο ξεκίνησε. Έστω ότι ο δεύτερος πράκτορας MA_2 έχει κινηθεί μπρος και πίσω ανάμεσα στα σημάδια j φορές και έχει μετακινηθεί r επιπλέον κόμβους μακρυά από τον κόμβο από τον οποίο ξεκίνησε. Η απόσταση μεταξύ των δύο σημαδιών θα μειωθεί κατά ένα αν ο πράκτορας MA_1 προλάβει να μετακινήσει τα δύο σημάδια πριν ο άλλος πράκτορας MA_2 συναντήσει το επόμενο σημάδι. Αυτό απαιτεί να ισχύει ότι

$$\begin{aligned} id &= j(n-d) + r \\ r &\geq 0 \\ n - d - r &> d \end{aligned} \tag{4.1}$$

για κάποια $0 \leq j < i$ και r. Αυτές οι απαιτήσεις σημαίνουν ότι αν ισχύει

$$\frac{j}{i+j} \leq \frac{d}{n} < \frac{j+1}{i+j+2} \tag{4.2}$$

τότε η απόσταση μεταξύ των δύο σημαδιών θα μειωθεί σε $d-1$ σε το πολύ

$$id + n - d = (i-1)d + n \tag{4.3}$$

βήματα. Θέτοντας $j = i - 1$ στην Εξίσωση 4.2 παίρνουμε την έκφραση

$$\frac{i-1}{2i-1} \leq \frac{d}{n} < \frac{i}{2i+1} \tag{4.4}$$

Το αριστερό μέρος της Εξίσωσης 4.4 είναι ισοδύναμο με $i - 1 \leq \frac{d}{n-2d}$. Επίσης όταν ισχύει

$$d - \frac{i-1}{2i-1}n < k \leq d - \frac{i-2}{2i-3}n \tag{4.5}$$

η απόσταση μεταξύ των δύο σημαδιών θα μειωθεί σε $d - k$ και ο λόγος $\frac{d-k}{n}$ θα ικανοποιεί την

$$\frac{i-2}{2i-3} \leq \frac{d-k}{n} < \frac{i-1}{2i-1} \tag{4.6}$$

Προκύπτει ότι η απόσταση ανάμεσα στα σημάδια θα μειωθεί σε το πολύ $d - k$ μέσα σε το πολύ

$$k((i-1)d + n) \leq k\left(\frac{d^2}{n-2d} + n\right) \leq \left(d - \frac{i-2}{2i-3}n\right)\left(\frac{d^2}{n-2d} + n\right) \tag{4.7}$$

βήματα, έτσι η συνάντηση θα συμβεί σε χρόνο

$$O\left(d^2\left(\frac{d^2}{n-2d} + n\right)\right).$$

■

Με μια καλύτερη ανάλυση ο παραπάνω αλγόριθμος αποδεικνύεται ότι οδηγεί σε συνάντηση τους πράκτορες (όταν η αρχική τους απόσταση είναι $d < n/2$) το πολύ σε χρόνο $O(n^2 \log n)$ και αυτό το φράγμα είναι σφιχτό.

Στη συνέχεια αναλύουμε το πρόβλημα της συνάντησης με ανίχνευση ($\mathcal{RD}$) χρησιμοποιώντας μετακινήσιμα σημάδια και μελετούμε το ισοζύγιο μεταξύ του πλήθους των σημαδιών που έχει στη διάθεσή του ο κάθε πράκτορας και του χρόνου που χρειάζεται για να λυθεί το πρόβλημα σε έναν δακτύλιο n κόμβων. Ειδικότερα θα μελετήσουμε το πρόβλημα για δύο πράκτορες με σταθερή μνήμη και ένα ή περισσότερα μετακινήσιμα σημάδια. Τα κυριότερα αποτελέσματα για ένα και δύο σημάδια φαίνονται στον Πίνακα 4.2. Στην πρώτη στήλη φαίνεται το πλήθος των σημαδιών που είναι διαθέσιμα σε κάθε πράκτορα. Η δεύτερη στήλη έχει τον αριθμό των κατευθύνσεων του δακτυλίου (1 σημαίνει μιας κατεύθυνσης και 2 δύο κατευθύνσεων). Στην τρίτη και τέταρτη στήλη φαίνεται ο χρόνος που απαιτείται για τη λύση του προβλήματος. Το σύμβολο ∞ στον πίνακα σημαίνει ότι η λύση του προβλήματος είναι αδύνατη. Η μνήμη που απαιτείται από όλους τους αλγόριθμους είναι

Πίνακας 4.2: Ο χρόνος που απαιτείται για τη συνάντηση δύο πρακτόρων με σταθερή μνήμη και μετακινήσιμα σημάδια σε έναν συγχρονισμένο δακτύλιο n κόμβων.

Συνθήκες		Χρόνος που απαιτείται για τη συνάντηση	
Σημάδια	Κατευθύνσεις	$\mathcal{RD}$	$\mathcal{RP}$
1	1	∞	∞
1	2	∞	$\Theta(n^2)$
2	1	$\Theta(n^2)$	$\Theta(n^2)$
2	2	$\Theta(n^2)$	$\Theta(n^2)$

$O(1)$. Στη συνέχεια παρουσιάζουμε τους αλγόριθμους που αποδεικνύουν τα πάνω φράγματα ενώ για τα κάτω φράγματα των οποίων οι αποδείξεις είναι πιο δύσκολες ο αναγνώστης που ενδιαφέρεται παραπέμπεται στο άρθρο [Czyzowicz et al., 2008].

Αρχικά ας θεωρήσουμε την περίπτωση όπου κάθε πράκτορας έχει ένα μόνο σημάδι και ο δακτύλιος έχει δύο κατευθύνσεις. Σε αυτήν την περίπτωση το πρόβλημα της συνάντησης χωρίς ανίχνευση λύνεται όπως αποδεικνύεται στο θεώρημα που ακολουθεί.

Θεώρημα 4.11 ([Czyzowicz et al., 2008]) Στους δακτύλιους με δύο κατευθύνσεις το πρόβλημα της συνάντησης ($\mathcal{RP}$) επιλύεται σε $O(n^2)$ χρόνο για δύο πράκτορες που έχουν σταθερή μνήμη και ένα μετακινήσιμο σημάδι ο καθένας.

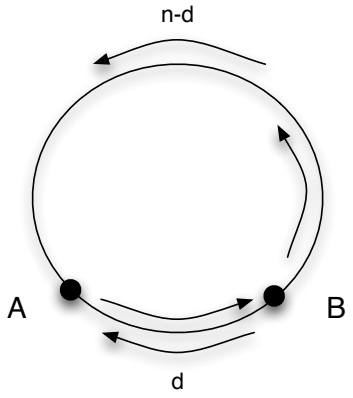
Απόδειξη. Μετατρέπουμε τον προηγούμενο αλγόριθμο ως εξής: προσθέτουμε ένα μετρητή ελέγχου ο οποίος μετρά modulo 3 για να ανιχνεύσει αλλαγές στη θέση του ενός σημαδιού σε σχέση με το άλλο. Ο νέος Αλγόριθμος 15 φαίνεται παρακάτω.

Παρατηρήστε ότι αν οι πράκτορες ξενικήσουν να κινούνται ο ένας προς τον άλλον θα συναντηθούν σε χρόνο $O(n)$. Ας υποθέσουμε τώρα ότι κινούνται στην ίδια κατεύθυνση. Έστω A και B

Αλγόριθμος 15 (Συνάντηση δύο πρακτόρων με σταθερή μνήμη σε μη-προσανατολισμένο δακτύλιο μετά από $O(n^2)$ χρόνο)

1: Άφησε το σημάδι στον κόμβο εκκίνησης.
2: Κινήσου δεξιόστροφα μέχρι να βρεις ένα σημάδι ή να συναντήσεις τον άλλο πράκτορα μετρώντας την απόσταση x που έχεις ταξιδέψει modulo τρία.
3: **Αν** δεν συνάντησες τον άλλο πράκτορα
4: **Επανέλαβε**
5: Άλλαξε κατεύθυνση, και μετακίνησε το σημάδι στον διπλανό κόμβο στη νέα κατεύθυνση.
6: Συνέχισε να κινείσαι στη νέα κατεύθυνση μέχρι να βρεις ένα σημάδι μετρώντας την απόσταση y που έχεις ταξιδέψει modulo 3.
7: **Μέχρι** $y \equiv (x-1) \bmod 3$ ή να συναντήσεις τον άλλο πράκτορα
8: **Αν** δεν συνάντησες τον άλλον πράκτορα.
9: Σταμάτα και περίμενε για τον άλλο πράκτορα.

οι πράκτορες και d και $n-d$ οι μεταξύ τους αποστάσεις, (βλέπε Σχήμα 4.3). Έστω $t_1, t_2, t_3, \ldots$ οι



Σχήμα 4.3: Υπολογίζοντας το χρόνο συνάντησης δύο πρακτόρων που βρίσκονται σε απόσταση d ο ένας από τον άλλον. Οι πράκτορες ξεκινούν να κινούνται αριστερόστροφα. Ενώ ο πράκτορας A έχει συναντήσει το σημάδι του B και επιστρέφει προς τον κόμβο από τον οποίο ξεκίνησε, ο πράκτορας B δεν έχει ακόμη συναντήσει το σημάδι του A.

στιγμές κατά τις οποίες ο πράκτορας A συναντά ένα σημάδι (δηλαδή t_i είναι η στιγμή που ξεκινά η i-στή επανάληψη του βρόχου). Έστω $t'_1, t'_2, t'_3, \ldots$ οι αντίστοιχες στιγμές για τον πράκτορα B. Έστω χωρίς βλάβη της γενικότητας ότι $d < n-d$ και $\delta = (n-d)-d$. Ισχύει ότι $d = t_1 \equiv x_A \bmod 3$ και $n-d = t'_1 \equiv x_B \bmod 3$. Παρατηρούμε επίσης ότι όσο ισχύει $t_{i+1} < t'_i$, ο πράκτορας B θα

μετακινήσει ένα σημάδι πριν ο πράκτορας A το συναντήσει, και ο A καταλαβαίνει ότι η απόσταση μεταξύ των σημαδιών δεν έχει αλλάξει. Όσο ισχύει $t_1 \neq t'_1$, είναι εύκολο να αποδειχθεί επαγωγικά ότι σε κάθε επανάληψη η διαφορά χρόνου $t'_i - t_i$ μεταξύ των πρακτόρων αυξάνει κατά δ, δηλαδή $t'_i - t_i = i\delta$. Αυτό σημαίνει ότι έπειτα από $\lceil t_1/\delta \rceil$ επαναλήψεις ισχύει $t_{i+1} \geq t'_i$. Άν $t_{i+1} = t'_i$, τότε οι πράκτορες συναντιούνται στον κόμβο με το σημάδι. Αλλιώς ο πράκτορας A συναντά το σημάδι πριν ο B το μετακινήσει, δηλαδή ο A ταξίδεψε μια απόσταση $t_1 - 1 \equiv x_A - 1 \bmod 3$. Τη στιγμή εκείνη ο A σταματά και περιμένει τον B. Εφόσον ο B μέχρι τώρα έχει μετρήσει μόνο ίσες αποστάσεις θα συνεχίσει να κινείται και τελικά θα φτάσει στον κόμβο που περιμένει ο A.

Αφού υπάρχουν t_1/δ επαναλήψεις από t_1 βήματα η κάθε μία και ένας επιπλέον χρόνος το πολύ t'_1 που χρειάζεται ο πράκτορας B για να φτάσει στο σημείο συνάντησης, η συνάντηση θα συμβεί σε χρόνο $O((t_1/\delta)t_1 + t'_1)$. Αφού ισχύει $\delta \geq 1$, $t_1 < n/2$ και $t'_1 = t_1 + \delta < n$, ο χρόνος που προκύπτει είναι $O(n^2)$. ■

Ο προηγούμενος αλγόριθμος λύνει το πρόβλημα $\mathcal{RP}$ αλλά όχι και το $\mathcal{RD}$, αφού θα εκτελείται για πάντα αν οι πράκτορες βρίσκονταν αρχικά σε συμμετρικές θέσεις.

Στη συνέχεια αποδεικνύουμε ότι το πρόβλημα της συνάντησης με ανίχνευση μπορεί να λυθεί (ακόμη και σε δακτύλιους με μία κατεύθυνση) αν εξοπλίσουμε κάθε πράκτορα με δύο μετακινήσιμα σημάδια.

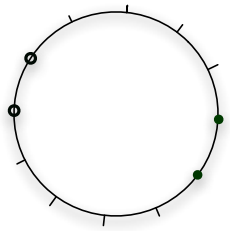
Θεώρημα 4.12 ([Czyzowicz et al., 2008]) Το πρόβλημα της συνάντησης με ανίχνευση ($\mathcal{RD}$) είναι επιλύσιμο σε δακτύλιους με μία κατεύθυνση από δύο πράκτορες με σταθερή μνήμη οι οποίοι έχουν από δύο πανομοιότυπα μετακινήσιμα σημάδια σε χρόνο $O(n^2)$.

Απόδειξη. Παρουσιάζουμε τον Αλγόριθμο 16 ο οποίος λύνει το πρόβλημα της συνάντησης με ανίχνευση χρησιμοποιώντας δύο μετακινήσιμα σημάδια για κάθε πράκτορα. Κάθε πράκτορας αφήνει

Αλγόριθμος 16 (Συνάντηση με ανίχνευση δύο πρακτόρων με σταθερή μνήμη σε μη-προσανατολισμένο δακτύλιο)

1: Άφησε το ένα σημάδι στην αφετηρία και το άλλο σημάδι στον γειτονικό κόμβο προς τα δεξιά.
2: **Επανέλαβε**
3: Κινήσου δεξιόστροφα και μετακίνησε κάθε δεύτερο σημάδι που συναντάς μια θέση δεξιά.
4: **Μέχρι** να συναντήσεις δύο σημάδια στον ίδιο κόμβο.
5: Συνέχισε μέχρι να συναντήσεις κάποιον κόμβο με σημάδι.
6: **Αν** στον κόμβο υπάρχουν δύο σημάδια τότε η συνάντηση είναι αδύνατη
7: **Αλλιώς** περίμενε.

ένα σημάδι στον κόμβο από τον οποίο ξεκινά και το άλλο σημάδι στον γειτονικό κόμβο προς τα δεξιά (βλ. Σχήμα 4.4). Έπειτα ταξιδεύει δεξιόστροφα και μετακινεί κάθε δεύτερο σημάδι που συναντά κατά έναν κόμβο προς τα δεξιά (παρατηρήστε ότι αυτή η διαδικασία θα διατηρήσει τα σημάδια που έχουν τοποθετηθεί στους κόμβους-αφετηρίες). Η διαδικασία επαναλαμβάνεται μέχρι που κάποιος πράκτορας συναντά δύο σημάδια στον ίδιο κόμβο. Όταν συμβεί αυτό, ο πράκτορας κάνει άλλον ένα



Σχήμα 4.4: Δύο πράκτορες σε έναν δακτύλιο λύνουν το πρόβλημα της συνάντησης με ανίχνευση έχοντας σταθερή μνήμη και δύο μετεκινήσιμα σημάδια ο καθένας.

γύρο στο δακτύλιο για να διαπιστώσει αν τα υπόλοιπα δύο σημάδια βρίσκονται και αυτά στον ίδιο κόμβο. Αν ναι τότε οι πράκτορες ξεκίνησαν σε απόσταση $n/2$ και η συνάντηση δεν είναι εφικτή. Αν τα άλλα σημάδια δεν βρίσκονται στον ίδιο κόμβο τότε ο πράκτορας περιμένει στον κόμβο με το ένα σημάδι και ο άλλος πράκτορας θα έρθει να τον συναντήσει.

Ας διαιρέσουμε τον υπολογισμό σε γύρους από n βήματα. Κατά τη διάρκεια ενός γύρου κάθε πράκτορας ολοκληρώνει έναν κύκλο στο δακτύλιο και τα δεύτερα σημάδια έχουν μετακινηθεί κατά δύο κόμβους. Καθώς η αρχική απόσταση του δεύτερου σημαδιού από το πρώτο σημάδι που είχε αφήσει ο άλλος πράκτορας είναι το πολύ $d - 1 \leq n/2 - 1$, ο χρόνος εκτέλεσης του αλγορίθμου είναι το πολύ n φορές τον αριθμό των γύρων συν $n/2$ βήματα για τον τελικό έλεγχο, δηλαδή $n((n/2 - 1)/2) + n/2 = O(n^2)$. ■

Παρατηρήστε ότι ο παραπάνω αλγόριθμος θα λειτουργήσει σε δακτύλιους μονής ή διπλής κατεύθυνσης δίνοντας τον ίδιο χρόνο στη χειρότερη περίπτωση. Σε ένα δακτύλιο διπλής κατεύθυνσης, αν οι πράκτορες συμφωνούν στις κετευθύνσεις του δακτυλίου ο αλγόριθμος θα εκτελεστεί όπως και στο δακτύλιο μονής κατεύθυνσης, αλλιώς οι πράκτορες θα κινηθούν ο ένας προς τον άλλον και θα συναντηθούν σε χρόνο $O(n)$.

Το επόμενο θεώρημα μας παρέχει ένα αντιστάθμισμα μεταξύ του πλήθους t των σημαδιών που χρησιμοποιούνται και του χρόνου που απαιτείται για την επίλυση του $\mathcal{RD}$. Θεωρούμε ότι ο κάθε πράκτορας έχει τουλάχιστον τρία μετακινήσιμα σημάδια, δηλαδή $t \geq 3$.

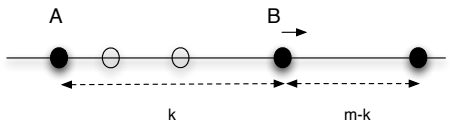
Θεώρημα 4.13 ([Czyzowicz et al., 2008]) Θεωρήστε έναν συγχρονισμένο δακτύλιο διπλής κατεύθυνσης με n κόμβους. Το πρόβλημα της συνάντησης με ανίχνευση ($\mathcal{RD}$) είναι επιλύσιμο από δύο πράκτορες που έχουν $t \geq 3$ μετακινήσιμα σημάδια και $O(\log t)$ bits μνήμη ο καθένας σε χρόνο $O(mn)$, όπου m είναι ο μικρότερος ακέραιος για τον οποίο ισχύει $\binom{m-1}{t-2} \geq n - 1$.

Απόδειξη. Έστω ότι ο κάθε πράκτορας έχει t σημάδια στην κατοχή του. Ένα βασικό συστατικό της απόδειξης είναι η υλοποίηση ενός μετρητή C_t, ο οποίος μπορεί να μετρά μέχρι το n. Δηλαδή μετρά το πλήθος των κόμβων m μεταξύ δύο σημαδιών. Οι τιμές που παίρνει αυτός ο μετρητής κωδικοποιούνται μέσα σε αυτήν την ακολουθία κόμβων. Το ένα από τα σημάδια που διαχωρίζει την

ακολουθία βρίσκεται στον κόμβο αφετηρία του πράκτορα και το άλλο είναι το τελευταίο σημάδι που άφησε ο πράκτορας σε απόσταση m από την αφετηρία του.

Η βασική ιδέα για τη λύση του προβλήματος $\mathcal{RD}$ είναι η εξής: Ο πράκτορας κινείται από την αφετηρία του στην αφετηρία του άλλου πράκτορα ενώ αυξάνει τον μετρητή κατά ένα σε κάθε γύρο. Όταν ο μετρητής φτάσει τη μέγιστη τιμή, ο πράκτορας συνεχίζει να κινείται από την αφετηρία του στην αφετηρία του άλλου πράκτορα αλλά τώρα μειώνει την τιμή του μετρητή κατά ένα σε κάθε γύρο. Παρατηρήστε ότι ο μετρητής μπορεί να μειωθεί ή να αυξηθεί σε χρόνο $O(m)$ σε κάθε γύρο. Όταν ο μετρητής πάρει την τιμή 0 και πριν ο πράκτορας συναντήσει την αφετηρία του, κινείται στην άλλη αφετηρία και αν ο μετρητής του άλλου πράκτορα έχει επίσης την τιμή 0 τότε η αρχική τοποθέτηση των πρακτόρων ήταν συμμετρική και η συνάντηση είναι αδύνατη. Αλλιώς ο πράκτορας περιμένει εκεί για να συναντηθεί με τον άλλο πράκτορα. Έτσι λοιπόν ο αλγόριθμος που μόλις περιγράψαμε ανιχνεύει αν η συνάντηση είναι δυνατή και αν ναι την πραγματοποιεί. Ο χρόνος εκτέλεσης είναι $O(mn)$ αφού κάθε γύρος διαρκεί n βήματα.

Ο χρόνος εκτέλεσης εξαρτάται από το πόσο "συμπαγής" είναι ο μετρητής C_t αφού το κόστος της κίνησης είναι ανάλογο της τιμής του m, η οποία είναι η απόσταση μεταξύ των δύο σημαδιών που διαχωρίζουν τον μετρητή C_t. Έτσι λοιπόν για ένα δεδομένο αριθμό t από σημάδια απομένει να υπολογίσουμε το m έτσι ώστε ο πράκτορας να μπορεί να υλοποιήσει έναν μετρητή ο οποίος να μπορεί να μετρήσει μέχρι το n. Από τα t σημάδια που έχει ο κάθε πράκτορας, χρησιμοποιεί το ένα για να μαρκάρει την αφετηρία του και του απομένουν $t-1$ για να υλοποιήσει τον μετρητή. Περιγράφουμε τώρα τις τεχνικές λεπτομέρειες για την υλοποίηση του μετρητή με τα $t-1$ σημάδια που έχουν απομείνει. Όπως φαίνεται στο Σχήμα 4.5 ο μετρητής θα διαχωρίζεται με δύο σημάδια που βρίσκονται στους κόμβους A, B, σε μεταξύ τους απόσταση m. Το ένα σημάδι βρίσκεται στην



Σχήμα 4.5: Αυξάνοντας τον μετρητή ο οποίος διαχωρίζεται από δύο σημάδια στους κόμβους A, B σε απόσταση m ο ένας από τον άλλον αλλάζοντας τις θέσεις των σημαδιών που βρίσκονται στους ενδιάμεσους κόμβους. Ο κόμβος A είναι η αφετηρία (αριστερό διαχωριστικό) και ο B είναι το δεξί διαχωριστικό του πράκτορα. Για παράδειγμα τα $t-3$ εσωτερικά σημάδια (που έχουν αναπαρασταθεί με λευκούς κύκλους) μπορούν να χρησιμοποιηθούν για να μαρκάρουν $\binom{k-2}{t-3}$ τιμές του μετρητή. Το k είναι το τρέχον μήκος του μετρητή και m είναι το μέγιστο μήκος που απαιτείται έτσι ώστε ο μετρητής C_t να μπορεί να μετρήσει μέχρι το μέγεθος του δακτυλίου n.

αφετηρία A του πράκτορα. Απομένουν $t-1$ σημάδια. Άλλο ένα σημάδι στον κόμβο B αυξάνει το εύρος του μετρητή. Αφού ο πράκτορας μπορεί να μετρήσει μέχρι το $t-1$ (έχοντας $t-1$ σημάδια), όλοι οι δυνατοί συνδυασμοί από $t-3$ σημάδια μεταξύ δύο ακραίων σημαδιών σε απόσταση k

μπορούν να δοκιμαστούν και έπειτα να ακολουθήσει η αύξηση του k και κατόπιν να επαναληφτεί μέχρι να βρεθεί η αφετηρία του άλλου πράκτορα, όπου $k \leq m$.

Απομένει να μελετήσουμε ποιό είναι το μέγεθος του μετρητή που να εγγυάται ότι μπορεί να μετρήσει μέχρι το n. Για ένα δεδομένο k, υπάρχουν $\binom{k-2}{t-3}$ συνδυασμοί (θεωρώντας ότι δύο σημάδια δεν μπορούν να βρίσκονται στον ίδιο κόμβο). Αθροίζοντας για όλα τα $k \leq m$ μέχρι να φταστεί η αφετηρία του άλλου πράκτορα μας δίνει το πολύ

$$\sum_{k=2}^{m} \binom{k-2}{t-3} = \binom{m-1}{t-2}$$

συνδυασμούς (βλέπε [Knuth, 1997], σελ. 56). Αφού η αφετηρία του άλλου πράκτορα είναι το πολύ $n-1$ κόμβους μακρυά, ο μετρητής C_t χρειάζεται να μετρά μέχρι το $n-1$. Άρα η τιμή του m δεν χρειάζεται να ξεπεράσει το μικρότερο m έτσι ώστε $\binom{m-1}{t-2} \geq n-1$. Επιπλέον παρατηρήστε ότι οι δύο πράκτορες πρέπει οπωσδήποτε να έχουν μνήμη $O(\log t)$ bits έτσι ώστε να μπορούν να μετρήσουν μέχρι το t και συνεπώς να μπορέσουν να διακρίνουν τα δύο διαχωριστικά του μετρητή C_t. ■

Με απλούς υπολογισμούς βρίσκουμε ότι ο μικρότερος ακέραιος m έτσι ώστε $\binom{m-1}{t-2} \geq n-1$ είναι $O(n^{\frac{1}{t-2}}t)$ και συνεπώς ο χρόνος εκτέλεσης για τη λύση του προβλήματος της συνάντησης με ανίχνευση είναι $O(n^{\frac{t-1}{t-2}}t)$. Από το Θεώρημα 4.13 παίρνουμε το παρακάτω πόρισμα.

Πόρισμα 4.14 ([Czyzowicz et al., 2008]) Το πρόβλημα της συνάντησης με ανίχνευση ($\mathcal{RD}$) μπορεί να επιλυθεί από δύο πράκτορες που έχουν $t > 2$ μετακινήσιμα σημάδια και $O(\log t)$ μνήμη ο καθένας, σε χρόνο $O(n^{\frac{t-1}{t-2}}t)$ σε έναν δακτύλιο n κόμβων διπλής κατεύθυνσης. Επιπλέον, αν $t = \log n$ τότε ο χρόνος που χρειάζεται είναι $O(n \log n)$. ■

4.3 Συνάντηση με χρήση σημαδιών σε Torus

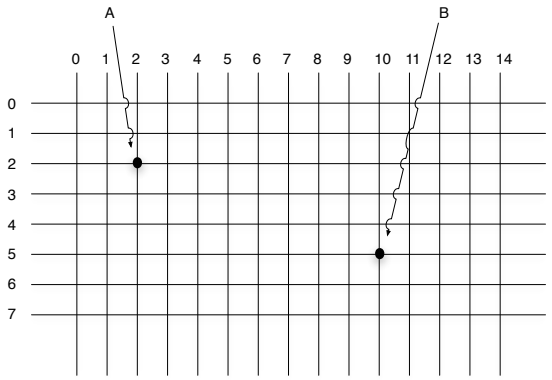
Κατά κάποιο τρόπο, όσο λιγότερο συμμετρικό είναι ένα γράφημα τόσο πιο εύκολο είναι να λυθεί το πρόβλημα της συνάντησης. Διαισθητικά αυτό συμβαίνει γιατί οι πράκτορες μπορούν να εκμεταλευτούν τις ασυμμετρίες (π.χ., τη ύπαρξη ενός διακεκριμένου κόμβου ή ενός μοναδικού υπογραφήματος με ειδική δομή) έτσι ώστε να συναντηθούν. Σε αυτήν την ενότητα μελετούμε το πρόβλημα της συνάντησης σε ένα torus αναδεικνύοντας μια ενδιαφέρουσα σχέση μεταξύ της συμμετρίας. του αριθμού των σημαδιών που χρησιμοποιούνται και της μνήμης που απαιτείται από τους πράκτορες.

Χρησιμοποιούμε εδώ το ίδιο ντετερμινιστικό μοντέλο με εκείνο της προηγούμενης ενότητας, με τη διαφορά της τοπολογίας. Ειδικότερα το μοντέλο μας αποτελείται από δύο ανώνυμους και πανομοιότυπους πράκτορες που είναι τοποθετημένοι σε ένα ανώνυμο, συγχρονισμένο torus με προσανατολισμό. Το torus αποτελείται από n δακτύλιους και κάθε ένας από αυτούς τους δακτύλιους

από m κόμβους. Αφού το torus είναι προσανατολισμένο μπορούμε να θεωρήσουμε ότι αποτελείται από n κάθετους δακτύλιους. Κάθε οριζόντιος δακτύλιος του torus αποτελείται από n κόμβους ενώ ένας κάθετος δακτύλιος αποτελείται από m κόμβους. Ονομάζουμε ένα τέτοιο torus ώς $n \times m$ torus. Οι πράκτορες συμφωνούν ως προς τον προσανατολισμό του torus. Κάθε πράκτορας έχει στη διάθεσή του έναν αριθμό από πανομοιότυπα σημάδια. Οι πράκτορες εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο και ξεκινούν την εκτέλεση την ίδια στιγμή και ενώ βρίσκονται στην ίδια αρχική κατάσταση.

Θεωρούμε ότι η μνήμη που χρειάζεται ένας πράκτορας είναι τουλάχιστον ανάλογη με τον αριθμό $\Theta(\log(\sigma))$ των bits που απαιτούνται για την κωδικοποίηση των $\sigma \geq 2$ καταστάσεων. Όσο του επιτρέπει η μνήμη του, ένας πράκτορας μπορεί να μετρήσει τους κόμβους που συναντά μεταξύ δύο σημαδιών, το πλήθος των κόμβων του torus, κλπ. Οι πράκτορες δεν γνωρίζουν το πλήθος των κόμβων του torus ή οποιαδήποτε άλλη παράμετρο του δικτύου, εκτός από τη διάστασή του.

Η συνάντηση συμβαίνει όταν οι πράκτορες είτε βρεθούν σε έναν κόμβο είτε διασχίσουν ταυτόχρονα από διαφορετικές κατευθύνσεις την ίδια ακμή του δικτύου. Θα μελετήσουμε τη *συνάντηση χωρίς ανίχνευση* ($\mathcal{RP}$) και τη *συνάντηση με ανίχνευση* ($\mathcal{RD}$).



Σχήμα 4.6: Δύο πράκτορες σε ένα 15×8 (δισδιάστατο) torus. Ο πράκτορας A έχει συντεταγμένες $(2, 2)$. Ο πράκτορας B έχει συντεταγμένες $(10, 5)$. Η απόστασή τους είναι $d(A, B) = (\min\{|A_1 - B_1|, (n - |A_1 - B_1|)\}, \min\{|A_2 - B_2|, (m - |A_2 - B_2|)\}) = (\min\{|2 - 10|, (15 - |2 - 10|)\}, \min\{|2 - 5|, (8 - |2 - 5|)\}) = (\min\{8, 7\}, \min\{3, 5\}) = (7, 3)$.

Η *απόσταση* μεταξύ δύο κόμβων (x_1, x_2) και (y_1, y_2) σε ένα δισδιάστατο torus $n \times m$, είναι ένα δισδιάστατο διάνυσμα (d_1, d_2) όπου $d_1 = \min\{|x_1 - y_1|, (n - |x_1 - y_1|)\}$ και $d_2 = \min\{|x_2 - y_2|, (m - |x_2 - y_2|)\}$. Ένα παράδειγμα δύο πρακτόρων σε ένα torus φαίνεται στο Σχήμα 4.6.

Θεώρημα 4.15 ([Kranakis et al., 2006]) Θεωρήστε δύο πανομοιότυπους πράκτορες τοποθετημένους σε ένα δισδιάστατο $(n \times m)$ προσανατολισμένο torus, έτσι ώστε η αρχική τους απόσταση

είναι είτε $(n/2, 0)$ (με το n άρτιο) είτε $(0, m/2)$ (με το m άρτιο) είτε $(n/2, m/2)$ (με τα n, m άρτια). Οι πράκτορες ξεκινούν στην ίδια αρχική κατάσταση. Τότε, οι πράκτορες δεν μπορούν να συναντηθούν σε κόμβο ή ακμή όσα μετακινήσιμα σημάδια και όση μνήμη και να έχουν.

Απόδειξη. Έστω D η αρχική απόσταση των πρακτόρων, με $D = (n/2, 0)$ ή $D = (0, m/2)$ ή $D = (n/2, m/2)$. Αφού οι πράκτορες ξεκινούν στην ίδια αρχική κατάσταση S_0, όσο δεν αφήνουν κάποιο σημάδι, μεταβαίνουν ταυτόχρονα στην ίδια κατάσταση S_t για οποιοδήποτε t, κινούνται στην ίδια κατεύθυνση, και συνεπώς διατηρούν την αρχική τους απόσταση. Έτσι όταν αφήσουν το πρώτο τους σημάδι, το κάνουν ταυτόχρονα τη στιγμή t_k, ενώ βρίσκονται στην ίδια κατάσταση S_k και έχοντας την αρχική απόσταση D. Συνεπώς η απόσταση μεταξύ των πρώτων σημαδιών που άφησαν οι πράκτορες είναι επίσης D, δηλαδή η απόσταση του πρώτου σημαδιού t_A που άφησε ο πράκτορας A, από το πρώτο σημάδι t_B που άφησε ο πράκτορας B είναι D. Για οποιαδήποτε χρονική στιγμή $t' > t_k$, και όσο οι πράκτορες δεν συναντούν κάποιο σημάδι, κινούντα προς την ίδια κατεύθυνση, μεταβαίνουν ταυτόχρονα στην ίδια κατάσταση $S_{t'}$ και διατηρούν την μεταξύ τους απόσταση D. Αν αφήσουν κάποιο σημάδι, το αφήνουν ταυτόχρονα και πάντα τα σημάδια που αφήνουν την ίδια χρονική στιγμή έχουν μεταξύ τους απόσταση D. Ας υποθέσουμε ότι ο ένας από τους πράκτορες, έστω ο A, συναντά κάποιο σημάδι. Τότε έχουμε τις εξής περιπτώσεις:

i) Έστω ότι ο A συναντά κάποιο σημάδι t_A το οποίο είχε ο ίδιος αφήσει παλιότερα σε κάποιο κόμβο (υπενθυμίζουμε ότι όλα τα σημάδια είναι πανομοιότυπα και συνεπώς ο A δεν γνωρίζει ότι το t_A το είχε αφήσει ο ίδιος). Την ίδια χρονική στιγμή, και αφού οι πράκτορες μέχρι εκείνη τη στιγμή κινούνται στην ίδια κατεύθυνση και έχουν απόσταση D, ο πράκτορας B συναντά το σημάδι t_B (το οποίο είχε αφήσει ο ίδιος νωρίτερα).

ii) Έστω ότι ο A συναντά κάποιο σημάδι t_B (το οποίο είχε αφήσει ο B νωρίτερα) at time t_l. Αυτό σημαίνει ότι τη στιγμή t_l, ο πράκτορας A βρίσκεται σε απόσταση D από το σημάδι t_A. Αφού μέχρι τη στιγμή t_l, οι πράκτορες κινούνταν στην ίδια κατεύθυνση, μεταβαίνοντας στις ίδιες καταστάσεις και καλύπτοντας την ίδια απόσταση, ο πράκτορας B βρίσκεται σε απόσταση D από το σημάδι t_B. Όμως αυτή είναι η θέση που βρίσκεται το σημάδι t_A. Συνεπώς τη στιγμή t_l, και οι δύο πράκτορες συναντούν σημάδια.

Με άλλα λόγια οι πράκτορες την ίδια χρονική στιγμή μεταβαίνουν στην ίδια κατάσταση, έχουν τον ίδιο αριθμό σημαδιών, και οι κόμβοι που βρίσκονται είτε δεν έχουν σημάδια είτε έχουν τον ίδιο αριθμό σημαδιών. Συνεπώς συμπεριφέρονται ακριβώς με τον ίδιο τρόπο και διατηρούν την μεταξύ τους απόσταση για πάντα. ■

Το Θεώρημα 4.15 αποτελεί γενίκευση του Θεωρήματος 4.5 το οποίο λέει ότι είναι αδύνατο για δύο πράκτορες που έχουν από ένα μη-μετακινήσιμο σημάδι ο καθένας, να συναντηθούν σε έναν δακτύλιο με n κόμβους αν η αρχική τους απόσταση είναι $n/2$, όπου το n είναι άρτιος.

4.3.1 Οι απαιτήσεις σε μνήμη

Παρουσιάζουμε κάτω φράγματα για τη μνήμη που χρειάζονται οι πράκτορες έτσι ώστε να συναντηθούν. Ας δούμε πρώτα πόσους κόμβους μπορεί να επισκεφτεί ένας πράκτορας σε σχέση με τη μνήμη του.

Λήμμα 4.16 Θεωρήστε έναν πράκτορα με $\sigma \geq 2$ καταστάσεις, δίχως σημάδια. Μπορούμε πάντα (για οποιοδήποτε αλγόριθμο) να επιλέξουμε ένα $n \times n$ προσανατολισμένο torus, όπου $n > \sigma$ έτσι ώστε, όποια και να είναι η αρχική θέση του πράκτορα, δεν μπορεί να επισκεφτεί όλους τους κόμβους του δικτύου. Ο πράκτορας μπορεί να επισκεφτεί το πολύ $n(\sigma - 1) + 1$ κόμβους.

Απόδειξη. Αν διαλέξουμε ένα προσανατολισμένο $n \times n$ torus, όπου $n > \sigma$ τότε ο πράκτορας πρέπει να επαναλάβει κάποια κατάσταση πριν επισκεφτεί όλους τους κόμβους. Έστω S η πρώτη κατάσταση που επαναλαμβάνει. Έστω $v = (v_x, v_y)$ ο κόμβος στον οποίο βρίσκεται ο πράκτορας όταν μεταβαίνει στην κατάσταση S για πρώτη φορά και v' ο κόμβος στον οποίο βρίσκεται ο πράκτορας όταν μεταβαίνει στην S για δεύτερη φορά. Καλούμε p_x, p_y την οριζόντια και κάθετη απόσταση αντίστοιχα μεταξύ των v και v'. Αφού η κατάσταση S είναι η πρώτη που επαναλαμβάνεται, ο συνολικός αριθμός των κόμβων που έχει επισκεφτεί ο πράκτορας μέχρι να επαναλάβει για πρώτη φορά την S είναι το πολύ $\sigma + 1$. Ειδικότερα, ο συνολικός αριθμός κόμβων που έχει επισκεφτεί ο πράκτορας μετά τη στιγμή που βρέθηκε για πρώτη φορά στην κατάσταση S και μέχρι να βρεθεί ξανά στην S (δηλαδή, των επισκέψεων μεταξύ των κόμβων v και v' χωρίς να μετράμε τον v) είναι το πολύ $\sigma - 1$ (παρατηρήστε ότι η αρχική κατάσταση S_0 εμφανίζεται μόνο στην αρχή).

Από τη στιγμή που ο πράκτορας βρίσκεται πάλι στην κατάσταση S πρέπει να επαναλάβει την ίδια τροχιά (p_x, p_y) και να επισκεφτεί άλλους το πολύ $\sigma - 1$ νέους κόμβους μέχρι να ξαναβρεθεί στην S. Ας ονομάσουμε τις συντεταγμένες των κόμβων του torus $0, \ldots, n-1$ οριζόντια και κάθετα. Αν v_x, v_y είναι οι συντεταγμένες του κόμβου v, τότε μετά από n επαναλήψεις της κατάστασης S, η θέση του πράκτορα είναι:

$$(v_x + np_x) \mod n = v_x$$

$$(v_y + np_y) \mod n = v_y$$

Αυτό σημαίνει ότι ο πράκτορας βρίσκεται ξανά στον κόμβο v και την κατάσταση S. Θα πρέπει να συνεχίσει την κίνηση περνώντας από τους ίδιους ακριβώς κόμβους. Μέχρι εκείνη τη στιγμή, ο πράκτορας έχει επισκεφτεί το πολύ $(\sigma + 1) + (n-1)(\sigma - 1) - 1 = n(\sigma - 1) + 1 < n^2$ κόμβους. ■

Ακολουθώντας παρόμοια μεθοδολογία μπορεί να αποδειχθεί το παρακάτω λήμμα.

Λήμμα 4.17 Έστω ένας πράκτορας με $\sigma \geq 2$ καταστάσεις και ένα μη-μετακινήσιμο σημάδι. Μπορούμε πάντα (για οποιοδήποτε αλγόριθμο) να επιλέξουμε ένα $n \times n$ προσανατολισμένο torus, όπου $n > \sigma^2$ έτσι ώστε, όποια και να είναι η αρχική θέση του πράκτορα, δεν μπορεί να επισκεφτεί

όλους τους κόμβους του δικτύου. Ο πράκτορας μπορεί να επισκεφτεί το πολύ $\sigma + (\sigma - 1)^2(n + 1) < n^2$ κόμβους. ■

Ας θεωρήσουμε τώρα δύο πράκτορες τοποθετημένους στο torus. Με βάση το γεγονός ότι ένας πράκτορας με ένα μη-μετακινήσιμο σημάδι δεν μπορεί να επισκεφτεί όλους τους κόμβους του torus (Λήμμα 4.17), ο adversary μπορεί να 'κρύψει' το σημάδι t_A σε έναν κόμβο που δεν επισκέπτεται ποτέ ο πράκτορας B και το σημάδι t_B σε έναν κόμβο που δεν επισκέπτεται ποτέ ο πράκτορας A. Με αυτή την τεχνική μπορούμε να αποδείξουμε ότι:

Λήμμα 4.18 Έστω δύο πράκτορες με σ καταστάσεις και ένα μη-μετακινήσιμο σημάδι ο καθένας. Μπορούμε πάντα (για οποιοδήποτε αλγόριθμο) να επιλέξουμε ένα $n \times n$ προσανατολισμένο torus, όπου $n > 2\sigma^2$ έτσι ώστε οι πράκτορες δεν μπορούν να συναντηθούν. ■

Σε αντίθεση με την περίπτωση των μη-μετακινήσιμων σημαδιών, αν οι πράκτορες μπορούν να μετακινήσουν τα σημάδια, τότε είναι εύκολο να σκεφτεί κανείς έναν αλγόριθμο με τον οποίον ένας πράκτορας επισκέπτεται όλους τους κόμβους οποιουδήποτε torus. Για παράδειγμα θεωρήστε τον παρακάτω αλγόριθμο για έναν πράκτορα με ένα μετακινήσιμο σημάδι.

Αλγόριθμος 17 (Επίσκεψη όλων των κόμβων ενός τόρου από έναν πράκτορα με σταθερή μνήμη)

1: Άφησε το σημάδι στον κόμβο εκκίνησης;
2: Κινήσου δεξιόστροφα μέχρι να συναντήσεις το δεύτερο σημάδι;
3: Μετακίνησε το σημάδι στο γειτονικό κόμβο προς τα κάτω;
4: **Επανέλαβε** από το βήμα 2;

Παρά το γεγονός αυτό, αποδεικνύουμε παρακάτω ότι και στην περίπτωση που οι πράκτορες μπορούν να μετακινήσουν τα σημάδια ο adversary μπορεί πάντα να επιλέξει τις αρχικές τους θέσεις έτσι ώστε να μπορούν να συναντήσουν και να μετακινήσουν μόνο το δικό τους σημάδι.

Ορισμός 4.19 Αν υπάρχουν δύο κόμβοι s, s' που είναι οι δύο κόμβοι εκκίνησης για τους πράκτορες A and B έτσι ώστε ο A αφήνει το σημάδι t_A σε έναν κόμβο που δεν επισκέπτεται ποτέ ο B και ο B αφήνει το σημάδι t_B σε έναν κόμβο που δεν επισκέπτεται ποτέ ο A τότε λέμε ότι οι s, s' ικανοποιούν την ιδιότητα π.

Λήμμα 4.20 Έστω δύο πράκτορες με σ καταστάσεις και ένα μετακινήσιμο σημάδι ο καθένας. Μπορούμε πάντα (για οποιοδήποτε αλγόριθμο) να επιλέξουμε ένα $n \times n$ προσανατολισμένο torus, όπου $n > 2\sigma^2$ έτσι ώστε η συνάντηση είναι αδύνατη.

Απόδειξη. Με βάση το Λήμμα 4.18, αν ισχύει $n > 2\sigma^2$ ο adversary μπορεί πάντα να τοποθετήσει αρχικά τους πράκτορες έτσι ώστε αν συναντήσουν κάποιο σημάδι, θα είναι το δικό τους σημάδι μέχρι τη στιγμή που αποφασίζουν να το μετακινήσουν. Έστω ότι κάποια στιγμή αποφασίζουν να μετακινήσουν το σημάδι τους. Αφού οι πράκτορες είναι πανομοιότυποι και ξεκινούν στην ίδια αρχική κατάσταση, θα συναντήσουν το σημάδι τους την ίδια στιγμή και θα αποφασίσουν να το μετακινήσουν ταυτόχρονα. Εφόσον μέχρι εκείνη τη στιγμή έχουν διατηρήσει την αρχική τους απόσταση,

θα τοποθετήσουν τα σημάδια τους και θα ξαναξεκινήσουν να κινούνται διατηρώντας την ίδια απόσταση και έτσι οι νέοι κόμβοι εκκίνησης συνεχίζουν να ικανοποιούν την ιδιότητα π. Συνεπώς δεν πρόκειται να συναντηθούν ποτέ. ■

Έτσι προκύπτει το επόμενο θεώρημα:

Θεώρημα 4.21 ([Kranakis et al., 2006]) Δύο πράκτορες με ένα μετακινήσιμο σημάδι ο καθένας, χρειάζονται τουλάχιστον $\Omega(\log n)$ μνήμη για να λύσουν το πρόβλημα της συνάντησης σε ένα $n \times n$ προσανατολισμένο torus.

Απόδειξη. Έστω πως οι πράκτορες έχουν r bits μνήμη. Συνεπώς μπορούν να έχουν το πολύ $\sigma = 2^r$ καταστάσεις. Από το Λήμμα 4.20 αν $n > 2\sigma^2$ οι πράκτορες δεν μπορούν να συναντηθούν. Συνεπώς χρειάζονται τουλάχιστον μνήμη $r = \Omega(\log n)$. ■

4.3.2 Αλγόριθμοι συνάντησης

Σε αυτήν την ενότητα αποδεικνύουμε ότι $O(\log n + \log m)$ μνήμη είναι αρκετή για τη λύση του προβλήματος της συνάντησης με ανίχνευση από δύο πράκτορες με ένα μη-μετακινήσιμο σημάδι ο καθένας σε ένα οποιοδήποτε $n \times m$ προσανατολισμένο torus. Συνεπώς τα προβλήματα $\mathcal{RP}$ και $\mathcal{RD}$ απαιτούν $\Theta(\log n + \log m)$ μνήμη σε ένα $n \times m$ προσανατολισμένο torus όταν οι πράκτορες έχουν ένα (μετακινήσιμο ή μη-μετακινήσιμο) σημάδι.

Επιπλέον μελετούμε την περίπτωση στην οποία οι πράκτορες έχουν δύο μετακινήσιμα σημάδια και σταθερή μνήμη και αποδεικνύουμε ότι το πρόβλημα $\mathcal{RD}$ μπορεί να λυθεί σε ένα $n \times n$ προσανατολισμένο torus.

4.3.2.1 Συνάντηση με μνήμη

Αρχικά περιγράφουμε έναν αλγόριθμο που λύνει το πρόβλημα $\mathcal{RD}$ για δύο πράκτορες που έχουν ένα μη-μετακινήσιμο σημάδι και $O(\log n + \log m)$ μνήμη ο καθένας σε οποιοδήποτε $n \times m$ προσανατολισμένο torus. Υπενθυμίζουμε ότι οι πράκτορες δεν γνωρίζουν τα n, m αλλά όπως θα δούμε μπορούν να χρησιμοποιήσουν τη μνήμη τους για να τα υπολογίσουν. Η περιγραφή του αλγόριθμου (Αλγόριθμος 18), δίνεται παρακάτω.

Αλγόριθμος 18 Αλγόριθμος για το $\mathcal{RD}$ σε ένα $n \times m$ προσανατολισμένο torus με 1 μη-μετακινήσιμο σημάδι και $O(\log n + \log m)$ μνήμη

```
1: SameRing
2: DifRing
```

Ο πράκτορας (και οι δύο πράκτορες εκτελούν τον ίδιο αλγόριθμο) ξεκινά να κινείται στον αρχικό οριζόντιο δακτύλιο. Αφήνει το σημάδι του και μετρά τους κόμβους που περνά μέχρι να συναντήσει κάποιο σημάδι δύο φορές. Αν οι αποστάσεις μεταξύ των σημαδιών είναι διαφορετικές, τότε

η συνάντηση είναι εφικτή. Αλλιώς κάνει το ίδιο στον αρχικό κάθετο δακτύλιο. Αν δεν έχει συναντήσει τον άλλον πράκτορα τότε διατρέχει έναν-έναν τους οριζόντιους δακτύλιους του torus μετρώντας τους κόμβους. Αν συναντήσει κάποιο σημάδι τη στιγμή που κινείται προς τα κάτω περνώντας από τον ένα οριζόντιο δακτύλιο στον άλλο, τότε η συνάντηση είναι αδύνατη. Αλλιώς, αν συναντήσει κάποιο σημάδι τη στιγμή που κινείται δεξιόστροφα σε έναν οριζόντιο δακτύλιο (το οποίο σημαίνει πως οι πράκτορες είχαν ξεκινήσει σε διαφορετικούς δακτύλιους), τότε: Αν τουλάχιστον ένας από τους μετρητές που μετρά τις οριζόντιες ή κάθετες αποστάσεις από το σημάδι του είναι διαφορετικός από $n/2$ ή $m/2$ αντίστοιχα, τότε η συνάντηση μπορεί να πραγματοποιηθεί. Αλλιώς σταματά και η συνάντηση είναι αδύνατη.

Όπως περιγράφεται από τον Αλγόριθμο 18, οι πράκτορες εκτελούν πρώτα την Διαδικασία `SameRing`. Αν δεν συναντηθούν τότε είτε είχαν ξεκινήσει σε συμμετρικές θέσεις στον ίδιο δακτύλιο ή είχαν ξεκινήσει σε διαφορετικούς δακτύλιους. Σε οποιαδήποτε περίπτωση εκτελούν τη Διαδικασία `DifRing`. Η εξερεύνηση τελειώνει μετά από το πολύ $O(nm)$ βήματα, ενώ χρειάζονται $O(\log n + \log m)$ μνήμη για το μέτρημα.

Procedure SameRing

1: Άφησε το σημάδι
2: Κινήσου δεξιόστροφα μετρώντας τους κόμβους μέχρι να δεις ένα σημάδι
3: $c_1 \leftarrow$αριθμός των κόμβων
4: Κινήσου δεξιόστροφα μετρώντας τους κόμβους μέχρι να δεις ένα σημάδι
5: $c_2 \leftarrow$αριθμός των κόμβων
6: **if** $c_2 \neq c_1$ **then**
7: Rendezvous(horizontal, c_1, c_2)
8: **else**
9: Κινήσου προς τα κάτω μετρώντας τους κόμβους μέχρι να δεις ένα σημάδι
10: $c_3 \leftarrow$αριθμός των κόμβων
11: Κινήσου προς τα κάτω μετρώντας τους κόμβους μέχρι να δεις ένα σημάδι
12: $c_4 \leftarrow$αριθμός των κόμβων
13: **if** $c_4 \neq c_3$ **then**
14: Rendezvous(vertical, c_3, c_4)
15: **end if**
16: **end if**

Λήμμα 4.22 Αν οι πράκτορες βρίσκονται αρχικά στον ίδιο δακτύλιο ενός $n \times m$ προσανατολισμένου torus σε μη-συμμετρικές θέσεις τότε η Διαδικασία `SameRing` θα τους οδηγήσει σε συνάντηση.

Απόδειξη. Μετά από $c_1 + c_2$ βήματα οι πράκτορες συναντούν τα σημάδια τους και άρα βρίσκονται πάλι στους κόμβους από τους οποίους ξεκίνησαν. Αν $c_1 \neq c_2$, αυτό σημαίνει ότι οι πράκτορες αρχικά βρίσκονταν στον ίδιο οριζόντιο δακτύλιο. Εκτελούν τη Διαδικασία `Rendezvous` σε αυτόν τον

Procedure Rendezvous(ring, c_1, c_2)

```
1:  if ring = horizontal then
2:     if c2 > c1 then
3:        Κινήσου δεξιόστροφα
4:     else
5:        Κινήσου αριστερόστροφα
6:     end if
7:  end if
8:  if ring = vertical then
9:     if c2 > c1 then
10:       Κινήσου προς τα κάτω
11:    else
12:       Κινήσου προς τα πάνω
13:    end if
14: end if
```

οριζόντιο δακτύλιο και συναντιούνται. Αν $c_2 = c_1$ τότε οι πράκτορες πρέπει αρχικά να βρίσκονταν στον ίδιο κάθετο δακτύλιο. Μετά από $c_3 + c_4$ βήματα προς τα κάτω συναντούν ξανά τα σημάδια τους στον κάθετο δακτύλιο με $c_3 \neq c_4$. Εκτελούν τη Διαδικασία `Rendezvous` σε αυτόν τον κάθετο δακτύλιο και συναντιούνται. ■

Με βάση το Λήμμα 4.22, αν μετά την εκτέλεση της Διαδικασίας `SameRing` οι πράκτορες δεν συναντηθούν, τότε είτε βρίσκονταν αρχικά στον ίδιο δακτύλιο σε συμμετρικές θέσεις οι βρίσκονταν σε διαφορετικούς δακτύλιους. Σε οποιαδήποτε περίπτωση πρέπει να ισχύει $c_1 = c_2$ και $c_3 = c_4$.

Λήμμα 4.23 Αν οι πράκτορες βρίσκονταν αρχικά στον ίδιο δακτύλιο σε συμμετρικές θέσεις ή σε διαφορετικούς δακτύλιους ενός $n \times m$ προσανατολισμένου torus τότε η Διαδικασία `DifRing` είναι ένας $\mathcal{RD}$ αλγόριθμος.

Απόδειξη. Οι πράκτορες εξερευνούν έναν προς έναν όλους τους οριζόντιους δακτύλιους, πηγαίνοντας αρχικά ένα βήμα προς τα κάτω και στη συνέχεια το πολύ $2c_1$ βήματα δεξιόστροφα. Εάν ένας πράκτορας συναντήσει κάποιο σημάδι τη στιγμή που πάει προς τα κάτω (περνώντας από τον έναν οριζόντιο δακτύλιο στον άλλον), τότε αυτό το σημάδι είναι είτε το δικό του σημάδι (που σημαίνει ότι οι πράκτορες ξεκίνησαν στον ίδιο οριζόντιο δακτύλιο) είτε το σημάδι του άλλου πράκτορα (που σημαίνει ότι οι πράκτορες ξεκίνησαν στον ίδιο κάθετο δακτύλιο). Σε κάθε περίπτωση η συνάντηση είναι αδύνατη. Εάν ένας πράκτορας συναντήσει κάποιο σημάδι ενώ κινείται δεξιόστροφα σε έναν οριζόντιο δακτύλιο τότε, έχοντας μετρήσει την απόσταση προς τα δεξιά (c_5) και προς τα κάτω (c_6) μεταξύ της αρχικής του θέσης και αυτού του σημαδιού, αν $c_5 \neq c_1/2$ ή $c_6 \neq c_3/2$ τότε οι πράκτορες μπορούν να συναντηθούν εύκολα εκτελώντας τη Διαδικασία `Rendezvous2`. Αλλιώς (όταν $c_5 = c_1/2$ και $c_6 = c_3/2$) η συνάντηση είναι αδύνατη. ■

Procedure DifRing

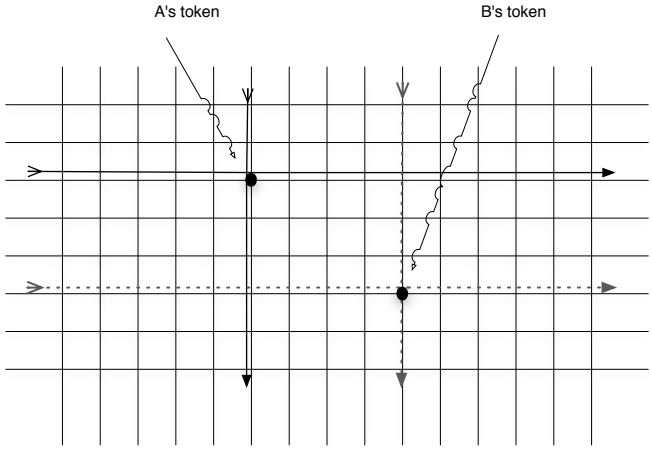
repeat
  Κινήσου προς τα κάτω στον επόμενο οριζόντιο δακτύλιο
  repeat
    Κινήσου δεξιόστροφα μετρώντας τους κόμβους
    $c_5 \leftarrow$ο αριθμός των κόμβων
  until ($c_5 = 2c_1$) OR (συνάντησες σημάδι)
until να συναντήσεις σημάδι
$c_6 \leftarrow$ο αριθμός των κόμβων προς τα κάτω
if (συνάντησες σημάδι ενώ πήγαινες προς τα κάτω) **then**
  σταμάτα: η συνάντηση είναι αδύνατη
else
  if $c_6 \neq c_3/2$ **then**
    Rendezvous2($c_6, c_3/2$)
  else
    if $c_5 \neq c_1/2$ **then**
      Rendezvous2($c_5, c_1/2$)
    else
      σταμάτα: η συνάντηση είναι αδύνατη
    end if
  end if
end if

Procedure Rendezvous2(ct, ck)

if $ct < ck$ **then**
  Άλλαξε οριζόντια κατεύθυνση και κινήσου reverse c_5 βήματα οριζόντια και μετά προς τα κάτω μέχρι να συναντήσεις σημάδι και περίμενε
end if
if $ct > ck$ **then**
  Περίμενε
end if

Ένα παράδειγμα φαίνεται στο Σχήμα 4.7. Ο Αλγόριθμος 18 μαζί με τα Λήμματα 4.22, 4.23 μας δίνουν το επόμενο θεώρημα.

Θεώρημα 4.24 ([Kranakis et al., 2006]) Το πρόβλημα της συνάντησης με ανίχνευση σε ένα $n \times m$ προσανατολισμένο torus μπορεί να λυθεί από δύο πράκτορες με ένα μη-μετακινήσιμο σημάδι και $O(\log n + \log m)$ μνήμη ο καθένας, σε χρόνο $O(nm)$.



Σχήμα 4.7: Δύο πράκτορες με $O(\log n + \log m)$ μνήμη και ένα μη-μετακινήσιμο σημάδι.

4.3.2.2 Συνάντηση με σταθερή μνήμη

Δίνουμε τώρα έναν αλγόριθμο που λύνει το πρόβλημα $\mathcal{RD}$ για δύο πράκτορες με δύο μετακινήσιμα σημάδια και σταθερή μνήμη σε οποιοδήποτε τετραγωνικό ανώνυμο, συγχρονισμένο και προσανατολισμένο torus.

Αρχικά περιγράφουμε και αναλύουμε μερικές διαδικασίες που χρησιμοποιούμε στους αλγόριθμους.

Procedure HorScan

1: **repeat**
2: Κινήσου κάτω, δεξιά, πάνω
3: **until** να συναντήσεις κάποιο σημάδι

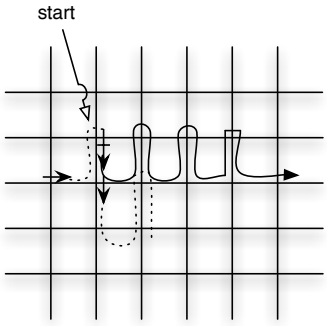
Στην Διαδικασία `HorScan` ο πράκτορας σταματά ακριβώς τη στιγμή που συναντά ένα σημάδι. Δηλαδή αν για παράδειγμα συναντά το σημάδι μετά την κίνηση προς τα δεξιά, σταματά χωρίς να κινηθεί προς τα πάνω.

Ο πράκτορας που εκτελεί τη Διαδικασία `FindTokenHor`, εξερευνά έναν προς έναν τους οριζόντιους δακτύλιους του torus όπως στο Σχήμα 4.8 μέχρι να συναντήσει κάποιο σημάδι μετά την κίνηση προς τα κάτω ή προς τα δεξιά. Παρακάτω αναλύουμε τη Διαδικασία `FindTokenHor` και αποδεικνύουμε κάποιες ιδιότητες.

Έστω ότι οι πράκτορες αφήνουν και τα δύο τους σημάδια στις αρχικές τους θέσεις και εκτελούν τη Διαδικασία `FindTokenHor`. Κατά τη διάρκεια της πρώτης εκτέλεσης της `HorScan` (βήμα 2 της Διαδικασίας `FindTokenHor`), ένας πράκτορας πρέπει να συναντήσει κάποιο σημάδι για πρώτη

Procedure FindTokenHor

1: **repeat**
2: HorScan
3: **if** συναντήσεις σημάδι μετά από κίνηση προς τα πάνω **then**
4: HorScan
5: Κινήσου ένα βήμα προς τα κάτω αφήνοντας (ή μετακινώντας) το δεύτερο σημάδι
6: **end if**
7: **until** να συναντήσεις σημάδι μετά από κίνηση προς τα κάτω ή προς τα δεξιά



Σχήμα 4.8: Ένας πράκτορας που εκτελεί τη Διαδικασία `FindTokenHor`

φορά, είτε μετά την κίνησή του προς τα κάτω στο πρώτο βήμα, είτε μετά την κίνησή του προς τα πάνω ή προς τα δεξιά σε κάποιο επόμενο βήμα (δεν είναι δυνατόν να συναντήσει κάποιο σημάδι ενώ κινείται προς τα κάτω σε κάποιο επόμενο βήμα της `HorScan` αφού θα έπρεπε να έχει συναντήσει κάποιο σημάδι νωρίτερα ενώ κινείται προς τα δεξιά).

Αν συναντήσει κάποιο σημάδι ενώ κινείται προς τα πάνω, αυτό το σημάδι μπορεί να το είχε αφήσει εκείνος ή ο άλλος πράκτορας. Εκτελώντας ξανά τη Διαδικασία `HorScan` (βήμα 4 της Διαδικασίας `FindTokenHor`), τότε είναι εύκολο να δούμε ότι το πρώτο σημάδι που συναντά τώρα βρίσκεται στον κόμβο από τον οποίο ξεκίνησε ο πράκτορας και το συνάντησε μετά την κίνηση προς τα πάνω. Επιπλέον, αυτό σημαίνει ότι ο από κάτω οριζόντιος δακτύλιος δεν είχε σημάδια. Με άλλα λόγια, αν ένας πράκτορας συναντήσει κάποιο σημάδι μετά την κίνηση προς τα πάνω, σημαίνει ότι είτε είναι το δικό του σημάδι (σε αυτήν την περίπτωση οι πράκτορες δεν ξεκίνησαν στον ίδιο οριζόντιο δακτύλιο) είτε είναι το σημάδι του άλλου πράκτορα (σε αυτήν την περίπτωση οι πράκτορες ξεκίνησαν στον ίδιο οριζόντιο δακτύλιο). Στη συνέχεια ο πράκτορας μετακινεί ένα από τα σημάδια του κατά έναν κόμβο προς τα κάτω (βήμα 5 της Διαδικασίας `FindTokenHor`) και επαναλαμβάνει από το βήμα 1. Παρατηρήστε ότι ένας πράκτορας δεν μετακινεί ποτέ όλα τα σημάδια από τον κόμβο εκκίνησης καιπάντα μετακινεί κάποιο σημάδι κατά μήκος του κάθετου δακτυλίου

που περιέχει τον κόμβο εκκίνησης. Ας συμβολίσουμε *πρώτο* (T_1) το σημάδι που μένει στον κόμβο εκκίνησης και *δεύτερο* (T_2) το σημάδι που μετακινεί ο πράκτορας.

Έστω ότι κάποια στιγμή ο πράκτορας βγαίνει από το βρόχο στο βήμα 7 επειδή συνάντησε ένα σημάδι κατά την κίνησή του προς τα κάτω. Αυτό είναι είτε το πρώτο του σημάδι (που σημαίνει ότι οι πράκτορες ξεκίνησαν στον ίδιο οριζόντιο δακτύλιο) είτε το πρώτο σημάδι του άλλου πράκτορα (που σημαίνει ότι οι πράκτορες ξεκίνησαν στον ίδιο κάθετο δακτύλιο). Συνεπώς, και στις δύο περιπτώσεις, η συνάντηση ενός σημαδιού μετά την κίνηση προς τα κάτω σημαίνει ότι οι πράκτορες ξεκίνησαν στον ίδιο δακτύλιο.

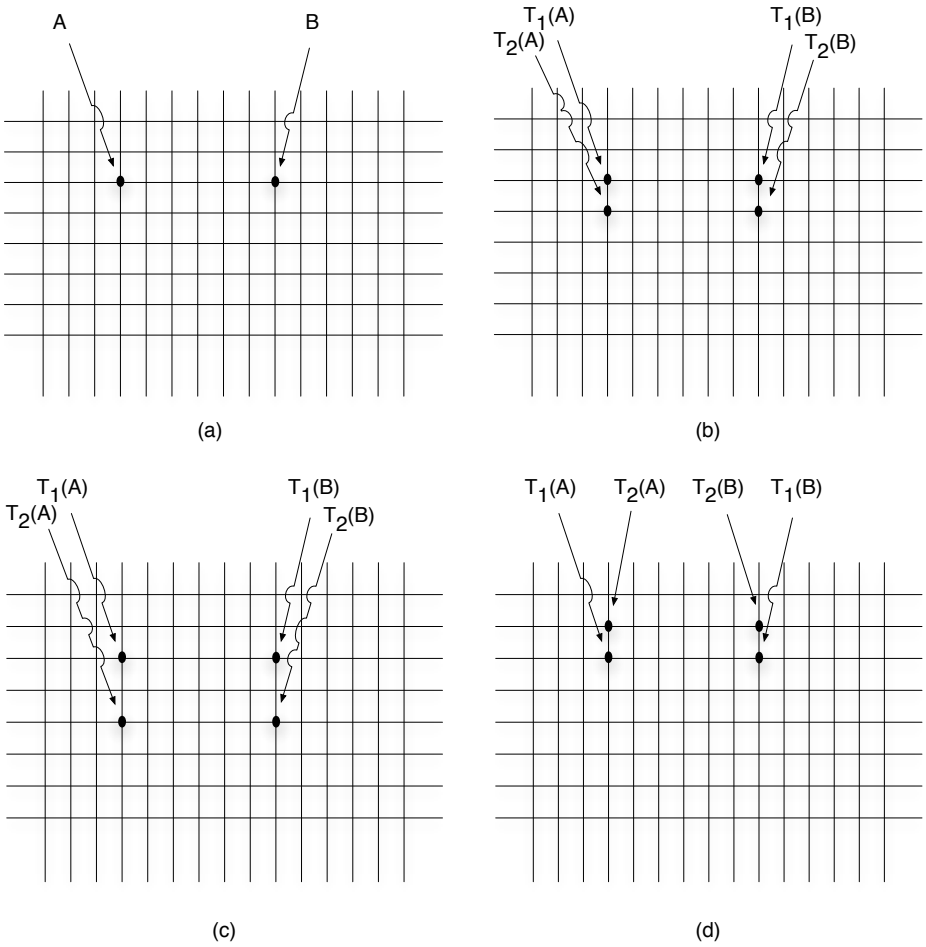
Αν ο πράκτορας βγαίνει από το βρόχο στο βήμα 7 επειδή συνάντησε ένα σημάδι κατά την κίνησή του προς τα δεξιά τότε είναι ξεκάθαρο πως αυτό είναι το πρώτο σημάδι του άλλου πράκτορα και οι πράκτορες είχαν ξεκινήσει σε διαφορετικούς οριζόνιους και κάθετους δακτύλιους.

Άρα ένας πράκτορας που εκτελεί τη Διαδικασία `FindTokenHor` γνωρίζοντας ότι έχει ξεκινήσει είτε στον ίδιο δακτύλιο με τον άλλον πράκτορα (αν βγήκε από το βρόχο στο βήμα 7 έχοντας συναντήσει ένα σημάδι μετά την κίνηση προς τα κάτω) είτε σε διαφορετικό οριζόντιο και κάθετο δακτύλιο με τον άλλον πράκτορα (αν βγήκε από το βρόχο στο βήμα 7 έχοντας συναντήσει ένα σημάδι μετά την κίνηση προς τα δεξιά).

Παρακάτω δίνουμε μια σειρά από παραδείγματα για την καλύτερη κατανόηση των διαδικασιών.

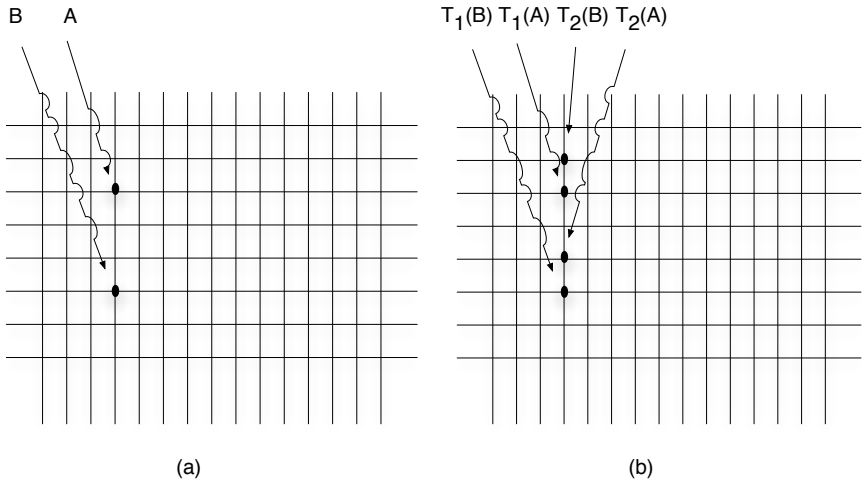
Παράδειγμα 1: Έστω ότι οι πράκτορες ξεκινούν στον ίδιο οριζόντιο δακτύλιο και αφήνουν τα σημάδια τους την ίδια στιγμή όπως φαίνεται στο Σχήμα 4.9(a). Όταν εκτελούν για πρώτη φορά τη Διαδικασία `HorScan` (στο βήμα 2 της Διαδικασίας `FindTokenHor`), ο πράκτορας A συναντά τα σημάδια του B ενώ κινείται προς τα πάνω και ο B συναντά τα σημάδια του A ενώ κινείται προς τα πάνω (ίσως όχι την ίδια στιγμή). Έπειτα εκτελούν ξανά τη Διαδικασία `HorScan` (στο βήμα 4 της Διαδικασίας `FindTokenHor`) και συναντούν τα σημάδια τους ενώ κινούνται προς τα πάνω (την ίδια στιγμή αφού οι πράκτορες έχουν διανύσει την ίδια απόσταση). Στη συνέχεια μετακινούν ένα σημάδι προς τα κάτω (όπως φαίνεται στο Σχήμα 4.9(b)) και επαναλαμβάνουν από το βήμα 1 της Διαδικασίας `FindTokenHor`. Τώρα ο πράκτορας A συναντά το δεύτερο σημάδι ($T_2(B)$) του B ενώ κινείται προς τα πάνω και ο B συναντά το δεύτερο σημάδι ($T_2(A)$) του A ενώ κινείται προς τα πάνω (εκτελώντας την Διαδικασία `HorScan` στο βήμαat 2 της Διαδικασίας `FindTokenHor`). Έπειτα εκτελούν τη Διαδικασία `HorScan` στο βήμα 4 της Διαδικασίας `FindTokenHor` και συναντούν τα δικά τους δεύτερα σημάδια την ίδια στιγμή ενώ κινούνται προς τα πάνω. Στη συνέχεια μετακινούν αυτά τα σημάδια έναν κόμβο προς τα κάτω όπως στο Σχήμα 4.9(c) και επαναλαμβάνουν από το βήμα 1. Λίγο αργότερα θα μετακινήσουν το δεύτερο σημάδι τους όπως στο Σχήμα 4.9(d) (στον οριζόντιο δακτύλιο που βρίσκεται πάνω από τον αρχικό οριζόντιο δακτύλιο) και κατόπιν θα κινηθούν προς τα κάτω (εκτελώντας τη Διαδικασία `HorScan` στο βήμα 2 της Διαδικασίας `FindTokenHor`), συναντώντας (την ίδια στιγμή) ένα σημάδι και βγαίνοντας έτσι από το βρόχο στο βήμα 7. Συνεπώς σωστά συμπεραίνουν ότι είχαν ξεκινήσει στον ίδιο δακτύλιο.

Παράδειγμα 2: Έστω ότι οι πράκτορες ξεκινούν στον ίδιο κάθετο δακτύλιο και αφήνουν και τα δύο τους σημάδια την ίδια στιγμή στους κόμβους εκκίνησης όπως στο Σχήμα 4.10(a). Οι



Σχήμα 4.9: Δύο πράκτορες A και B που ξεκινούν στον ίδιο οριζόντιο δακτύλιο: (a) Κάθε πράκτορας αφήνει και τα δύο του σημάδια στον κόμβο εκκίνησης. (b) Κάθε πράκτορας μετακινεί ένα από τα σημάδια του προς τα κάτω. (c) Κάθε πράκτορας μετακινεί επαναληπτικά το ίδιο σημάδι T_2 προς τα κάτω. (d) Τελικά κάθε πράκτορας μετακινεί το σημάδι του (T_2) στον οριζόντιο δακτύλιο που βρίσκεται πάνω από τον οριζόντιο δακτύλιο που ήταν τοποθετημένος αρχικά.

πράκτορες θα συναντούν πάντα (την ίδια στιγμή) κάποιο σημάδι ενώ πηγαίνουν προς τα πάνω εκτελώντας την Διαδικασία `HorScan` στο βήμα 2 ή 4 της Διαδικασίας `FindTokenHor`, μέχρι να τοποθετήσουν (όχι αναγκαστικά την ίδια στιγμή) το δεύτερό τους σημάδι στον οριζόντιο δακτύλιο που βρίσκεται επάνω από τον αρχικό οριζόντιο δακτύλιο του άλλου όπως στο Σχήμα 4.10(b). Στη συνέχεια, όταν εκτελούν τη Διαδικασία `HorScan` στο βήμα 2 της Διαδικασίας `FindTokenHor`,

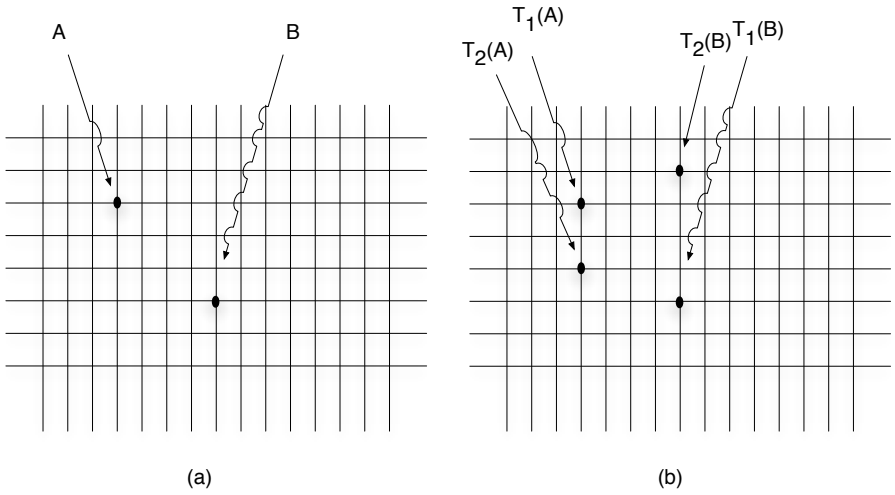


Σχήμα 4.10: Δύο πράκτορες A και B ξεκινούν στον ίδιο κάθετο δακτύλιο: (a) Κάθε πράκτορας αφήνει και τα δύο του σημάδια στον κόμβο εκκίνησης. (b) Τελικά κάθε πράκτορας τοποθετεί ένα σημάδι του (T_2) στον οριζόντιο δακτύλιο που βρίσκεται πάνω από τον αρχικό οριζόντιο δακτύλιο του άλλου.

συναντούν το πρώτο σημάδι του άλλου πράκτορα (όχι αναγκαστικά την ίδια στιγμή) ενώ κινούνται προς τα κάτω. Συνεπώς συμπεραίνουν σωστά ότι είχαν ξεκινήσει στον ίδιο δακτύλιο.

Παράδειγμα 3: Έστω ότι οι πράκτορες ξεκινούν (την ίδια στιγμή) σε διαφορετικό οριζόντιο και κάθετο δακτύλιο και αφήνουν τα σημάδια τους όπως στο Σχήμα 4.11(a). Οι πράκτορες θα συναντούν πάντα (την ίδια στιγμή) κάποιο σημάδι ενώ κινούνται προς τα πάνω εκτελώντας τη Διαδικασία `HorScan` στο βήμα 2 ή 4 της Διαδικασίας `FindTokenHor`, μέχρι να τοποθετήσουν (όχι αναγκαστικά την ίδια στιγμή) το δεύτερό τους σημάδι στον οριζόντιο δακτύλιο που βρίσκεται πάνω από τον αρχικό οριζόντιο δακτύλιο του άλλου πράκτορα όπως στο Σχήμα 4.11(b). Τώρα, καθώς εκτελούν τη Διαδικασία `HorScan` στο βήμα 2 της Διαδικασίας `FindTokenHor`, θα συναντήσουν (πιθανόν όχι ταυτόχρονα) κάποιο σημάδι ενώ κινούνται προς τα δεξιά και θα βγουν από το βρόχο συμπεραίνοντας σωστά ότι ξεκίνησαν σε διαφορετικούς δακτύλιους.

Παρατηρήστε ότι στα Παραδείγματα 2, 3, αν η αρχική κάθετη απόσταση μεταξύ των πρακτόρων είναι 1 τότε τουλάχιστον ένας από τους πράκτορες (ή και οι δύο αν το torus αποτελείται από 2 μόνο οριζόντιους δακτύλιους) θα συναντήσει το πρώτο σημάδι του άλλου πράκτορα αμέσως μετά την κίνησή του προς τα κάτω (στο Παράδειγμα 2) ή προς τα δεξιά (στο Παράδειγμα 3) ενώ εκτελεί τη Διαδικασία `HorScan` στο βήμα 2 της Διαδικασίας `FindTokenHor`. Δηλαδή σε αυτήν την περίπτωση ο πράκτορας δεν μετακινεί όλα τα σημάδια του. Και σε αυτήν την περίπτωση τα συμπεράσματα που βγάζει ο πράκτορας είναι σωστά. Χρησιμοποιούμε επίσης τις Διαδικασίες `VerScan` και `FindTokenVer` με τις οποίες οι πράκτορες εξερευνούν έναν προς έναν τους κάθετους δακτύλιους του torus όπως στο Σχήμα 4.12. Οι Διαδικασίες `VerScan` και `FindTokenVer` έχουν τις



Σχήμα 4.11: Δύο πράκτορες A και B ξεκινούν σε διαφορετικό οριζόντιο και κάθετο δακτύλιο: (a) Κάθε πράκτορας αφήνει και τα δύο του σημάδια στον κόμβο εκκίνησης. (b) Τελικά κάθε πράκτορας τοποθετεί ένα σημάδι του (T_2) στον οριζόντιο δακτύλιο που βρίσκεται πάνω από τον αρχικό οριζόντιο δακτύλιο του άλλου.

Procedure VerScan

1: **repeat**
2: Κινήσου δεξιά, κάτω, αριστερά
3: **until** να συναντήσεις κάποιο σημάδι

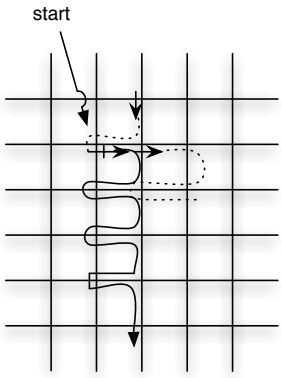
ίδιες ιδιότητες με τις Διαδικασίες `HorScan` και `FindTokenHor` αντίστοιχα, αντικαθιστώντας την κατεύθυνση 'κάτω' με την κατεύθυνση 'δεξιά', την κατεύθυνση 'δεξιά' με την κατεύθυνση 'κάτω' και την κατεύθυνση 'επάνω με την κατεύθυνση 'αριστερά'. Αν οι πράκτορες έχουν ξεκινήσει σε διαφορετικό οριζόντιο και κάθετο δακτύλιο τότε (ομοίως όπως προηγουμένως) εκτελώντας την Διαδικασία `FindTokenVer` θα τελειώσουν, συναντώντας (όχι αναγκαστικά ταυτόχρονα) κάποιο σημάδι ενώ κινούνται προς τα κάτω (βλέπε Σχήμα 4.13). Και οι δύο διαδικασίες `FindTokenHor` και `FindTokenVer` χρειάζονται χρόνο $O(nm)$.

Χρησιμοποιούμε επίσης τον Αλγόριθμο 16 που έχει παρουσιαστεί στην ενότητα 4.2.2 ο οποίος λύνει το πρόβλημα της συνάντησης με ανίχνευση σε έναν προσανατολισμένο δακτύλιο με δύο πράκτορες που έχουν δύο σημάδια και σταθερή μνήμη.

Συνδυάζοντας τις παραπάνω διαδικασίες περιγράφουμε τώρα την κύρια Διαδικασία `SearchTorus` η οποία θα χρησιμοποιηθεί στον Αλγόριθμο 19 ο οποίος λύνει το πρόβλημα $\mathcal{RD}$ για δύο πράκτορες με σταθερή μνήμη σε ένα $n \times n$ προσανατολισμένο torus.

Procedure FindTokenVer

1: **repeat**
2: VerScan
3: **if** συναντήσεις κάποιο σημάδι ενώ κινείσαι αριστερά **then**
4: VerScan
5: Κινήσου ένα βήμα δεξιά και άφησε (ή μετακίνησε εκεί) το δεύτερο σημάδι
6: **end if**
7: **until** να συναντήσεις ένα σημάδι ενώ κινείσαι προς τα κάτω



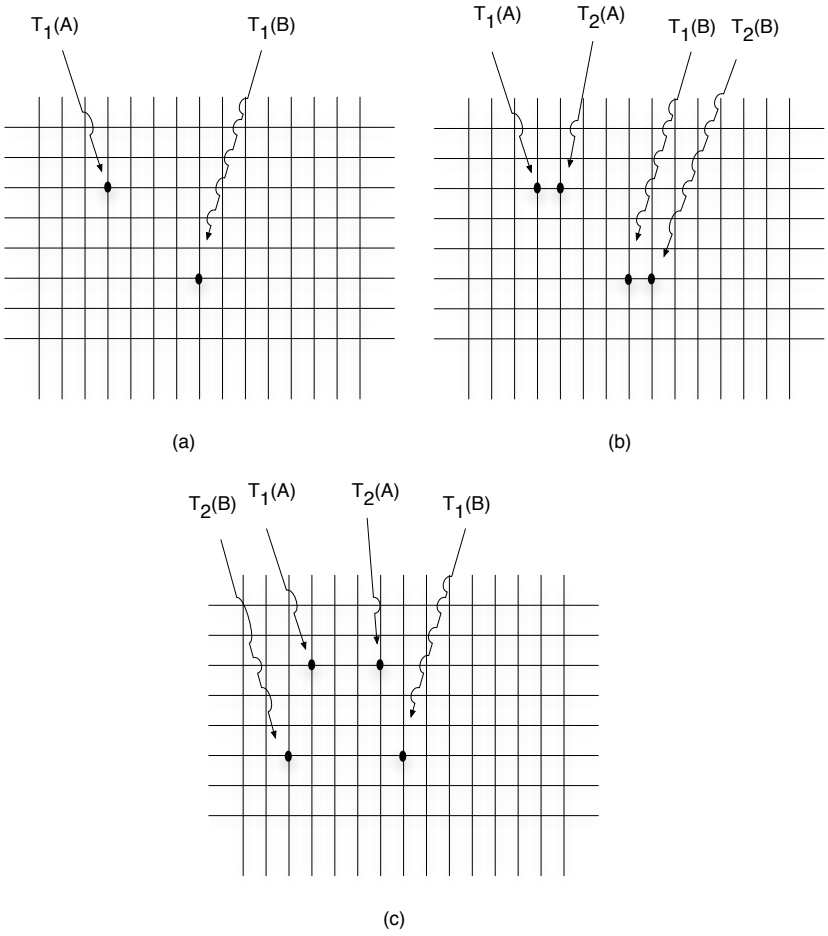
Σχήμα 4.12: Ένας πράκτορας που εκτελεί την Διαδικασία `FindTokenVer`.

Οι πράκτορες εξερευνούν έναν προς έναν τους οριζόντιους δακτύλιους του torus (εκτελώντας την Διαδικασία `FindTokenHor`) για να ανακαλύψουν αν είχαν ξεκινήσει στον ίδιο δακτύλιο. Αν ναι, τότε εκτελούν τον Αλγόριθμο 16 της Ενότητας 4.2.2. Αλλιώς προσπαθούν να προλάβουν ο ένας τον άλλον στο torus χρησιμοποιώντας ένα μονοπάτι που έχουν σημαδέψει με τα σημάδια τους. Αν δεν συναντηθούν τότε εξερευνούν έναν προς έναν τους κάθετους δακτύλιους του torus (εκτελώντας τη Διαδικασία `FindTokenVer`). Προσπαθούν πάλι να προλάβουν ο ένας τον άλλον στο torus. Αν αυτή τη φορά δεν συναντηθούν συμπεραίνουν πως η συνάντηση είναι αδύνατη. Ο Αλγόριθμος 19 χρειάζεται $O(n^2)$ χρόνο.

Έτσι προκύπτει το επόμενο θεώρημα.

Θεώρημα 4.25 ([Kranakis et al., 2006]) Το πρόβλημα της συνάντησης με ανίχνευση σε ένα $n \times n$ προσανατολισμένο torus μπορέι να λυθεί με δύο πράκτορες που χρησιμοποιούν δύο μετακινήσιμα σημάδια ο καθένας και έχουν σταθερή μνήμη, σε χρόνο $O(n^2)$. ■

Ο ενδιαφερόμενος αναγνώστης μπορεί να βρει τις λεπτομέρειες της απόδειξης του Θεωρήματος 4.25 στο άρθρο [Kranakis et al., 2006].



Σχήμα 4.13: Δύο πράκτορες A και B ξεκινούν σε διαφορετικό οριζόντιο και κάθετο δακτύλιο: (a) Κάθε πράκτορας αφήνει και τα δύο του σημάδια στον κόμβο εκκίνησης. (b) Κάθε πράκτορας μετακινεί ένα από τα σημάδια του προς τα δεξιά. (c) Τελικά κάθε πράκτορας μετακινεί ένα σημάδι του (T_2) στον κάθετο δακτύλιο που βρίσκεται αριστερά από τον αρχικό κάθετο δακτύλιο του άλλου πράκτορα.

Τελειώνουμε αυτήν την ενότητα με τις εξής παρατηρήσεις.

Σε ένα torus, φαίνεται ότι υπάρχει μια γνήσια ιεραρχία σε σχέση με τη δύναμη των σημαδιών και της μνήμης για την επίλυση του προβλήματος της συνάντησης: οποιοσδήποτε σταθερός αριθμός από μη-μετακινήσιμα σημάδια είναι λιγότερο δυνατό από δύο μετακινήσιμα σημάδια. Ενώ η ιεραρχία σταματά στα τρία σημάδια (ένας αλγόριθμος για συνάντηση με ανίχνευση σε ένα $n \times m$ torus για πράκτορες με σταθερή μνήμη έχει παρουσιαστεί στο άρθρο [Kranakis et al., 2006]), πα-

Procedure SearchTorus

```
1: Άφησε τα δύο σημάδια
2: FindTokenHor
3: if συναντήσεις κάποιο σημάδι ενώ κινείσαι προς τα κάτω then
4:    (* Ξεκίνησαν στον ίδιο δακτύλιο *)
5:    Κινήσου ένα βήμα προς τα πάνω
6:    if συναντήσεις ακριβώς ένα σημάδι then
7:       πάρε το σημάδι
8:       κινήσου προς τα επάνω μέχρι να συναντήσεις κάποιο σημάδι
9:       άφησε το σημάδι
10:   end if
11:   Αλγόριθμος 16 της Ενότητας 4.2.2 στον κάθετο δακτύλιο
12:   Αλγόριθμος 16 στον οριζόντιο δακτύλιο
13:   if δεν υπάρξει συνάντηση then
14:      σταμάτα: η συνάντηση είναι αδύνατη
15:   end if
16: else
17:   (* Ξεκίνησαν σε διαφορετικό δακτύλιο *)
18:   κινήσου προς τα επάνω μέχρι να συναντήσεις κάποιο σημάδι
19:   Κινήσου ένα βήμα προς τα κάτω
20:   repeat
21:      κινήσου ένα βήμα προς τα αριστερά, περίμενε για ένα βήμα
22:   until (να υπάρξει συνάντηση) OR (να συναντήσεις κάποιο σημάδι για δεύτερη φορά)
23:   if δεν υπάρξει συνάντηση then
24:      Synchronize
25:      άφησε ένα σημάδι
26:      FindTokenVer
27:      κινήσου προς τα αριστερά μέχρι να συναντήσεις κάποιο σημάδι
28:      κινήσου ένα βήμα προς τα δεξιά
29:      repeat
30:         κινήσου ένα βήμα προς τα πάνω, περίμενε για ένα βήμα
31:      until (να υπάρξει συνάντηση) OR (να συναντήσεις κάποιο σημάδι για δεύτερη φορά)
32:   end if
33: end if
```

ραμένει άγνωστο αν τρία σημάδια είναι αυστηρά πιο δυνατά από δύο σε σχέση με το πρόβλημα της συνάντησης με ανίχνευση. Άλλο ένα ανοιχτό ενδιαφέρον πρόβλημα είναι το πλήθος των σημαδιών που χρειάζονται για τη συνάντηση με ή χωρίς ανίχνευση σε ένα torus χωρίς προσανατολισμό όταν οι πράκτορες έχουν σταθερή μνήμη. Πιστεύουμε πως οι αλγόριθμοι αυτής της ενότητας είναι δυ-

Procedure Synchronize

Κινήσου προς τα πάνω
repeat
    περίμενε για ένα βήμα, κινήσου ένα βήμα προς τα αριστερά
until να συναντήσεις κάποιο σημάδι
πάρε το σημάδι
if δεν υπάρχουν σημάδια στον κόμβο **then**
    (σημαίνει ότι η αρχική απόσταση των πρακτόρωνήταν* > 1 *προς τα κάτω *)*
    κινήσου προς τα πάνω μέχρι να συναντήσεις κάποιο σημάδι
end if
άφησε το σημάδι
FindTokenHor
κινήσου ένα βήμα προς τα πάνω
κινήσου προς τα δεξιά μέχρι να συναντήσεις κάποιο σημάδι
πάρε το σημάδι
κινήσου ένα βήμα προς τα κάτω
κινήσου προς τα δεξιά μέχρι να συναντήσεις κάποιο σημάδι

Αλγόριθμος 19 RVD2n

SearchTorus
if δεν υπάρξει συνάντηση **then**
    σταμάτα: η συνάντηση είναι αδύνατη
end if

νατόν να επεκταθούν και με τη χρήση μερικών επιπλέον σημαδιών να επιλύσουν το πρόβλημα σε ένα torus χωρίς προσανατολισμό.

4.4 Σχόλια και βιβλιογραφικές παρατηρήσεις

Ο κάτοχος βραβείου Nobel, T. C. Schelling θεωρείται γενικώς εκείνος που εισήγαγε το πρόβλημα της συνάντησης στο βιβλίο του [Schelling, 1960] σχετικά με στρατηγικές σύγκρουσης. Σύμφωνα με το άρθρο [Alpern and Gal, 2003][σελ. xii] ένας πιο μαθηματικός φορμαλισμός προτάθηκε από τον Steve Alpern το 1976, ειδικότερα για το Πρόβλημα των Τηλεφώνων: Σε κάθε ένα από δύο δωμάτια έχουν τοποθετηθεί n τηλεφωνα συνδεδεμένα τυχαία. Σε διακριτές στιγμές $t = 0, 1, 2, \ldots$ δύο παίκτες, ένας σε κάθε δωμάτιο, σηκώνουν τα τηλέφωνα και προσπαθούν να επικοινωνήσουν. Πώς μπορούν να ελαχιστοποιήσουν το χρόνο t έτσι ώστε να χρησιμοποιήσουν τηλέφωνα που είναι συνδεδεμένα μεταξύ τους και να επικοινωνήσουν; Ένας πιο ακριβής μαθηματικός φορμαλισμός του προβλήματος παρουσιάστηκε στο άρθρο [Alpern, 1995].

Το πρόβλημα της συνάντησης έχει επίσης μελετηθεί σαν ένα παιχνίδι αναζήτησης για δύο πράκτορες τοποθετημένους σε μια συνεχή ή διακριτή περιοχή με σκοπό να ελαχιστοποιηθεί ο χρόνος που απαιτείται για συνάντηση με δεδομένη κάποια αρχική κατανομή των πρακτόρων. Λεπτομέρειες για αυτή την περίπτωση του προβλήματος μπορούν να βρεθούν στα άρθρα [Alpern, 1995, Alpern, 2002, Alpern, 2001, Alpern and Gal, 1995, Alpern and Gal, 2002, Baston and Gal, 1998, Baston and Gal, 2001] ενώ στο βιβλίο [Alpern and Gal, 2003] ο ενδιφερόμενος αναγνώστης μπορεί να βρει μια ανάλυση του προβλήματος χρησιμοποιώντας τεχνικές της Θεωρίας Παιγνίων.

Το μοντέλο για το πρόβλημα της συνάντησης που μελετήθηκε σε αυτό το κεφάλαιο έχει περισσότερο σχέση με την περιοχή της Θεωρητικής Πληροφορικής. Οι σημαντικότερες διαφορές με άλλα μοντέλα είναι ότι δίνει έμφαση στο συνδυασμό παραμέτρων διακριτής βελτιστοποίησης και κατανεμημένων υπολογισμών στο δίκτυο, στις ικανότητες υπολογισμού που έχουν οι πράκτορες, στη χρονική πολυπλοκότητα και στο ισοζύγιο μεταξύ αυτών.

Αναφορικά με κάτω φράγματα του χρόνου που απαιτείται για συνάντηση πρακτόρων με περισσότερα από δύο σημάδια, πολύ λίγα είναι γνωστά. Για την περίπτωση δακτυλίων μονής κατεύθυνσης αναφέρουμε το αποτέλεσμα [Czyzowicz et al., 2008][Θεώρημα 7] το οποίο αποδεικνύει μια σχέση μεταξύ του χρόνου συνάντησης και του αριθμού των σημαδιών που έχει κάθε πράκτορας. Οι συγγραφείς δείχνουν ότι το πρόβλημα συνάντησης χωρίς ανίχνευση ($\mathcal{RP}$) για δύο πράκτορες που έχουν σταθερή μνήμη και t μετακινήσιμα σημάδια ο καθένας απαιτεί $\Theta(n^2/t)$ χρόνο σε ένα δακτύλιο μονής κατεύθυνσης με n κόμβους.

Βιβλιογραφία

[Alpern, 1995] Alpern, S. (1995). The rendezvous search problem. *SIAM Journal of Control and Optimization*, 33:673–683.

[Alpern, 2001] Alpern, S. (2001). Rendezvous in one and more dimensions. Technical report, The London School of Economics.

[Alpern, 2002] Alpern, S. (2002). Rendezvous search: A personal perspective. *Operations Research*, 50(5):772–795.

[Alpern and Gal, 1995] Alpern, S. and Gal, S. (1995). Rendezvous search on the line with distinguishable players. *SIAM Journal Control and Optimization*, 33:1270–1276.

[Alpern and Gal, 2002] Alpern, S. and Gal, S. (2002). Searching for an agent who may or may not want to be found. *Operations Research*, 50(2):311–323.

[Alpern and Gal, 2003] Alpern, S. and Gal, S. (2003). *The Theory of Search Games and Rendezvous*. Kluwer Academic Publishers.

[Apostol, 1997] Apostol, T. M. (1997). *Introduction to Analytical Number Theory*. Springer Verlag.

[Attiya et al., 1988] Attiya, H., Snir, M., and Warmuth, M. K. (1988). Computing on an anonymous ring. *J. of the ACM*, 35(4):845–875.

[Baston and Gal, 1998] Baston, V. and Gal, S. (1998). Rendezvous on the line when the players' initial distance is given by an unknown probability distribution. *SIAM Journal of Control and Optimization*, 36(6):1880–1889.

[Baston and Gal, 2001] Baston, V. and Gal, S. (2001). Rendezvous search when marks are left at the starting points. *Naval Research Logistics*, 47(6):722–731.

[Cormen et al., 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT press.

[Czyzowicz et al., 2008] Czyzowicz, J., Dobrev, S., Kranakis, E., and Krizanc, D. (2008). The power of tokens: Rendezvous and symmetry detection for two mobile agents in a ring. In *Proceedings of 34th International Conference on Current Trends in Theory and Practice of Computer Science*, LNCS 4910, pages 234–246.

[Knuth, 1997] Knuth, D. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley.

[Kranakis et al., 2006] Kranakis, E., Krizanc, D., and Markou, E. (2006). Mobile agent rendezvous in a synchronous torus. In *Proceedings of 7th Latin American Theoretical Informatics Symposium*, LNCS 3887, pages 653–664.

[Kranakis et al., 2003] Kranakis, E., Krizanc, D., Santoro, N., and Sawchuk, C. (2003). Mobile agent rendezvous search problem in the ring. In *Proceedings of International Conference on Distributed Computing Systems*, pages 592–599.

[Sawchuk, 2004] Sawchuk, C. (2004). *Mobile Agent Rendezvous in the Ring*. PhD thesis, School of Computer Science, Carleton University, Ottawa, Canada.

[Schelling, 1960] Schelling, T. C. (1960). *The Strategy of Conflict*. Harvard University Press, Cambridge, MA.

ΚΕΦΑΛΑΙΟ 5

Το Πρόβλημα της Συνάντησης Πολλών Πρακτόρων

Σε αυτό το κεφάλαιο μελετούμε το πρόβλημα της συνάντησης πολλών πρακτόρων. Αρχικά μελετούμε το πρόβλημα για πράκτορες που μπορούν να μεταφέρουν από ένα μη-μετακινήσιμο σημάδι, σε ένα συγχρονισμένο και προσανατολισμένο δακτύλιο. Παρουσιάζουμε μια σειρά από αλγόριθμους για τα προβλήματα της συνάντησης με ανίχνευση και χωρίς ανίχνευση. Στη συνέχεια μελετούμε το πρόβλημα σε ασύγχρονους δακτύλιους, χρησιμοποιώντας το μοντέλο Look-Compute-Move. Μετά την τυπική περιγραφή του μοντέλου, παρουσιάζουμε αρνητικά αποτελέσματα και ντετερμινιστικούς αλγόριθμους.

5.1 Συνάντηση με σημάδια σε συγχρονισμένο δακτύλιο

Σε αυτήν την ενότητα μελετούμε το πρόβλημα της συνάντησης για $k \geq 2$ κινητούς πράκτορες που μπορούν να μεταφέρουν σημάδια, σε έναν συγχρονισμένο και προσανατολισμένο δακτύλιο n κόμβων. Όπως και στην περίπτωση των δύο πρακτόρων, αρχικά αναλύουμε τις συνθήκες κάτω από τις οποίες μπορεί να λυθεί το πρόβλημα. Έπειτα δίνουμε αλγόριθμους για την επίλυση της συνάντησης με ανίχνευση (όπου οι πράκτορες πρέπει μέσα σε πεπερασμένο χρόνο να συναντηθούν ή να αναφέρουν ότι η συνάντηση είναι αδύνατη) ή χωρίς ανίχνευση.

Αποδεικνύεται ότι η συνάντηση είναι δυνατή μόνο στην περίπτωση που η τιμές των k ή n είναι γνωστές. Αφού είναι πάντα δυνατός ο υπολογισμός του k με δεδομένο το n ή του n με δεδομένο το k κάνοντας έναν κύκλο το δακτύλιο (δηλαδή n επιπλέον βήματα) παρουσιάζουμε τους αλγόριθμούς μας θεωρώντας γνωστό το k. Με αυτές τις συνθήκες, η συνάντηση είναι εφικτή αν και μόνο αν η ακολουθία των αποστάσεων μεταξύ των σημαδιών δεν είναι περιοδική.

Παρουσιάζουμε τρεις αλγόριθμους που λύνουν το πρόβλημα της συνάντησης με ανίχνευση όταν το k είναι γνωστό. Οι πράκτορες χρησιμοποιούν ένα μη-μετακινήσιμο σημάδι ο καθένας. Όταν η ακολουθία των αποστάσεων $\mathcal{S}$ μεταξύ των σημαδιών δεν είναι περιοδική, οι αλγόριθμοι οδηγούν τους πράκτορες σε συνάντηση. Αν η ακολουθία $\mathcal{S}$ είναι περιοδική (και συνεπώς η συνάντηση είναι αδύνατη) τότε οι πράκτορες που εκτελούν αυτούς τους αλγόριθμους σταματούν. Παρουσιάζουμε

Πίνακας 5.1: Το πρόβλημα της Συνάντησης με $k \geq 2$ πράκτορες σε έναν δακτύλιο μεγέθους n.

Αλγόριθμος	Χρόνος	Μνήμη
20	$O(n)$	$O(k \log n)$
21	$O(kn)$	$O(\log n)$
22	$O(\frac{n \log n}{\log \log n})$	$O(k \log \log n)$
23	$O(n)$	$O(\log n)$
24	$O(n)$	$O(\log k)$
25	$O(n \log k)$	$O(\log k)$

επίσης τρεις αλγόριθμους που λύνουν το πρόβλημα της συνάντησης όταν τα k και n ικανοποιούν κάποιους περιορισμούς (όπως το να είναι πρώτοι αριθμοί). Η χωρική και χρονική πολυπλοκότητα αυτών των αλγόριθμων φαίνεται στον Πίνακα 5.1.

Αξίζει να αναφέρουμε ότι οποιοσδήποτε αλγόριθμος που λύνει το πρόβλημα για $k > 2$ πράκτορες χρειάζεται τουλάχιστον $\Omega(\log \log n + \log k)$ μνήμη ([Flocchini et al., 2004]).

5.1.1 Μη-επιλυσιμότητα του προβλήματος της συνάντησης

Όπως φαίνεται από το παρακάτω θεώρημα, η γνώση του k ή του n είναι αναγκαία για την επίλυση του προβλήματος της συνάντησης πανομοιότυπων πρακτόρων σε έναν ανώνυμο και συγχρονισμένο δακτύλιο.

Θεώρημα 5.1 ([Barriere et al., 2007]) Δεν υπάρχει αλγόριθμος που να λύνει το πρόβλημα της συνάντησης πρακτόρων σε δακτύλιο αν ο αριθμός των κόμβων n του δακτυλίου και ο αριθμός k των πρακτόρων είναι άγνωστοι. ■

Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στο άρθρο [Barriere et al., 2007][Ενότητα 4, Θεώρημα 2] για τις λεπτομέρειες της απόδειξης. Ακόμη και αν τα k ή n είναι γνωστά μπορεί να μην είναι δυνατή η επίλυση του προβλήματος. Θυμίζουμε ότι δύο πράκτορες δεν μπορούν να συναντηθούν σε έναν δακτύλιο με άρτιο πλήθος κόμβων n, όταν βρίσκονται σε απόσταση $n/2$ ο ένας από τον άλλον. Έστω $\mathcal{S} = d_1, \ldots, d_k$ η ακολουθία των αποστάσεων μεταξύ των σημαδιών θεωρώντας ότι τα σημάδια έχουν τοποθετηθεί από τους πράκτορες στις αρχικές τους θέσεις. Μια ακολουθία είναι *περιοδική* αν αποτελείται από $m > 1$ επαναλήψεις μιας υποακολουθίας από $j \geq 1$ αποστάσεις, όπου τα m, j διαιρούν το k. Είναι δυνατόν να αποδειχθεί το παρακάτω αποτέλεσμα.

Θεώρημα 5.2 ([Flocchini et al., 2004]) Αν το k είναι γνωστό, η συνάντηση είναι δυνατή αν και μόνο αν η ακολουθία $\mathcal{S}$ των αποστάσεων μεταξύ των σημαδιών δεν είναι περιοδική. ■

5.1.2 Συνάντηση με ανίχνευση

Με δεδομένα τα αποτελέσματα της ενότητας 5.1.1, για να επιλυθεί το πρόβλημα πρέπει η ακολουθία $\mathcal{S}$ των αποστάσεων μεταξύ των σημαδιών να μην είναι περιοδική. Παρουσιάζουμε τώρα αλγόριθμους που επιτυγχάνουν τη συνάντηση σε έναν προσανατολισμένο δακτύλιο.

Ας θεωρήσουμε ότι κάθε πράκτορας έχει $O(k \log n)$ μνήμη. Οι πράκτορες γνωρίζουν το k αλλά όχι αναγκαστικά το n.

Αλγόριθμος 20 (Συνάντηση k πρακτόρων σε προσανατολισμένο δακτύλιο όταν οι πράκτορες γνωρίζουν το k)

1: Τοποθέτησε το σημάδι στον κόμβο εκκίνησης.
2: Κινήσου δεξιόστροφα.
3: Υπολόγισε τις k αποστάσεις μεταξύ των σημαδιών $d_1, \ldots, d_k$.
4: **Αν** η ακολουθία $\mathcal{S} = d_1, \ldots, d_k$ είναι περιοδική, τότε σταμάτα. /* Η συνάντηση είναι αδύνατη. */
5: Θέσε *forward* = $d_1, \ldots, d_k$ και *reverse* = $d_k, \ldots, d_1$
6: Έστω *lexi(someSequence)* η λεξικογραφικά μέγιστη κυκλική μετάθεση της ακολουθίας *someSequence*.
7: Θέσε *forward = lexi(forward)* και *reverse = lexi(reverse)*.
8: **Αν** οι ακολουθίες *forward* και *reverse* διαφέρουν, τότε
i) βρες ποια από αυτές τις ακολουθίες είναι λεξικογραφικά μέγιστη και πήγαινε στον κόμβο που ξεκινά αυτή η ακολουθία.
ii) αλλιώς MA_i και MA_j οι πράκτορες που βρίσκονται στην αρχή των ακολουθιών *forward* και *reverse* αντίστοιχα.
9: **Αν** MA_i και MA_j πρόκειται για τον ίδιο πράκτορα, τότε πήγαινε στον κόμβο που βρίσκεται ο MA_i.
10: **Αν** MA_i και MA_j είναι διαφορετικοί πράκτορες, τότε κοίταξε τα μονοπάτια μεταξύ των MA_i και MA_j.
i) Αν μόνο ένα από τα μονοπάτια έχει περιττό αριθμό κόμβων, τότε πήγαινε στον μεσαίο κόμβο αυτού του μονοπατιού.
ii) Αν και τα δύο μονοπάτια έχουν περιττό αριθμό κόμβων τότε a) αν τα μονοπάτια διαφέρουν σε μήκος, πήγαινε στον μεσαίο κόμβο του μικρότερου μονοπατιού, b) αλλιώς συνέκρινε τις ακολουθίες των αποστάσεων για τα δύο μονοπάτια και πήγαινε στον μεσαίο κόμβο του λεξικογραφικά μέγιστου μονοπατιού.
iii) Αν και τα δύο μονοπάτια έχουν άρτιο αριθμό κόμβων τότε κινήσου δεξιόστροφα και αν βρίσκεσαι στο λεξικογραφικά ελάχιστο μονοπάτι σταμάτα στο άκρο του.

Θεώρημα 5.3 ([Flocchini et al., 2004]) Αν κάθε πράκτορας έχει μνήμη $O(k \log n)$, τότε ο Αλγόριθμος 20 λύνει το πρόβλημα της συνάντησης σε χρόνο $O(n)$. ■

Αν οι πράκτορες έχουν μνήμη $O(\log n)$, τότε το πρόβλημα παραμένει επιλύσιμο.

Θεωρήστε τον Αλγόριθμο 21. Σε κάθε γύρο, ένας πράκτορας μπορεί να είναι ανενεργός και έτσι να περιμένει για το υπόλοιπο του αλγόριθμου στον αρχικό του κόμβο. Αφού οι πράκτορες συμφωνούν στον προσανατολισμό και κινούνται προς την ίδια κατεύθυνση, ένας ενεργός πράκτορας μπορεί να αναγνωρίσει όλους τους ανενεργούς πράκτορες οι οποίοι έχουν σταματήσει. Όταν η μνήμη περιορίζεται σε $O(\log n)$, ένας πράκτορας χρειάζεται περισσότερες από μία περιστροφή το δακτυλίου για να μάθει αν η ακολουθία $\mathcal{S}$ δεν είναι περιοδική.

Αλγόριθμος 21 (Συνάντηση πολλών πρακτόρων με $O(\log n)$ μνήμη σε προσανατολισμένο δακτύλιο)

1: Τοποθέτησε το σημάδι στον κόμβο εκκίνησης.
2: Θέσε $c = 1$. /* Ο αριθμός του τρέχοντος γύρου. */
3: Θέσε $active = 1$. /* Δείχνει αν ο πράκτορας είναι ενεργός. */
4: Θέσε $inactive = 0$. /* Μετρά τον αριθμό των ανενεργών πρακτόρων. */
5: Κινήσου δεξιόστροφα.
6: Αύξησε τη μεταβλητή $inactive$ κάθε φορά που συναντάς έναν ανενεργό πράκτορα.
7: Υπολόγισε την απόσταση από το c-οστό σημάδι, d_c, δηλαδή, αν $c = 1$, υπολόγισε την απόσταση από το πρώτο σημάδι και αν $c = 2$, από το δεύτερο σημάδι, κλπ.
8: Συνέχισε να κινείσαι και συνέκρινε την απόσταση d_c με κάθε απόσταση μεταξύ του πρώτου και τελευταίου από τα $c + 1 \leq k$ διαδοχικά σημάδια.
9: **Αν** βλέπεις μια απόσταση d_i τέτοια ώστε $d_i > d_c$, τότε συνέχισε στην ίδια κατεύθυνση και γίνε ανενεργός (θέσε $active = 0$), όταν φτάσεις τον κόμβο από τον οποίον ξεκίνησες.
10: **Αν** δεν βλέπεις μια απόσταση d_i τέτοια ώστε $d_i > d_c$, τότε να παραμείνεις ενεργός όταν επιστρέψεις στον κόμβο εκκίνησης.
11: **Αν** είσαι ο μόνος πράκτορας που παραμένει ενεργός (δηλαδή $inactive = k - 1$), τότε κάνε έναν κύκλο και κανόνισε τη συνάντηση.
12: **Αν** $c = k - 1$ και $inactive < k - 1$, τότε σταμάτα. /* Όλες οι αποστάσεις είναι ίσες και συνεπώς η συνάντηση είναι αδύνατη. */
13: **Αλλιώς** θέσε $c = c + 1$ και $inactive = 0$.
14: **Επανέλαβε** από το βήμα 5.

Θεώρημα 5.4 ([Flocchini et al., 2004]) Αν κάθε πράκτορας έχει μνήμη $O(\log n)$, τότε ο Αλγόριθμος 21 λύνει το πρόβλημα της συνάντησης σε χρόνο $O(kn)$. ■

Οι Αλγόριθμοι 20 και 21 λύνουν το πρόβλημα όταν κάθε πράκτορας έχει $O(k \log n)$ και $O(\log n)$ μνήμη αντίστοιχα. Όπως φαίνεται στον Αλγόριθμο 22, είναι επίσης δυνατή η επίλυση του προβλήματος όταν κάθε πράκτορας έχει μνήμη $O(k \log \log n)$. Έστω $p_1, \ldots, p_r$ η αύξουσα ακολουθία από r πρώτους αριθμούς έτσι ώστε $\prod_{i=1}^{r} p_i > n$. Όπως προηγουμένως, ένας ενεργός πράκτορας πρέπει να μπορεί να αναγνωρίσει αν ένας άλλος πράκτορας είναι ενεργός.

Αλγόριθμος 22 (Συνάντηση πολλών πρακτόρων με $O(k \log\log n)$ μνήμη σε προσανατολισμένο δακτύλιο)

1: Τοποθέτησε το σημάδι στον κόμβο εκκίνησης.
2: Θέσε active = 1.
3: Έστω p_r ο μικρότερος πρώτος αριθμός έτσι ώστε $\prod_{i=1}^{r} p_i > n$.
4: Θέσε $p_i = p_1 = 2$.
5: Έστω $\alpha = k$, το πλήθος των ενεργών πρακτόρων.
6: Κινήσου δεξιόστροφα και υπολόγισε τις αποστάσεις mod p_i μεταξύ των σημαδιών των α ενεργών πρακτόρων, δηλαδή, $d_1, ..., d_\alpha \bmod p_i$.
7: Έστω *forward* = $d_1, ..., d_\alpha \bmod p_i$.
8: Έστω *reverse* = $d_\alpha, \ldots, d_1 \bmod p_i$.
9: **Αν** η ακολουθία *forward* είναι περιοδική, δηλαδή, *forward* = $(d_1, ..., d_{\alpha/a})^a \bmod p_i$, τότε
i) αν βρίσκεσαι στην αρχή της ακολουθίας $(d_1, ..., d_{\alpha/a})$, τότε παρέμεινε ενεργός.
ii) αλλιώς θέσε active = 0.
iii) αν $p_i < p_r$, τότε a) θέσε $p_i = p_{i+1}$, $\alpha = a$, και επανέλαβε από το βήμα 6. b) αλλιώς σταμάτα αφού η συνάντηση είναι αδύνατη.
10: **Αν** η ακολουθία *forward* δεν είναι περιοδική, τότε έστω *lexi(someSequence)* η λεξικογραφικά μέγιστη κυκλική μετάθεση της ακολουθίας *someSequence*.
11: Εκτέλεσε τα βήματα 7 έως 10 του Αλγόριθμου 20.

Πριν αποδείξουμε το βασικό αποτέλεσμα του Θεωρήματος 5.6 δείχνουμε το παρακάτω λήμμα.

Λήμμα 5.5 Θεωρήστε κάποιον πρώτο αριθμό p τέτοι ώστε $2 < p \leq r$. Υποθέστε ότι για όλους τους πρώτους $p_i < p$, η ακολουθία των αποστάσεων mod p_i μεταξύ των α_i ενεργών πρακτόρων είναι περιοδική έτσι ώστε $d_1, \ldots, d_{\alpha_i} \equiv (d_1, \ldots, d_{\alpha_i/a})^a \bmod p_i$ και $a \mid \alpha_i$. Έστω ότι ο πρώτος πράκτορας σε κάθε εμφάνιση της υποακολουθίας $d_1, \ldots, d_{\alpha_i/a}$ παραμένει ενεργός ενώ οι υπόλοιποι πράκτορες γίνονται ανενεργοί. Αν η ακολουθία των αποστάσεων mod p μεταξύ των a ενεργών πρακτόρων είναι περιοδική, τότε οι αρχικές αποστάσεις μεταξύ των σημαδιών μπορούν να χωριστούν σε $t \mid k$ υποακολουθίες ίσου μήκους με αθροίσματα $\sigma_1, \sigma_2, \ldots, \sigma_t$ τα οποία είναι ίδια modulo όλους τους πρώτους αριθμούς $p_1, p_2, \ldots, p$. ■

Θεώρημα 5.6 ([Flocchini et al., 2004]) Αν κάθε πράκτορας έχει μνήμη $O(k \log\log n)$, τότε ο Αλγόριθμος 22 λύνει το πρόβλημα της συνάντησης σε χρόνο $O\left(\frac{n \log n}{\log\log n}\right)$.

Απόδειξη. Ο Αλγόριθμος 22 λύνει το πρόβλημα της συνάντησης αν οι πράκτορες που τον εκτελούν σταματούν όταν η συνάντηση είναι αδύνατη και συναντιούνται όταν η συνάντηση είναι δυνατή. Ας υποθέσουμε ότι για όλους τους αριθμούς $p_i \leq p_r$, η ακολουθία των αποστάσεων mod p_i μεταξύ των ενεργών πρακτόρων είναι περιοδική. Τότε οι πράκτορες που εκτελούν τον αλγόριθμο σταματούν

στο βήμα 9 και αναφέρουν ότι η συνάντηση είναι αδύνατη. Από το Λήμμα 5.5 προκύπτει ότι στον τελευταίο γύρο του Αλγόριθμου 22, όπου $p = p_r$, οι αρχικές αποστάσεις μεταξύ των σημαδιών μπορούν να χωριστούν σε $a_r \mid k$ υποακολουθίες ίσου μήκους με αθροίσματα $\sigma_i^r, i = 1, \ldots, a_r$, τα οποία είναι όμοια mod p για όλους τους $p \leq p_r$.

Από το Κινέζικο Θεώρημα Υπολοίπων προκύπτει ότι τα αθροίσματα $\sigma_i^r, i = 1, \ldots, a_r$, είναι ίδια mod $\prod_{i=1}^{r} p_i$. Αφού $\prod_{i=1}^{r} p_i > n$, τότε οι αρχικές αποστάσεις μπορούν να χωριστούν $a_r \mid k$ σε ίδιου μήκους υποακολουθίες με αθροίσματα σ_i^r τα οποία είναι ίσα για όλα τα $i, i = 1, \ldots, a_r$. Συνεπώς ισχύει ότι $n = a_r \sigma_i^r$ για κάθε i και άρα $a_r \mid n$. Αφού $\gcd(k, n) = g > 1$, η ακολουθία $\mathcal{S}$ είναι περιοδική, και ο Αλγόριθμος 22 σωστά σταματά τους πράκτορες στο βήμα 9.

Ο Αλγόριθμος 22 πρέπει επίσης να εγγυάται τη συνάντηση όταν αυτή είναι δυνατή, δηλαδή όταν η ακολουθία $\mathcal{S}$ δεν είναι περιοδική. Έστω ότι αυτό δεν συμβαίνει, δηλαδή η ακολουθία $\mathcal{S}$ δεν είναι περιοδική και παρόλαυτά οι πράκτορες που εκτελούν τον αλγόριθμο δεν συναντιούνται. Αυτό σημαίνει ότι για όλα τα $p_i \leq p_r$, οι ακολουθίες που υπολογίστηκαν στο βήμα 7 του αλγόριθμου είναι περιοδικές και συνεπώς θα εκτελεστούν όλοι οι γύροι του αλγόριθμου. Στον τελευταίο γύρο, όπου $p_i = p_r$, ο Αλγόριθμος 22 θα σταματήσει τους πράκτορες με την ένδειξη ότι η συνάντηση είναι αδύνατη. Όμως από το Κινέζικο Θεώρημα των Υπολοίπων, προκύπτει ότι η ακολουθία $\mathcal{S}$ είναι περιοδική το οποίο είναι άτοπο. Άρα ο Αλγόριθμος 22 λύνει το πρόβλημα της συνάντησης.

Αν κάθε πράκτορας έχει μνήμη $O(k \log \log n)$, τότε ο Αλγόριθμος 22 οδηγεί τους πράκτορες σε συνάντηση όταν αυτή είναι δυνατή και τους σταματά όταν είναι αδύνατη. Στην χειρότερη περίπτωση η συνάντηση είναι αδύνατη και οι πράκτορες πρέπει να ολοκληρώσουν όλους τους r γύρους του αλγορίθμου, όπου r είναι ο μικρότερος από τους πρώτους αριθμούς έτσι ώστε $\prod_{i=1}^{r} p_i > n$. Κάθε ένας από τους r γύρους διαρκεί n βήματα και συνεπώς η χρονική πολυπλοκότητα είναι $O(rn)$. Στο άρθρο [Kranakis et al., 2003] αποδεικνύεται ότι $r \, \epsilon \, O(\frac{\log n}{\log \log n})$, και έτσι η χρονική πολυπλοκότητα του Αλγόριθμου 22 είναι $O\left(\frac{n \log n}{\log \log n}\right)$. ■

5.1.3 Συνάντηση χωρίς ανίχνευση με ειδικές συνθήκες

Στην προηγούμενη ενότητα παρουσιάσαμε αλγόριθμους για τη συνάντηση με ανίχνευση με διαφορετικά ισοζύγια χρόνου/μνήμης. Κάτω από συγκεκριμένες συνθήκες είναι δυνατόν να βελτιώσουμε αυτά τα ισοζύγια όταν δεν υπάρχει ανάγκη ελέγχου των περιπτώσεων που η συνάντηση είναι αδύνατη. Μια τέτοια συνθήκη είναι για παράδειγμα όταν το n είναι πρώτος αριθμός. Ο διαχειριστής ενός δικτύου μπορεί να έχει την ευχέρεια να επιλέξει τις τιμές των k και n έτσι ώστε να ικανοποιείται αυτή η συνθήκη.

Αν ισχύει $\gcd(k', n) = 1, \forall k' \leq k$, για παράδειγμα αν ο n είναι πρώτος ή είναι το γινόμενο δύο πρώτων μεγαλύτερο από k, τότε η ακολουθία $\mathcal{S}_i$ δεν είναι περιοδική για οποιοδήποτε i. Ο Αλγόριθμος 23 λειτουργεί σε έναν προσανατολισμένο δακτύλιο. Ένα *active* σημάδι βρίσκεται σε κόμβο που δεν είναι κατειλημμένος ενώ ένα *inactive* σημάδι βρίσκεται σε κόμβο που υπάρχουν κάποιος πράκτορας.

Αλγόριθμος 23 (Συνάντηση k πρακτόρων σε προσανατολισμένο δακτύλιο όταν $\gcd(k', n) = 1$, $\forall k' \leq k$)

1: Τοποθέτησε το σημάδι στον κόμβο εκκίνησης.
2: Θέσε $active = 1$.
3: Θέσε $count = 0$.
4: Κινήσου δεξιόστροφα.
5: Υπολόγισε τις αποστάσεις μεταξύ των τριών επόμενων ενεργών σημαδιών, δηλαδή, d_1, d_2, d_3, και αύξησε τη μεταβλητή $count$ για κάθε ανενεργό σημάδι που συναντάς.
6: **Αν** $count = k - 1$, κανόνισε τη συνάντηση. (Μόνο ενεργοί πράκτορες απομένουν.)
7: **Αν** $d_2 > d_1$ και $d_2 \geq d_3$, τότε παρέμεινε ενεργός.
8: **Αλλιώς** γίνε ανενεργός, δηλαδή θέσε $active = 0$, συνέχισε να κινείσαι προς την ίδια κατεύθυνση μέχρι τον κόμβο εκκίνησης και περίμενε εκεί.
9: **Επανέλαβε** από το βήμα 3.

Θεώρημα 5.7 ([Flocchini et al., 2004]) Όταν οι πράκτορες συμφωνούν ως προς τον προσανατολισμό του δακτυλίου και $\gcd(k', n) = 1$, για κάθε $k' \leq k$, τότε ο Αλγόριθμος 23 λύνει το πρόβλημα της συνάντησης χρησιμοποιώντας $O(\log n)$ μνήμη σε $O(n)$ χρόνο. ■

Όταν το k είναι πρώτος, ο δακτύλιος είναι προσανατολισμένος, και ισχύει ότι $\gcd(k', n) = 1$, $\forall k' \leq k$, τότε ένας λίγο διαφορετικός αλγόριθμος από τον Αλγόριθμο 23 λύνει το πρόβλημα της συνάντησης με $O(\log k)$ μνήμη σε χρόνο $O(n)$.

Θεώρημα 5.8 ([Flocchini et al., 2004]) Όταν οι πράκτορες συμφωνούν ως προς τον προσανατολισμό του δακτυλίου, το k είναι πρώτος, και ισχύει ότι $\gcd(k', n) = 1$, $\forall k' \leq k$, τότε ο Αλγόριθμος 24 λύνει το πρόβλημα της συνάντησης με $O(\log k)$ μνήμη σε $O(n)$ χρόνο. ■

Ο Αλγόριθμος 25 λύνει το πρόβλημα όταν $\gcd(k', n) = 1, \forall k' \leq k$, χωρίς την ανάγκη το k να είναι πρώτος.

Θεώρημα 5.9 ([Flocchini et al., 2004]) Όταν οι πράκτορες συμφωνούν ως προς τον προσανατολισμό του δακτυλίου και ισχύει $\gcd(k', n) = 1, \forall k' \leq k$, τότε ο Αλγόριθμος 25 λύνει το πρόβλημα της συνάντησης με $O(\log k)$ μνήμη σε $O(n \log k)$ χρόνο. ■

5.2 Συνάντηση σε ασύγχρονους δακτύλιους

Σε αυτή την ενότητα θα μελετήσουμε το πρόβλημα της συνάντησης σε τοπολογίες ανώνυμων και μη-προσανατολισμένων δακτυλίων χρησιμοποιώντας ένα μοντέλο στο οποίο οι πράκτορες έχουν εξαιρετικά περιορισμένες δυνατότητες: είναι αυτόνομοι και ανώνυμοι, δεν μπορούν να επικοινωνήσουν μεταξύ τους, κινούνται ασύγχρονα στο δίκτυο χρησιμοποιώντας τον ίδιο ντετερμινιστικό αλγόριθμο και δεν έχουν μνήμη. Αυτό το μοντέλο προτάθηκε στο άρθρο [Prencipe, 2001] και χρησιμοποιήθηκε αρχικά για την συνάντηση πρακτόρων στο επίπεδο. Αργότερα χρησιμοποιήθηκε

Αλγόριθμος 24 (Συνάντηση k πρακτόρων σε προσανατολισμένο δακτύλιο όταν $\gcd(k', n) = 1$, $\forall k' \leq k$) και το k είναι πρώτος αριθμός)

1: Τοποθέτησε το σημάδι στον κόμβο εκκίνησης.
2: Θέσε active = 1.
3: Κινήσου δεξιόστροφα.
4: Εκτέλεσε τον γύρο 1 του Αλγόριθμου 23 υπολογίζοντας τις αποστάσεις μεταξύ των σημαδιών mod k.
/* Όλοι οι πράκτορες θα επιστρέψουν στους αντίστοιχους κόμβους εκκίνησης και όσοι έχουν γίνει ανενεργοί έχουν θέσει active = 0. */
5: /* Εκτέλεσε τον Αλγόριθμο 23 σαν να βρίσκεσαι σε δακτύλιο μεγέθους k Οι αποστάσεις που ενδιαφέρουν είναι το πλήθος των σημαδιών σε κόμβους με πράκτορες μεταξύ κάθε ζεύγους σημαδιών σε άδειους κόμβους. */
6: Υπολόγισε το πλήθος των σημαδιών σε κόμβους με πράκτορες, δηλαδή των σημαδιών που βρίσκονται σε κόμβους με ανενεργούς πράκτορες, τα οποία συναντάς στο δρόμο με τα επόμενα τρία σημάδια σε άδειους κόμβους, δηλαδή m_1, m_2, m_3.
7: **Αν** $m_1 = k - 1$, κανόνισε τη συνάντηση.
/* Υπάρχει μόνο ένας ενεργός πράκτορας */
8: **Αν** $m_2 > m_1$ και $m_2 \geq m_3$, τότε παρέμεινε ενεργός.
9: **Αλλιώς** γίνε ανενεργός, δηλαδή θέσε $active = 0$, συνέχισε να κινείσαι προς την ίδια κατεύθυνση μέχρι τον κόμβο εκκίνησης και περίμενε εκεί.
10: **Επανέλαβε** από το βήμα 5.

για την επίλυση προβλημάτων σε δίκτυα διαφορετικών τοπολογιών. Τα αποτελέσματα αυτού του κεφαλαίου δημοσιεύτηκαν στο άρθρο [Klasing et al., 2008b].

5.2.1 Τυπική περιγραφή του μοντέλου

Θεωρούμε ένα δακτύλιο n κόμβων. Ο δακτύλιος είναι *μη-προσανατολισμένος*, δηλαδή δεν είναι καθορισμένη η δεξιόστροφη και η αριστερόστροφη κατεύθυνση και *ανώνυμος*, δηλαδή δεν υπάρχουν ετικέττες στους κόμβους του (ή ισοδύναμα, όλοι οι κόμβοι του έχουν την ίδια ετικέττα). Σε κάποιους από τους κόμβους του δακτυλίου βρίσκονται πράκτορες, το πολύ ένας πράκτορας σε κάθε κόμβο. Ο σκοπός των πρακτόρων είναι να συγκεντρωθούν σε έναν κόμβο του δακτυλίου και να παραμείνουν εκεί. Οι πράκτορες λειτουργούν σε επαναλαμβανόμενους κύκλους που αποτελούνται από τις φάσεις *Look-Compute-Move*. Σε κάθε κύκλο ο πράκτορας παίρνει μια *εικόνα* ή *στιγμιότυπο* του *σχηματισμού* εκείνη τη στιγμή (*Look*), έπειτα, με βάση αυτήν την εικόνα, κάνει υπολογισμούς και αποφασίζει αν θα μείνει στον ίδιο κόμβο ή αν θα κινηθεί σε έναν από τους γειτονικούς του κόμβους (*Compute*), και τέλος εάν αποφάσισε να κινηθεί σε κάποιο γειτονικό κόμβο, τότε κινείται στιγμιαία εκεί (*Move*). Οι φάσεις των κύκλων εκτελούνται *ασύγχρονα* για κάθε πράκτορα. Αυτό

Αλγόριθμος 25 (Συνάντηση k πρακτόρων σε προσανατολισμένο δακτύλιο όταν $\gcd(k', n) = 1$, $\forall k' \leq k$ μέσα σε $O(n \log k)$ χρόνο)

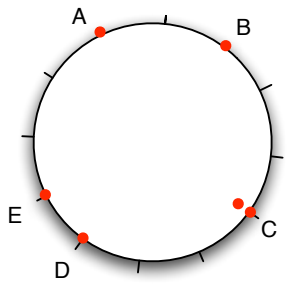
1: Τοποθέτησε το σημάδι στον κόμβο εκκίνησης.
2: Θέσε active = 1 και count = 0.
3: Κινήσου δεξιόστροφα.
4: Υπολόγισε τις αποστάσεις mod k για τα τρία επόμενα ενεργά σημάδια, δηλαδή d_1, d_2, d_3 mod k, και αύξησε τη μεταβλητή $count$ για κάθε ανενεργό σημάδι που συναντάς.
5: **Αν** ισχύει $count = k - 1$, κανόνισε τη συνάντηση.
/* Μόνο ενεργοί πράκτορες απομένουν. */
6: **Αν** $d_2 > d_1$ mod k και $d_2 \geq d_3$ mod k, τότε παρέμεινε ενεργός.
7: **Αλλιώς** γίνε ανενεργός, δηλαδή θέσε $active = 0$, και περίμενε.
8: **Επανέλαβε** από το βήμα 4.

σημαίνει ότι ο χρόνος μεταξύ των φάσεων Look, Compute, και Move είναι πεπερασμένος αλλά όχι γνωστός. Ο μόνος περιορισμός είναι ότι οι κινήσεις είναι στιγμιαίες, και συνεπώς κάθε πράκτορας που εκτελεί τη φάση Look βλέπει όλους τους άλλους πράκτορες να βρίσκονται σε κόμβους του δακτυλίου και όχι σε ακμές.

Λόγω του ασύγχρονου μοντέλου μπορεί να συμβεί το εξής: Ένας πράκτορας R παίρνει ένα στιγμιότυπο του σχηματισμού (εκτελώντας τη φάση Look) σε μια χρονική στιγμή t, στο οποίο απεικονίζονται οι πράκτορες σε συγκεκριμένους κόμβους. Έπειτα, τη χρονική στιγμή $t' > t$, ο πράκτορας R (εκτελώντας τη φάση Compute) αποφασίζει τον κόμβο στον οποίο θα κινηθεί. Τέλος, τη χρονική στιγμή $t'' > t'$ ο R (εκτελώντας τη φάση Move) κινείται στον κόμβο που αποφάσισε. Κατά τη διάρκεια αυτών των φάσεων που εκτελεί ο R κάποιοι από τους υπόλοιπους πράκτορες μπορεί να έχουν κινηθεί σε διαφορετικούς κόμβους από εκείνους στους οποίους αποτυπώνονται στο στιγμιότυπο του R.

Αυτό σημαίνει ότι οι πράκτορες μπορεί να αποφασίσουν να κινηθούν βασισμένοι σε στιγμιότυπα πολύ παλιών σχηματισμών, γεγονός που κάνει την επίλυση του προβλήματος της συνάντησης αρκετά δύσκολη. Εδώ πρέπει να τονίσουμε ότι οι πράκτορες είναι *oblivious*, δηλαδή δεν μπορούν να κρατήσουν στη μνήμη τους παλιά στιγμιότυπα. Έτσι λοιπόν ο κόμβος προς τον οποίο θα κινηθεί ένας πράκτορας αποφασίζεται από τον πράκτορα κατά τη διάρκεια της φάσης Compute και εξαρτάται αποκλειστικά και μόνο από το προηγούμενο στιγμιότυπο που πήρε ο πράκτορας στον ίδιο κύκλο κατά την εκτέλεση της φάσης Look. Οι πράκτορες είναι ανώνυμοι (δηλαδή δεν έχουν διαφορετικές ταυτότητες) και πανομοιότυποι εκτελώντας τον ίδιο ντετερμινιστικό αλγόριθμο. Δεν μπορούν να αφήσουν οποιοδήποτε σημάδι στους κόμβους του δακτυλίου που επισκέπτονται και δεν μπορούν να επικοινωνήσουν μεταξύ τους.

Οι πράκτορες έχουν την ικανότητα, κατά τη διάρκεια της φάσης Look, να διαπιστώσουν αν υπάρχει ένας ή περισσότεροι πράκτορες σε κάθε κόμβο. Εδώ πρέπει να τονίσουμε ότι ένας πράκτορας μπορεί να διαπιστώσει για οποιονδήποτε κόμβο v τα εξής: αν δεν υπάρχει πράκτορας, αν υπάρχει μόνο ένας πράκτορας ή αν υπάρχουν περισσότεροι πράκτορες από έναν στον v. Δηλαδή



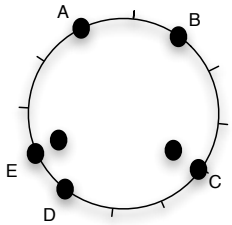
Σχήμα 5.1: Ένας σχηματισμός πρακτόρων με μία πολλαπλότητα. Το ζεύγος ακολουθιών που αναπαριστά αυτόν τον σχηματισμό, ξεκινώντας από τον πράκτορα A είναι $((2,3,3,1,3),(5))$. Αντίστοιχα, η αναπαράσταση του σχηματισμού αν ξεκινήσουμε από το B είναι $((3,3,1,3,2),(3))$, ενώ αν ξεκινήσουμε από το C είναι $((3,1,3,2,3),(0))$. Η εικόνα που έχει ο πράκτορας A για αυτόν τον σχηματισμό είναι $\{((2,3,3,1,3),(5)),((3,1,3,3,2),(7))\}$.

ο πράκτορας δεν μπορεί να διακρίνει μεταξύ των περιπτώσεων να βρίσκονται a ή b πράκτορες σε έναν κόμβο, αν $a,b>1$. Με άλλα λόγια ο πράκτορας δεν έχει την ικανότητα να μετρήσει το πλήθος των πρακτόρων που βρίσκονται σε κάποιον κόμβο αν αυτό το πλήθος είναι μεγαλύτερο του ενός πράκτορα.

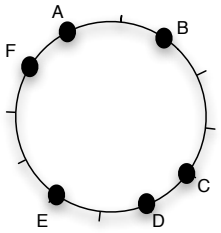
Κατά τη διάρκεια οποιασδήποτε μεθόδου που οδηγεί τους πράκτορες σε συνάντηση, οι πράκτορες κινούνται και σε οποιαδήποτε στιγμή καταλαμβάνουν κόμβους του δακτυλίου δημιουργώντας έναν *σχηματισμό*. Ένας σχηματισμός συμβολίζεται με το ζεύγος ακολουθιών $((a_1,\ldots,a_r),(b_1,\ldots,b_s))$, όπου οι ακέραιοι a_i and b_j έχουν την εξής ερμηνεία:

Ας διαλέξουμε κάποιο κόμβο συμβολίζοντάς τον u_1 στον οποίο βρίσκεται τουλάχιστον ένας πράκτορας και ας θεωρήσουμε τους διαδοχικούς κόμβους $u_1,u_2,u_3,\ldots,u_r$ στους οποίους κόμβους βρίσκεται τουλάχιστον ένας πράκτορας, όπως τους συναντάμε αν διατρέξουμε τον δακτύλιο κατά τη δεξιόστροφη φορά ξεκινώντας από τον u_1[1]. Ο ακέραιος a_i, για $i<r$, αναπαριστά την απόσταση (πλήθος ακμών) στο δακτύλιο μεταξύ των κόμβων u_i και u_{i+1}, ενώ ο ακέραιος a_r αναπαριστά την απόσταση (πλήθος ακμών) στο δακτύλιο μεταξύ των κόμβων u_r και u_1 (στη δεξιόστροφη κατεύθυνση). Κάθε κόμβος μεταξύ των $u_1,u_2,u_3,\ldots,u_r$ στον οποίον βρίσκονται πάνω από ένας πράκτορες καλείται *πολλαπλότητα*. Έστω $u_{v_1},\ldots,u_{v_s}$ οι διαδοχικοί (δεξιόστροφα) κόμβοι-πολλαπλότητες. Ο ακέραιος b_i αναπαριστά την απόσταση (πλήθος ακμών) δεξιόστροφα μεταξύ των κόμβων u_1 και u_{v_i}. Φυσικά είναι προφανές ότι διαφορετική επιλογή αρχικού κόμβου (u_1) προκαλεί διαφορετικά ζεύγη ακολουθιών. Οι αντίστοιχες ακολουθίες διαφορετικών ζευγαριών αναπαριστούν την ίδια τοποθέτηση των πρακτόρων πάνω στο δακτύλιο. Όταν μιλάμε για σχηματισμούς

[1] Σημειώνουμε εδώ ότι η δεξιόστροφη φορά εισάγεται μόνο για χάρη της παρουσίασης και του ορισμού των εννοιών. Οι πράκτορες δεν μπορούν να ορίσουν αυτή την έννοια εφόσον ο δακτύλιος δεν έχει προσανατολισμό.



Σχήμα 5.2: Ένας σχηματισμός πρακτόρων με δύο πολλαπλότητες. Το ζεύγος ακολουθιών που αναπαριστά αυτόν τον σχηματισμό, ξεκινώντας από τον πράκτορα A είναι $((2,3,3,1,3),(5,9))$. Αντίστοιχα, η αναπαράσταση του σχηματισμού αν ξεκινήσουμε από το B είναι $((3,3,1,3,2),(3,7))$, ενώ αν ξεκινήσουμε από το C είναι $((3,1,3,2,3),(0,4))$. Η εικόνα που έχει ο πράκτορας A για αυτόν τον σχηματισμό είναι $\{((2,3,3,1,3),(5,9)),((3,1,3,3,2),(3,7))\}$.

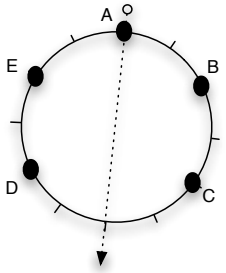


Σχήμα 5.3: Ένας περιοδικός σχηματισμός. Το ζεύγος ακολουθιών που αναπαριστά αυτόν τον σχηματισμό, ξεκινώντας από τον πράκτορα A είναι $(2,3,1,2,3,1)$. Οι εικόνες που έχουν οι πράκτορες A και D κωδικοποιούνται ως εξής: $\{(2,3,1,2,3,1),(1,3,2,1,3,2)\}$. Οι πράκτορες B και E έχουν την ίδια εικόνα για τον σχηματισμό: $\{(3,1,2,3,1,2),(2,1,3,2,1,3)\}$. Οι πράκτορες C και F έχουν επίσης την ίδια εικόνα: $\{(1,2,3,1,2,3),(3,2,1,3,2,1)\}$.

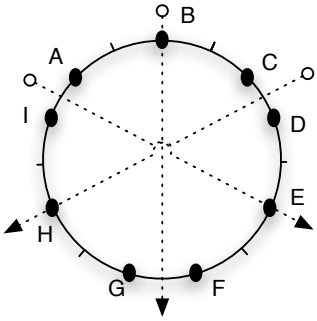
χωρίς πολλαπλότητες θα αναφέρουμε μόνο την πρώτη ακολουθία $(a_1,\ldots,a_r)$. Παραδείγματα σχηματισμών με μία και με δύο πολλαπλότητες φαίνονται στα Σχήματα 5.1 και 5.2 αντίστοιχα.

Ένας σχηματισμός $C=(a_1,\ldots,a_r)$ χωρίς πολλαπλότητες καλείται *περιοδικός* εάν είναι παράθεση τουλάχιστον δύο αντιγράφων κάποιας υποακολουθίας p του C. Ένας περιοδικός σχηματισμός φαίνεται στο Σχήμα 5.3.

Ένας σχηματισμός χωρίς πολλαπλότητες καλείται *συμμετρικός* εάν υπάρχει κάποιος άξονας συμμετρίας στο δακτύλιο, τέτοιος ώστε αυτός ο σχηματισμός να είναι συμμετρικός ως προς αυτόν τον άξονα. Εάν το πλήθος των πρακτόρων είναι περιττός αριθμός και S είναι ένας άξονας συμ-



Σχήμα 5.4: Ένας συμμετρικός σχηματισμός. Το ζεύγος ακολουθιών που αναπαριστά αυτόν τον σχηματισμό, ξεκινώντας από τον πράκτορα A είναι $(2, 2, 4, 2, 2)$. Η εικόνα που έχει ο πράκτορας A αναπαριστάται ως εξής: $\{(2, 2, 4, 2, 2), (2, 2, 4, 2, 2)\}$. Ξεκινώντας από τον πράκτορα B ή C ο σχηματισμός αναπαριστάται αντίστοιχα ως $(2, 4, 2, 2, 2)$ ή $(4, 2, 2, 2, 2)$. Οι πράκτορες B και E έχουν την εικόνα $\{(2, 4, 2, 2, 2), (2, 2, 2, 4, 2)\}$. Οι πράκτορες C και D έχουν την εικόνα $\{(4, 2, 2, 2, 2), (2, 2, 2, 2, 4)\}$.



Σχήμα 5.5: Ένας συμμετρικός και περιοδικός σχηματισμός. Το ζεύγος ακολουθιών που αναπαριστά αυτόν τον σχηματισμό, ξεκινώντας από τον πράκτορα A είναι $(2, 2, 1, 2, 2, 1, 2, 2, 1)$. Ο σχηματισμός αυτός έχει 3 άξονες συμμετρίας. Η εικόνα των πρακτόρων A, C, D, F, G, I είναι $\{(2, 2, 1, 2, 2, 1, 2, 2, 1), (1, 2, 2, 1, 2, 2, 1, 2, 2)\}$. Η εικόνα των πρακτόρων B, E, H είναι $\{(2, 1, 2, 2, 1, 2, 2, 1, 2), (2, 1, 2, 2, 1, 2, 2, 1, 2)\}$.

μετρίας τότε υπάρχει ακριβώς ένας πράκτορας πάνω στον άξονα S. Αυτός ο πράκτορας καλείται *αξονικός* ως προς τον S. Ένας συμμετρικός σχηματισμός φαίνεται στο Σχήμα 5.4.

Παρατηρήστε ότι ένας σχηματισμός μπορεί να ανήκει σε οποιαδήποτε από τις ακόλουθες κατηγορίες: συμμετρικός, περιοδικός, μη-συμμετρικός και μη-περιοδικός, ή συμμετρικός και περιοδικός. Ένας σχηματισμός που είναι περιοδικός και συμμετρικός φαίνεται στο Σχήμα 5.5.

Δύο πράκτορες καλούνται γειτονικοί, αν τουλάχιστον ένα από τα δύο μεταξύ τους διαστήματα στο δακτύλιο δεν περιέχει άλλον πράκτορα. Αυτό το διάστημα μεταξύ δύο γειτονικών πρακτόρων στο οποίο δεν περιέχεται άλλος πράκτορας καλείται *ελεύθερο*.

Θα περιγράψουμε τώρα με περισσότερες λεπτομέρειες, τί ακριβώς αντιλαμβάνεται και πως το αναπαριστά ένας πράκτορας κατά τη διάρκεια μιας φάσης Look. Έστω ένας πράκτορας R σε έναν σχηματισμό που αναπαριστάται με το ζεύγος ακολουθιών $((a_1, \ldots, a_r), (b_1, \ldots, b_s))$, όπου σε αυτήν την αναπαράσταση ο κόμβος που βρίσκεται ο πράκτορας R είναι ο u_1. Η *εικόνα* (*view*) του πράκτορα R είναι ένα σύνολο από δύο ζευγάρια ακολουθιών

$$\{((a_1, \ldots, a_r), (b_1, \ldots, b_s)), ((a_r, a_{r-1}, \ldots, a_1), (n - b_s, \ldots, n - b_1))\}$$

Αν ο κόμβος που βρίσκεται ο R είναι πολλαπλότητα τότε ορίζουμε την εικόνα του R ως

$$\{((a_1, \ldots, a_r), (0, b_2, \ldots, b_s)), ((a_r, a_{r-1}, \ldots, a_1), (0, n - b_s, \ldots, n - b_2))\}$$

Η ανάγκη της αναπαράστασης της εικόνας ενός πράκτορα ως σύνολο της δεξιόστροφης και αριστερόστροφης κωδικοποίησης του σχηματισμού προκύπτει από το γεγονός ότι ο δακτύλιος είναι μη-προσανατολισμένος και συνεπώς ο πράκτορας R δεν μπορεί να ονοματίσει ως δεξιόστροφη κάποια από τις δύο κωδικοποιήσεις. Έτσι λοιπόν κωδικοποιεί την εικόνα με το (μη-διατεταγμένο) σύνολο:

$$\{((a_1, \ldots, a_r), (b_1, \ldots, b_s)), ((a_r, a_{r-1}, \ldots, a_1), (n - b_s, \ldots, n - b_1))\}$$

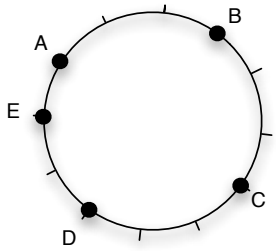
Όπως προηγουμένως, αν δεν υπάρχουν πολλαπλότητες θα παραλείπουμε τη δεύτερη ακολουθία σε κάθε ζεύγος και θα αναπαριστούμε την εικόνα με το σύνολο δύο ακολουθιών: $\{(a_1, \ldots, a_r), (a_r, a_{r-1}, \ldots, a_1)\}$. Για παράδειγμα, σε έναν δακτύλιο 9 κόμβων $1, \ldots, 9$ στον οποίο βρίσκονται τρείς πράκτορες στους κόμβους $1, 2, 4$, η εικόνα του πράκτορα R αν βρίσκεται στον κόμβο 1 αναπαριστάται με το σύνολο $\{(1, 2, 6), (6, 2, 1)\}$. Δείτε επίσης τις αναπαραστάσεις των εικόνων των πρακτόρων στα Σχήματα 5.1, 5.2, 5.3, 5.4 και 5.5. Ένας σχηματισμός χωρίς πολλαπλότητες καλείται *rigid* αν οι εικόνες όλων των πρακτόρων είναι ανά δύο διαφορετικές. Ένας rigid σχηματισμός φαίνεται στο Σχήμα 5.6.

Συνοπτική περιγραφή του μοντέλου

Στη συνέχεια συνοψίζουμε την περιγραφή του μοντέλου. Ένας αριθμός από πανομοιότυπους κινητούς πράκτορες:

- έχει τοποθετηθεί σε ένα *ανώνυμο* και *μη-προσανατολισμένο δακτύλιο* (το πολύ ένας πράκτορας σε κάθε κόμβο),
- λειτουργούν σε *Look-Compute-Move* κύκλους.

Κατά τη διάρκεια ενός κύκλου, ένας πράκτορας:



Σχήμα 5.6: Ένας rigid σχηματισμός. Το ζεύγος ακολουθιών που αναπαριστά αυτόν τον σχηματισμό, ξεκινώντας από τον πράκτορα A είναι $(3,3,3,2,1)$. Η εικόνες των πρακτόρων A, B, C, D, E είναι $\{(3,3,3,2,1),(1,2,3,3,3)\}$, $\{(3,3,2,1,3),(3,1,2,3,3)\}$, $\{(3,2,1,3,3),(3,3,1,2,3)\}$, $\{(2,1,3,3,3),(3,3,3,1,2)\}$ και $\{(1,3,3,3,2),(2,3,3,3,1)\}$ αντίστοιχα.

- παίρνει μια φωτογραφία του περιβάλλοντος (*Look*), έπειτα,
- παίρνει απόφαση να μείνει ακίνητος ή να κινηθεί σε κάποιον από τους γειτονικούς κόμβους (*Compute*),
- και στην τελευταία περίπτωση κινείται στιγμιαία στον κόμβο που επέλεξε (*Move*).

Κανόνες

1. Οι φάσεις των κύκλων εκτελούνται *ασύγχρονα* για κάθε πράκτορα.
2. Οι κινήσεις είναι *στιγμιαίες*.
3. Οι πράκτορες είναι *oblivious*, δηλαδή δεν κρατούν στη μνήμη τους παρατηρήσεις παλαιότερων κύκλων.
4. Οι πράκτορες είναι *ανώνυμοι* και εκτελούν *τον ίδιο ντετερμινιστικό* αλγόριθμο.

Παρατηρήσεις

- Οι αποφάσεις των πρακτόρων μπορεί να βασίζονται σε παρατηρήσεις πολύ παλιών σχηματισμών.
- Ο κόμβος στον οποίο πρόκειται να κινηθεί ο πράκτορας, αποφασίζεται κατά τη διάρκεια της λειτουργίας *Compute* και η απόφαση αυτή βασίζεται αποκλειστικά στην εικόνα του σχηματισμού που είχε ο πράκτορας κατά την προηγούμενη (στον ίδιο κύκλο) φάση *Look*.

Κατηγορίες σχηματισμών

Ορισμός 5.10 Ένας σχηματισμός C χωρίς πολλαπλότητες καλείται *περιοδικός* εάν είναι παράθεση τουλάχιστον δύο αντιγράφων κάποιας υποακολουθίας p της C.

Ορισμός 5.11 Ένας σχηματισμός χωρίς πολλαπλότητες καλείται *συμμετρικός* εάν υπάρχει κάποιος άξονας συμμετρίας στο δακτύλιο, τέτοιος ώστε αυτός ο σχηματισμός να είναι συμμετρικός ως προς αυτόν τον άξονα.

Ορισμός 5.12 Ένας σχηματισμός χωρίς πολλαπλότητες καλείται *rigid* εάν οι εικόνες όλων των ρομπότ είναι διαφορετικές ανά δύο.

Περιγραφή των αποτελεσμάτων

Στις επόμενες ενότητες του κεφαλαίου θα παρουσιάσουμε τα εξής αποτελέσματα:

Για περιττό πλήθος πρακτόρων:

- Η συνάντηση είναι δυνατή αν και μόνο αν ο αρχικός σχηματισμός στο δακτύλιο είναι μη-περιοδικός.
- Δίνουμε έναν αλγόριθμο που οδηγεί έναν οποιονδήποτε μη-περιοδικό σχηματισμό από πράκτορες σε συνάντηση.

Για άρτιο αριθμό από πράκτορες:

- Μπορούμε να αποφασίσουμε για το εάν είναι δυνατή η συνάντηση σε όλες τις περιπτώσεις εκτός από μία περίπτωση συμμετρίας.
- Δίνουμε έναν αλγόριθμο που οδηγεί τους πράκτορες σε συνάντηση σε κάθε τέτοια περίπτωση.

5.2.2 Ιδιότητες των σχηματισμών

Σε αυτήν την ενότητα αποδεικνύουμε κάποια λήμματα σχετικά με τις ιδιότητες των σχηματισμών.

Λήμμα 5.13 Εάν ένας σχηματισμός C χωρίς πολλαπλότητες είναι συμμετρικός τότε ο C είναι non-rigid.

Απόδειξη. Έστω δύο πράκτορες a, b οι οποίοι είναι τοποθετημένοι συμμετρικά πάνω στον δακτύλιο ως προς κάποιον άξονα συμμετρίας, και έχουν εικόνες $\{(a_1, \ldots, a_r), (a_r, a_{r-1}, \ldots, a_1)\}$, και $\{(b_1, \ldots, b_r), (b_r, b_{r-1}, \ldots, b_1)\}$ αντίστοιχα. Έστω χωρίς βλάβη της γενικότητας ότι στην εικόνα του a, η ακολουθία των αποστάσεων $a_+ = (a_1, \ldots, a_r)$ είναι σε δεξιόστροφη κατεύθυνση και στην εικόνα του b, η ακολουθία των αποστάσεων $b_- = (b_r, b_{r-1}, \ldots, b_1)$ είναι στην αριστερόστροφη κατεύθυνση[2].

Έστω a_w η απόσταση στην ακολουθία a_+ για την οποία u_{w+1} είναι ο κόμβος στον οποίο έχει τοποθετηθεί ο πράκτορας b. Τότε η $w-$στή απόσταση στην ακολουθία b_- είναι εκείνη μεταξύ των κόμβων u_2 και u_1 (όπου u_1 είναι ο κόμβος στον οποίο βρίσκεται ο πράκτορας a). Αφού οι πράκτορες a και b είναι τοποθετημένοι συμμετρικά, ισχύει ότι $a_1 = b_r = a_w = b_{r-w+1}$. Επίσης για τον ίδιο λόγο ισχύει $a_r = b_1$. Επιπλέον για κάθε j, εάν η $j-$στή απόσταση στην ακολουθία a_+ βρίσκεται μεταξύ οποιονδήποτε δύο αποστάσεων από τις a_1, a_w, a_r τότε θα πρέπει να εμφανίζεται ως η $j-$στή απόσταση στην ακολουθία b_- και να βρίσκεται μεταξύ των αντίστοιχων δύο αποστάσεων από τις b_r, b_{r-w+1}, b_1. Συνεπώς:

$$(a_1, \ldots, a_w, \ldots, a_r) = (b_r, \ldots, b_{r-w+1}, \ldots, b_1)$$

Αυτό σημαίνει πως οι πράκτορες έχουν την ίδια εικόνα και άρα ο σχηματισμός είναι non-rigid. ■

Λήμμα 5.14 Εάν ένας σχηματισμός C χωρίς πολλαπλότητες είναι περιοδικός τότε ο C είναι non-rigid.

Απόδειξη. Εάν ο σχηματισμός $C = (a_1, \ldots, a_r)$ είναι περιοδικός τότε αποτελείται από παραθέσεις τουλάχιστον δύο αντίγραφων κάποιας υποακολουθίας αποστάσεων $(a_1, \ldots, a_p)$. Έτσι ισχύει:

$$(a_1, \ldots, a_p, \ldots, a_r) = (a_{p+1}, \ldots, a_r, a_1, \ldots, a_p)$$

αφού η δεύτερη ακολουθία είναι μια κυκλική μετάθεση της πρώτης με βάση την επαναλαμβανόμενη υποακολουθία $(a_1, \ldots, a_p)$. Έστω $a_+ = (a_1, \ldots, a_p, \ldots, a_r)$ η ακολουθία των αποστάσεων κατά την δεξιόστροφη φορά στην εικόνα του πράκτορα a (όπου ο πράκτορας a είναι τοποθετημένος στον κόμβο u_1). Τότε η $(a_{p+1}, \ldots, a_r, a_1, \ldots, a_p)$ είναι η ακολουθία των αποστάσεων κατά την δεξιόστροφη φορά στην εικόνα του πράκτορα b (όπου ο πράκτορας b είναι τοποθετημένος στον κόμβο u_{p+1}). Συνεπώς οι πράκτορες a, b έχουν και πάλι την ίδια εικόνα και άρα ο σχηματισμός και σε αυτή την περίπτωση είναι non-rigid. ■

Λήμμα 5.15 Εάν ένας σχηματισμός C χωρίς πολλαπλότητες είναι non-rigid τότε είναι είτε συμμετρικός είτε περιοδικός.

[2] Τονίζουμε ότι η δεξιόστροφη και η αριστερόστροφη κατεύθυνση εισάγονται εδώ μόνο για χάρη της ανάλυσης, καθώς οι πράκτορες δεν έχουν συμφωνήσει στον προσανατολισμό του δακτυλίου.

Απόδειξη. Αφού από την υπόθεση, ο σχηματισμός C είναι non-rigid, σημαίνει ότι υπάρχουν δύο πράκτορες a, b που έχουν την ίδια εικόνα. Έστω χωρίς βλάβη της γενικότητας ότι στην εικόνα του πράκτορα a, η ακολουθία των αποστάσεων $a_+ = (a_1, \ldots, a_r)$ έχει δεξιόστροφη κατεύθυνση και η ακολουθία των αποστάσεων $a_- = (a_r, a_{r-1}, \ldots, a_1)$ έχει αριστερόστροφη κατεύθυνση και ανάλογα οι ακολουθίες για τον πράκτορα b είναι: $b_+ = (b_1, \ldots, b_r)$ και $b_- = (b_r, b_{r-1}, \ldots, b_1)$.

Συνεπώς ισχύει υποχρεωτικά μία από τις εξής δύο περιπτώσεις: (i)$(a_1, \ldots, a_r) = (b_r, b_{r-1}, \ldots, b_1)$ είτε, (ii)$(a_1, \ldots, a_r) = (b_1, \ldots, b_r)$.

- (i) Αν $(a_1, \ldots, a_r) = (b_r, b_{r-1}, \ldots, b_1)$: Έστω a_w η απόσταση στην ακολουθία a_+ όπου u_{w+1} είναι ο κόμβος στον οποίο είναι τοποθετημένος ο πράκτορας b. Συνεπώς ισχύει $a_w = b_r$. Τότε η $w-$στή απόσταση στην ακολουθία b_- είναι η απόσταση μεταξύ των κόμβων u_2 και u_1 (όπου u_1 είναι ο κόμβος στον οποίο είναι τοποθετημένος ο πράκτορας a). Επομένως ισχύει ότι $b_{r-w+1} = a_1$. Έστω a_s η $\lfloor \frac{w+1}{2} \rfloor$-στή απόσταση στην ακολουθία a_+ (η οποία απόσταση βρίσκεται μεταξύ των a_1 και a_w) και έστω a_m η $\lfloor \frac{w+r}{2} \rfloor$-στή απόσταση στην ακολουθία a_+ (η οποία βρίσκεται μεταξύ των a_w και a_r). Έχουμε, $a_1 = b_r = a_w = b_{r-w+1}$ και οι αποστάσεις a_s, a_m εμφανίζονται ως $s-$στή και $m-$στή αντίστοιχα στην ακολουθία b_-. Επίσης από την υπόθεση ισχύει ότι $a_r = b_1$. Επιπλέον για κάθε j, εάν η $j-$στή απόσταση στην ακολουθία a_+ βρίσκεται μεταξύ κάποιων δύο από τις a_1, a_s, a_w, a_m, a_r τότε εμφανίζεται ως η $j-$στή απόσταση στην ακολουθία b_- και βρίσκεται μεταξύ των αντίστοιχων δύο από τις $b_r, b_{r-s+1}, b_{r-w+1}, b_{r-m+1}, b_1$. Συνεπώς ισχύει:

  $(a_1, \ldots, a_s, \ldots, a_w, \ldots, a_m, \ldots, a_r) =$

  $$(b_r, \ldots, b_{r-s+1}, \ldots, b_{r-w+1}, \ldots, b_{r-m+1}, \ldots, b_1).$$

  Άρα ο σχηματισμός είναι συμμετρικός και ο άξονας συμμετρίας περνά από το μέσο της απόστασης a_s ή από τον κόμβο u_{s+1} και από το μέσο της απόστασης a_m ή από τον κόμβο u_{m+1}.

- (ii) Άν $(a_1, \ldots, a_r) = (b_1, \ldots, b_r)$: Ας υποθέσουμε χωρίς βλάβη της γενικότητας ότι πηγαίνοντας δεξιόστροφα από τη θέση που βρίσκεται ο πράκτορας a προς τη θέση που βρίσκεται ο πράκτορας b δεν υπάρχει άλλος πράκτορας c για τον οποίον να ισχύει $c_+ = a_+$ (αν υπάρχει τέτοιος πράκτορας τότε θεωρούμε εκείνον ως πράκτορα b). Έστω a_w η απόσταση στην ακολουθία a_+ όπου u_{w+1} είναι ο κόμβος στον οποίο είναι τοποθετημένος ο πράκτορας b. Τότε πρέπει να ισχύουν $a_{w+1} = b_1 = a_1$, $a_{w+2} = b_2 = a_2$ και αναλόγως για τους υπόλοιπους όρους των δύο ακολουθιών. Συνεπώς ισχύουν $a_{w+i} = b_i = a_i$, $1 \leq i \leq w$ και $a_{mw+i} = b_i = a_i$, όπου $mw + i < r$. Ισχυριζόμαστε και θα αποδείξουμε ότι οι παραπάνω ακολουθίες είναι περιοδικές, και πιο συγκεκριμένα ότι κάθε μια από αυτές αποτελείται από παραθέσεις αντιγράφων της υποακολουθίας $(a_1, \ldots, a_w)$. Εάν r είναι πολλαπλάσιο του w τότε το συμπέρασμα προκύπτει άμεσα. Διαφορετικά, έστω $r = mw + x$, όπου $1 \leq x < w$ (δηλαδή $a_r = a_{mw+x} = b_x = a_x$). Θεωρούμε την απόσταση $a_{r+w-x} = a_{w-x}$. Ισχύει $a_{r+w-x} = a_{(m+1)w} = b_{mw}$. Συνεπώς $a_{r+w-x+1} = b_{mw+1} = b_1 = a_1$ και γενικά $a_{r+w-x+i} = b_{mw+i} = b_i = a_i$. Όμως ο όρος a_{w-x} εμφανίζεται πριν από τον όρο a_w και συνεπώς ο όρος a_{w-x+1} εμφανίζεται πριν από τον

όρο $a_{w+1} = b_1$. Αυτό σημαίνει πως για τον πράκτορα c που βρίσκεται στον κόμβο u_{w-x+1} (ο οποίος είναι μεταξύ των a και b στην ακολουθία a_+) η ακολουθία c_+ στην εικόνα του είναι ίδια με την ακολουθία a_+ το οποίο είναι άτοπο.

■

Από τα Λήμματα 5.13, 5.14 και 5.15 συμπεραίνουμε ότι:

Πόρισμα 5.16 Ένας σχηματισμός χωρίς πολλαπλότητες είναι *non-rigid*, αν και μόνο αν είναι είτε *περιοδικός* είτε *συμμετρικός*.

Λήμμα 5.17 Εάν ένας σχηματισμός χωρίς πολλαπλότητες είναι *non-rigid* και *μη-περιοδικός* τότε έχει *ακριβώς έναν* άξονα συμμετρίας.

Απόδειξη. Λόγω του Πορίσματος 5.16, ο σχηματισμός πρέπει να είναι συμμετρικός με τουλάχιστον έναν άξονα συμμετρίας S_1. Ας υποθέσουμε ότι υπάρχει και ένας δεύτερος άξονας συμμετρίας S_2. Θεωρούμε δύο πράκτορες a, b οι οποίοι είναι τοποθετημένοι σε συμμετρικούς κόμβους ως προς τον άξονα S_1. Τότε όπως αποδείξαμε στο Πόρισμα 5.16, αν $a_+ = (a_1, \ldots, a_r)$ είναι η ακολουθία των αποστάσεων στη δεξιόστροφη κατεύθυνση στην εικόνα του πράκτορα a και $b_- = (b_r, b_{r-1}, \ldots, b_1)$ είναι η ακολουθία των αποστάσεων στη αριστερόστροφη κατεύθυνση στην εικόνα του πράκτορα b, ισχύει $a_+ = b_-$. Εάν έστω ένας από αυτούς τους πράκτορες (έστω ο πράκτορας a) βρίσκεται πάνω στον άξονα S_2 τότε $(a_1, \ldots, a_r) = (a_r, a_{r-1}, \ldots, a_1)$ και συνεπώς $(a_1, \ldots, a_r) = (b_1, \ldots, b_r)$. Σε αυτήν την περίπτωση όμως επιχειρηματολογώντας όπως στην περίπτωση (ii) της απόδειξης του Λήμματος 5.15, μπορούμε να συμπεράνουμε ότι ο σχηματισμός είναι περιοδικός, το οποίο οδηγεί σε άτοπο. Εάν κανείς από τους πράκτορες a, b δεν βρίσκεται πάνω στον άξονα S_2 τότε θεωρούμε έναν τρίτο πράκτορα c που έχει την ίδια εικόνα με τον b εξαιτίας του άξονα S_2. Αυτό σημαίνει ότι $(b_1, \ldots, b_r) = (c_r, c_{r-1}, \ldots, c_1)$ (όπου $c_- = (c_r, c_{r-1}, \ldots, c_1)$ είναι η ακολουθία των αποστάσεων στην αριστερόστροφη φορά στην εικόνα του c). Άρα $(a_1, \ldots, a_r) = (c_1, \ldots, c_r)$. Επιχειρηματολογώντας όπως προηγουμένως μπορούμε να συμπεράνουμε ότι και σε αυτή την περίπτωση ο σχηματισμός είναι περιοδικός, το οποίο είναι άτοπο. ■

Έστω ένας σχηματισμός χωρίς πολλαπλότητες που είναι non-rigid και μη-περιοδικός. Τότε με βάση το Πόρισμα 5.16, ο σχηματισμός είναι συμμετρικός και με βάση το Λήμμα 5.17 έχει μοναδικό άξονα συμμετρίας. Έστω S ο μοναδικός άξονας συμμετρίας. Αν το πλήθος των πρακτόρων είναι περιττός αριθμός τότε ακριβώς ένας από αυτούς βρίσκεται πάνω στον S και ο S περνά από τον αντιδιαμετρικό κόμβο αν το πλήθος n των κόμβων του δακτυλίου είναι άρτιος αριθμός, ή περνά από το μέσο κάποιας ακμής αν το n είναι περιττός αριθμός. Αν το πλήθος των πρακτόρων είναι άρτιος αριθμός τότε οι δυνατές περιπτώσεις είναι οι εξής:

- *συμμετρία ακμής-ακμής* : ο άξονας S περνά από το μέσο δύο ακμών,
- *συμμετρία κόμβου-ακμής* : τουλάχιστον ένας κόμβος βρίσκεται στον άξονα της συμμετρίας.

Παρατηρήστε ότι η πρώτη περίπτωση παραπάνω μπορεί να συμβεί μόνο σε έναν δακτύλιο με άρτιο πλήθος κόμβων στον οποίο βρίσκεται ένα άρτιο πλήθος πρακτόρων.

5.2.3 Αρνητικά αποτελέσματα

Λήμμα 5.18 Η συνάντηση δύο πρακτόρων είναι αδύνατη σε οποιονδήποτε σχηματισμό και οποιονδήποτε δακτύλιο.

Απόδειξη. Έστω ότι υπάρχει ένας σωστός αλγόριθμος για το πρόβλημα της συνάντησης δύο πρακτόρων. Σε οποιοδήποτε σχηματισμό οι πράκτορες έχουν την ίδια εικόνα. Ας δούμε τί κάνει ο αλγόριθμος όταν η απόσταση μεταξύ των πρακτόρων είναι ακριβώς μία ακμή. Εάν ο αλγόριθμος δίνει οδηγία στους πράκτορες να μην κινηθούν τότε ένας τέτοιος αλγόριθμος δεν μπορεί να είναι σωστός. Εάν ο αλγόριθμος δίνει την οδηγία να κινηθούν τότε είναι δυνατόν ο προγραμματισμός όλων των φάσεων των πρακτόρων να γίνει συγχρονισμένα, γεγονός που δεν θα τους επιτρέψει να συναντηθούν: οι πράκτορες θα βρίσκονται πάντα σε απόσταση περιττού μήκους (στην περίπτωση που ο αλγόριθμος τους δίνει την οδηγία να κινηθούν ο ένας προς τον άλλο ενώ βρίσκονται σε απόσταση 1 απλά θα αλλάξουν θέσεις παραμένοντας σε απόσταση 1). ■

Λήμμα 5.19 Εάν δεν υπάρχει η ικανότητα της ανίχνευσης πολλαπλότητας τότε η συνάντηση οποιουδήποτε σχηματισμού $k > 1$ πρακτόρων είναι αδύνατη σε οποιονδήποτε δακτύλιο.

Απόδειξη. Θα το αποδείξουμε με επαγωγή στο πλήθος k των πρακτόρων. Για $k = 2$ πράκτορες προκύπτει από το Λήμμα 5.18. Έστω ότι ισχύει για πλήθος $k' < k$. Θεωρούμε έναν σωστό αλγόριθμο για k πράκτορες. Έστω C ο σχηματισμός μόλις πριν δημιουργηθεί η πρώτη πολλαπλότητα. Αμέσως μετά τουλάχιστον ένας πράκτορας R κινείται σε έναν γειτονικό του κόμβο στον οποίο βρίσκεται κάποιος άλλος πράκτορας. Ο χρονισμός των φάσεων μπορεί να γίνει έτσι ώστε πρώτα προγραμματίζονται οι φάσεις Look-Compute-Move μόνο για τον πράκτορα R, και έπειτα η φάση Look για τους υπόλοιπους πράκτορες. Ο πράκτορας R δημιουργεί μια πολλαπλότητα και έτσι μειώνει των αριθμό των κόμβων στους οποίους βρίσκονται πράκτορες σε $k - 1$. Όλες οι επόμενες φάσεις Look θα αφορούν το πολύ $k - 1$ κόμβους στους οποίους βρίσκονται πράκτορες. Αφού δεν υπάρχει η ικανότητα της ανίχνευσης πολλαπλότητας, η συμπεριφορά των πρακτόρων θα είναι η ίδια με εκείνη της περίπτωσης λιγότερων από k. Από την επαγωγική υπόθεση προκύπτει ότι η συνάντηση όλων των πρακτόρων είναι αδύνατη. ■

Θεώρημα 5.20 Η συνάντηση είναι αδύνατη για κάθε *περιοδικό* σχηματισμό.

Απόδειξη. Έστω ένας περιοδικός σχηματισμός με περίοδο που επαναλαμβάνεται $t > 1$ φορές. Έστω το εξής σύστημα χρονισμού: πρώτα εκτελείται συγχρονισμένα η φάση Look για όλους τους πράκτορες, έπειτα συγχρονισμένα η φάση Compute για όλους, και μετά η φάση Move συγχρονισμένα για όλους. Υπενθυμίζουμε στον αναγνώστη ότι οι πράκτορες είναι oblivious, δηλαδή δεν μπορούν να κρατήσουν στη μνήμη τους παρατηρήσεις προηγούμενων κύκλων. Έστω ότι ο σχηματισμός παραμένει περιοδικός έως το γύρο r. Οι εικόνες όλων των t αντίστοιχων πρακτόρων που βρίσκονται στα t αντίγραφα της περιόδου είναι πανομοιότυπες στο γύρο r και συνεπώς ο σχηματισμός παραμένει περιοδικός στο γύρο $r + 1$, με t αντίγραφα της περιόδου. Συνεπώς επαγωγικά ο σχηματισμός παραμένει περιοδικός σε κάθε γύρο, με t αντίγραφα της περιόδου. Εφόσον $t > 1$, η συνάντηση όλων των πρακτόρων είναι αδύνατη. ■

Θεώρημα 5.21 Η συνάντηση είναι αδύνατη για κάθε *συμμετρικό ακμής - ακμής* σχηματισμό.

Απόδειξη. Έστω ένας σχηματισμός με συμμετρία ακμής-ακμής. Αυτό σημαίνει ότι το πλήθος των κόμβων του δακτυλίου καθώς και το πλήθος των πρακτόρων είναι άρτιοι αριθμοί. Έστω το εξής σύστημα χρονισμού: πρώτα εκτελείται συγχρονισμένα η φάση Look για όλους τους πράκτορες, έπειτα συγχρονισμένα η φάση Compute για όλους, και μετά η φάση Move συγχρονισμένα για όλους. Έστω ότι ο σχηματισμός παραμένει συμμετρικός έως το γύρο r. Έστω R' ο πράκτορας που βρίσκεται σε συμμετρική θέση ως προς τον πράκτορα R. Η απόσταση μεταξύ των δύο πρακτόρων R και R' πρέπει να είναι περιττός αριθμός. Οι πράκτορες R και R' έχουν την ίδια εικόνα στο γύρο r, και συνεπώς θα πρέπει να συμπεριφερθούν ακριβώς με τον ίδιο τρόπο (υπενθυμίζουμε ότι οι πράκτορες είναι oblivious). Άρα η απόστασή τους στο γύρο $r + 1$ είτε θα μείνει η ίδια, ή θα αυξηθεί κατά δύο ακμές ή θα μειωθεί κατά δύο ακμές. Αυτό σημαίνει ότι στο γύρο $r + 1$ ο σχηματισμός θα παραμείνει συμμετρικός (οι πράκτορες R και R' θα έχουν και πάλι την ίδια εικόνα), και η απόσταση μεταξύ των R, R' θα παραμείνει περιττός αριθμός. Από επαγωγή, ο σχηματισμός παραμένει συμμετρικός και η απόσταση μεταξύ ενός πράκτορα και του συμμετρικού του παραμένει περιττός αριθμός σε όλους τους γύρους. Άρα η συνάντηση είναι αδύνατη. ■

5.2.4 Αλγόριθμοι συνάντησης

Σε αυτήν την ενότητα παρουσιάζουμε και αναλύουμε, αλγόριθμους για σχηματισμούς στους οποίους είναι δυνατή η συνάντηση των πρακτόρων. Ξεκινάμε με την παρουσίαση μιας απλής αλγοριθμικής διαδικασίας που οδηγεί σε συνάντηση έναν οποιονδήποτε σχηματισμό με ακριβώς μία πολλαπλότητα. Στη συνέχεια δίνουμε έναν αλγόριθμο που οδηγεί σε συνάντηση έναν rigid σχηματισμό. Τέλος παρουσιάζουμε έναν αλγόριθμο που οδηγεί σε συνάντηση οποιονδήποτε μη-περιοδικό σχηματισμό ενός περιττού πλήθους πρακτόρων.

5.2.4.1 Σχηματισμοί με ακριβώς μία πολλαπλότητα

Η παρακάτω διαδικασία `Single-Multiplicity-Gathering` οδηγεί σε συνάντηση οποιονδήποτε σχηματισμό πρακτόρων στον οποίον υπάρχει ακριβώς μία πολλαπλότητα. Η διαδικασία κα-

ταφέρνει να συγκεντρώσει όλους τους πράκτορες στον κόμβο της πολλαπλότητας, αποφεύγοντας τη δημιουργία άλλης πολλαπλότητας.

Η ιδέα της διαδικασίας είναι η εξής: Έστω v ο κόμβος με την πολλαπλότητα. Αρχικά οι πράκτορες που βρίσκονται πιο κοντά στον v κινούνται μέχρι να φτάσουν στον v. Έπειτα κινούνται στον v οι πράκτορες που βρίσκονται τώρα πιο κοντά στον v, κ.ο.κ.

Οι λεπτομέρειες της διαδικασίας φαίνονται στον Αλγόριθμο 26.

Αλγόριθμος 26 Διαδικασία Single-Multiplicity-Gathering

```
1: if δεν βρίσκεσαι στην πολλαπλότητα then
2:     μην κινείσαι
3: else
4:     if κανένα από τα διαστήματα μεταξύ εσένα και της πολλαπλότητας δεν είναι ελεύθερο then
5:         μην κινείσαι
6:     else
7:         να κινηθείς προς την πολλαπλότητα διαλέγοντας το πιο σύντομο μονοπάτι από τα ελεύθερα
           διαστήματα ή διαλέγοντας οποιοδήποτε από αυτά σε περίπτωση ισότητας
8:     end if
9: end if
```

Λήμμα 5.22 Ο αλγόριθμος *Single-Multiplicity-Gathering* οδηγεί σε συνάντηση μετά από πεπερασμένο χρόνο έναν οποιονδήποτε σχηματισμό από πράκτορες με ακριβώς μία πολλαπλότητα.

Απόδειξη. Η διαδικασία εγγυάται ότι ένας πράκτορας κινείται μόνο στην περίπτωση που κάποιο διάστημα μεταξύ αυτού και της πολλαπλότητας είναι ελεύθερο, και σε αυτή την περίπτωση ο πράκτορας κινείται σε αυτό το ελεύθερο διάστημα. Αυτό σημαίνει ότι δεν θα δημιουργηθεί καμμία άλλη πολλαπλότητα εκτός από την αρχική. Αφού για κάθε σχηματισμό με ακριβώς μία πολλαπλότητα, υπάρχει τουλάχιστον ένας πράκτορας που δεν βρίσκεται στην πολλαπλότητα και έχει κάποιο ελεύθερο διάστημα προς την πολλαπλότητα, μετά από κάθε κύκλο κάποιος πράκτορας εκτός πολλαπλότητας κάνει ένα βήμα προς την πολλαπλότητα. Συνεπώς αν τη χρονική στιγμή t υπάρχουν ακόμη πράκτορες εκτός πολλαπλότητας, τότε μετά από πεπερασμένο χρόνο, κάποιος πράκτορας θα φτάσει στην πολλαπλότητα, μειώνοντας έτσι το πλήθος των πρακτόρων εκτός πολλαπλότητας. Επειδή οι πράκτορες στην πολλαπλότητα δεν κινούνται ποτέ, η συνάντηση όλων των πρακτόρων θα πραγματοποιηθεί μετά από πεπερασμένο χρόνο. ■

5.2.4.2 Η συνάντηση σε *rigid* σχηματισμούς

Δίνουμε τώρα μια διαδικασία η οποία οδηγεί σε συνάντηση οποιονδήποτε rigid σχηματισμό πρακτόρων. Η ιδέα της διαδικασίας είναι η εκλογή ενός πράκτορα και μετακίνησή του μέχρι να δημιουργηθεί μία πολλαπλότητα. Έπειτα καλείται η διαδικασία `Single-Multiplicity-Gathering`.

Ο πράκτορας που θα εκλεγεί πρέπει να είναι τέτοιος ώστε κατά τη διάρκεια της κίνησής του ο σχηματισμός να παραμένει rigid. Η διαδικασία πετυχαίνει αυτόν το σκοπό εκτελώντας επαναληπτικά τα εξής:

- Πρώτα εκλέγεται ένα ζεύγος γειτονικών πρακτόρων A, B οι οποίοι βρίσκονται σε μέγιστη (ελεύθερη) απόσταση ο ένας από τον άλλον (μπορεί να υπάρχουν πολλά τέτοια ζεύγη).

- Μετά επιλέγεται ένας από τους A, B, και συγκεκριμένα εκείνος ο οποίος έχει τη μικρότερη (ελεύθερη) απόσταση από κάποιον άλλον πράκτορα (εκτός των A, B). Οι περιπτώσεις ισοπαλιών μπορούν να επιλυθούν εύκολα (βλ. τις λεπτομέρειες του αλγόριθμου).

- Ο επιλεγμένος (από τους A, B) πράκτορας κινείται προς την κατεύθυνση που αυξάνει την απόσταση των A, B.

Για την εκλογή ενός πράκτορα, κάθε πράκτορας φτιάχνει μια ταξινομημένη λεξικογραφικά λίστα των εικόνων όλων των πρακτόρων. Εφόσον η εικόνα κάθε πράκτορα δεν είναι τίποτα άλλο από ένα ζεύγος πεπερασμένων ακολουθιών φυσικών αριθμών, τότε κάθε πράκτορας μπορεί να βάλει σε διάταξη το κάθε ζεύγος από αυτά (καλούμε *code* το διατεταγμένο ζεύγος μιας εικόνας) και στη συνέχεια να βάλει σε διάταξη όλα τα ζεύγη (codes).

Οι λεπτομέρειες της διαδικασίας φαίνονται στον Αλγόριθμο 27.

Λήμμα 5.23 Ο αλγόριθμος *Rigid-Gathering* οδηγεί μετά από πεπερασμένο χρόνο τους πράκτορες σε συνάντηση εάν ο αρχικός σχηματισμός είναι *rigid* και δεν έχει πολλαπλότητες.

Απόδειξη. Έστω ότι οι πράκτορες M και N σε απόσταση Max εκλέγονται κατά το πρώτο μέρος της διαδικασίας. Θεωρούμε χωρίς βλάβη της γενικότητας ότι η απόσταση μεταξύ των M και M' είναι μικρότερη από την απόσταση μεταξύ των N και N'. Έστω ότι η απόσταση μεταξύ των M και M_2 είναι a και η απόσταση μεταξύ των N και N_2 είναι b. Ο πράκτορας M κινείται προς τον πράκτορα M_2. Μετά από αυτή την κίνηση η απόσταση μεταξύ των M και N γίνεται $Max + 1$, και η απόσταση μεταξύ των M και M_2 γίνεται $a - 1$. Καμμία άλλη απόσταση μεταξύ γειτονικών πρακτόρων δεν αλλάζει. Συνεπώς στο νέο σχηματισμό, οι πράκτορες M και N είναι και πάλι γειτονικοί σε μέγιστη απόσταση. Ο νέος σχηματισμός είναι rigid διότι το ζεύγος γειτονικών πρακτόρων M και N είναι το μοναδικό σε μέγιστη απόσταση $Max + 1$ και η απόσταση $a - 1$ μεταξύ των M και M_2 είναι μικρότερη από την απόσταση b μεταξύ των N και N_2. Έτσι λοιπόν οι M και N εκλέγονται ξανά. Αφού η απόσταση $a - 1$ μεταξύ των M, M_2 είναι μικρότερη από την απόσταση b μεταξύ των N, N_2, ο πράκτορας M επιλέγεται και κινείται προς τον M_2. Άρα έως ότου να δημιουργηθεί μια πολλαπλότητα, μόνο ένας πράκτορας κινείται προς την ίδια κατεύθυνση. Αυτό εγγυάται ότι μετά από πεπερασμένο χρόνο θα δημιουργηθεί ακριβώς μία πολλαπλότητα. Έπειτα καλείται η διαδικασία `Single-Multiplicity-Gathering` η οποία ολοκληρώνει τον αλγόριθμο συνάντησης. ■

Αλγόριθμος 27 Διαδικασία Rigid-Gathering

1: $Max \leftarrow$η μεγαλύτερη από τις αποστάσεις a_i στην εικόνα του πράκτορα
2: $M \leftarrow$ο πράκτορας με το μεγαλύτερο (λεξικογραφικά) code της εικόνας ο οποίος έχει έναν γειτονικό πράκτορα σε απόσταση Max
3: $N \leftarrow$ο πράκτορας με το μεγαλύτερο (λεξικογραφικά) code της εικόνας ο οποίος έχει τον M γειτονικό σε απόσταση Max
4: /* ένα ζεύγος γειτονικών πρακτόρων σε απόσταση Max έχει επιλεγεί. */
5: $j \leftarrow 1$; $M_1 \leftarrow M$; $N_1 \leftarrow N$; $M_0 \leftarrow N$; $N_0 \leftarrow M$;
6: **repeat**
7: $j \leftarrow j + 1$;
8: $M_j \leftarrow$ο γειτονικός πράκτορας του M_{j-1} διαφορετικός από τον M_{j-2}
9: $N_j \leftarrow$ο γειτονικός πράκτορας του N_{j-1} διαφορετικός από τον N_{j-2}
10: **until** η απόσταση μεταξύ των N και N_j είναι διαφορετική από εκείνη μεταξύ των M και M_j
11: $N' \leftarrow N_j$; $M' \leftarrow M_j$;
12: **if** δεν υπάρχει πολλαπλότητα **then**
13: **if** η απόσταση μεταξύ των N και N' είναι μικρότερη από εκείνη μεταξύ των M και M' **then**
14: **if** είσαι ο πράκτορας N **then**
15: να κινηθείς προς τον N_2
16: **end if**
17: **else**
18: **if** είσαι ο πράκτορας M **then**
19: να κινηθείς προς τον M_2
20: **end if**
21: **end if**
22: **else**
23: Single-Multiplicity-Gathering
24: **end if**

5.2.4.3 Η συνάντηση περιττού αριθμού από πράκτορες

Τέλος παρουσιάζουμε τον αλγόριθμο `Odd-Gathering` (Αλγόριθμος 28) ο οποίος οδηγεί σε συνάντηση οποιονδήποτε μη-περιοδικό σχηματισμό ενός περιττού πλήθους πρακτόρων. Ο αλγόριθμος αυτός σε συνδυασμό με το Θεώρημα 5.20, επιλύουν όλες τις περιπτώσεις του προβλήματος της συνάντησης ενός περιττού πλήθους πρακτόρων.

Η ιδέα του αλγορίθμου έχει ως εξής. Έστω ένας οποιοσδήποτε μη-περιοδικός σχηματισμός ενός περιττού πλήθους πρακτόρων χωρίς πολλαπλότητες. Εάν ο σχηματισμός είναι rigid τότε εκτελείται η διαδικασία `Rigid-Gathering`. Διαφορετικά ο σχηματισμός πρέπει να είναι συμμετρικός (πρβλ. Πόρισμα 5.16) με ακριβώς έναν άξονα συμμετρίας (πρβλ. Λήμμα 5.17). Εφόσον το πλήθος

των πρακτόρων είναι περιττός αριθμός υπάρχει ακριβώς ένας αξονικός πράκτορας. Τότε εκτελείται επαναληπτικά η εξής διαδικασία:

Ο αξονικός πράκτορας μετακινείται σε έναν από τους γειτονικούς του κόμβους. Όπως αποδεικνύουμε αργότερα, τρεις περιπτώσεις μπορούν να εμφανιστούν για το σχηματισμό που προκύπτει.

(1) Ο σχηματισμός που προκύπτει έχει ακριβώς μία πολλαπλότητα: τότε εκτελείται η διαδικασία `Single-Multiplicity-Gathering`.

(2) Ο σχηματισμός που προκύπτει είναι rigid: τότε εκτελείται η διαδικασία `Rigid-Gathering`.

(3) Ο σχηματισμός που προκύπτει έχει ακριβώς έναν άξονα συμμετρίας διαφορετικό όμως από τον άξονα του σχηματισμού πριν την κίνηση (προφανώς ο προηγούμενος άξονας συμμετρίας, λόγω της κίνησης, δεν μπορεί να διατηρηθεί) στον οποίον υπάρχει ακριβώς ένας αξονικός πράκτορας.

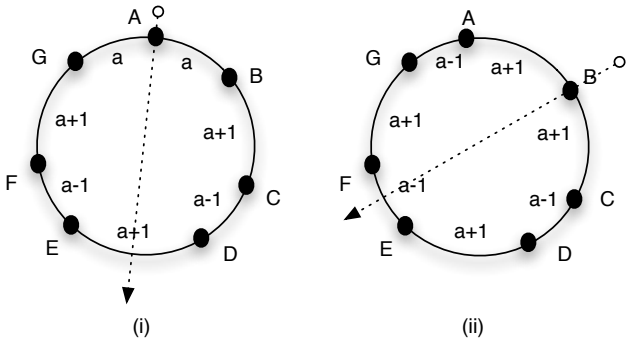
Αργότερα αποδεικνύουμε ότι μετά από έναν πεπερασμένο αριθμό επαναλήψεων της παραπάνω διαδικασίας θα προκύψει είτε η περίπτωση (1) είτε η περίπτωση (2), και συνεπώς η συνάντηση θα επιτευχθεί με την εκτέλεση των διαδικασιών `Single-Multiplicity-Gathering` και `Rigid-Gathering`.

Αλγόριθμος 28 Odd-Gathering

```
1:  if ο σχηματισμός είναι περιοδικός then
2:      η συνάντηση είναι αδύνατη
3:  else
4:      if ο σχηματισμός έχει ακριβώς μία πολλαπλότητα then
5:          Single-Multiplicity-Gathering
6:      else
7:          if ο σχηματισμός είναι rigid then
8:              Rigid-Gathering
9:          else
10:             if είσαι ο αξονικός πράκτορας then
11:                 να κινηθείς προς ένα οποιονδήποτε από τους γειτονικούς κόμβους
12:             end if
13:         end if
14:     end if
15: end if
```

Πριν αποδείξουμε την ορθότητα του Αλγόριθμου 28 ας δώσουμε ένα παράδειγμα εκτέλεσής του.

Παράδειγμα 5.24 Έστω ο σχηματισμός $C = (a, a+1, a-1, a+1, a-1, a+1, a)$ από 7 πράκτορες, όπου $a > 1$ (βλ. Σχήμα 5.7(i)). Ο σχηματισμός C είναι συμμετρικός μη-περιοδικός με



Σχήμα 5.7: Μία ακολουθία από μη-περιοδικούς, συμμετρικούς σχηματισμούς.

τον αξονικό πράκτορα να βρίσκεται σε απόσταση a από τους γειτονικούς του πράκτορες. Έπειτα από την κίνηση του αξονικού πράκτορα προς έναν από τους γειτονικούς του πράκτορες, προκύπτει ο σχηματισμός $C' = (a+1, a+1, a-1, a+1, a-1, a+1, a-1)$ όπως φαίνεται στο Σχήμα 5.7(ii), ο οποίος είναι επίσης συμμετρικός και μη-περιοδικός. Ο αξονικός πράκτορας στον C' βρίσκεται σε απόσταση $a+1$ από τους γειτονικούς του πράκτορες. Μετά την κίνηση του αξονικού πράκτορα στον C' προς κάποιον από τους γειτονικούς του πράκτορες, προκύπτει ένας rigid σχηματισμός στον οποίον η διαδικασία `Rigid-Gathering` μπορεί να εφαρμοστεί και να οδηγήσει τους πράκτορες σε συνάντηση.

Αρχικά αποδεικνύουμε ότι κατά την εκτέλεση του Αλγόριθμου 28 δεν μπορεί να προκύψει περιοδικός σχηματισμός.

Λήμμα 5.25 Έστω C ένας συμμετρικός σχηματισμός ενός περιττού πλήθους πρακτόρων χωρίς πολλαπλότητες. Έστω C' ο σχηματισμός που προκύπτει από τον C μετά την μετακίνηση του πράκτορα που βρίσκεται πάνω στον άξονα συμμετρίας, σε κάποιον από τους γειτονικούς κόμβους. Εάν ο σχηματισμός C' δεν έχει πολλαπλότητες τότε δεν είναι περιοδικός.

Απόδειξη. Έστω $C = (a, b_1, \ldots, b_{s-1}, b_s, b_{s-1}, \ldots, b_1, a)$ ένας συμμετρικός και μη-περιοδικός σχηματισμός. Μετά την κίνηση του αξονικού πράκτορα σε έναν από τους γειτονικούς του κόμβους, ο σχηματισμός C' που προκύπτει είναι της μορφής $(a+1, b_1, \ldots, b_{s-1}, b_s, b_{s-1}, \ldots, b_1, a-1)$. Έστω ότι ο C' είναι περιοδικός και έστω d η περίοδός του με μήκος p. Τότε ισχύουν $d_1 = a+1$ και $d_p = a-1$. Αλλά από τη συμμετρία του C προκύπτει $d_p = d_1$. Άτοπο. ■

Στη συνέχεια αποδεικνύουμε ότι η εφαρμογή του Αλγόριθμου 28 σε ένα συμμετρικό και μη-περιοδικό σχηματισμό προκαλεί μόνο έναν πεπερασμένο αριθμό από μεταβάσεις σε συμμετρικούς σχηματισμούς.

Έστω ένας σχηματισμός $C = (a_1, \ldots, a_r)$ χωρίς πολλαπλότητες. Καλούμε *range* του C το σύνολο $\{a_1, \ldots, a_r\}$. Για κάθε ακέραιο a_i στο range του C, καλούμε *weight* του a_i το πλήθος των εμφανίσεων αυτού του ακεραίου στην ακολουθία $(a_1, \ldots, a_r)$.

Λήμμα 5.26 Έστω `Cf` ένας συμμετρικός μη-περιοδικός σχηματισμός ενός περιττού πλήθους πρακτόρων χωρίς πολλαπλότητες. Υπάρχει ακριβώς μία τιμή στο range του `Cf` που έχει περιττό weight.

Απόδειξη. Από το Λήμμα 5.17, προκύπτει ότι ο σχηματισμός `Cf` έχει ακριβώς έναν άξονα συμμετρίας. Έστω S αυτός ο άξονας, και έστω C και D το μοναδικό ζεύγος γειτονικών πρακτόρων, οι οποίοι είναι τοποθετημένοι εκατέρωθεν του S (βλ. Σχήμα 5.4). Έστω x το μήκος του ελεύθερου διαστήματος μεταξύ των C και D. Έστω y μια οποιαδήποτε τιμή διαφορετική του x, στο range `Cf`. Για κάθε ζεύγος γειτονικών πρακτόρων με ελεύθερο διάστημα μεταξύ τους μήκους y, υπάρχει ένα συμμετρικό ζεύγος πρακτόρων με ελεύθερο διάστημα μεταξύ τους ίδιου μήκους. Αυτό σημαίνει πως το y πρέπει να έχει άρτιο weight. Επιπλέον, για κάθε ζεύγος γειτονικών πρακτόρων διαφορετικών από τους C, D, με ελεύθερο διάστημα μεταξύ τους μήκους x, υπάρχει ένα συμμετρικό ζεύγος πρακτόρων με ελεύθερο διάστημα μεταξύ τους ίδιου μήκους. Άρα εξαιτίας της ύπαρξης του ζεύγους C, D, προκύπτει ότι το x πρέπει να έχει περιττό weight. ■

Έστω `Cf` ένας συμμετρικός μη-περιοδικός σχηματισμός ενός περιττού πλήθους από πράκτορες χωρίς πολλαπλότητες (βλ. παράδειγμα Σχήματος 5.4). Η μοναδική τιμή με περιττό weight στον σχηματισμό `Cf` καλείται *chief* του `Cf`. Αυτή είναι η τιμή του μήκους του ελεύθερου διαστήματος (μεταξύ δύο γειτονικών πρακτόρων) από το μέσο του οποίου περνά ο άξονας συμμετρίας. Η απόσταση μεταξύ του αξονικού πράκτορα και των γειτονικών του πρακτόρων ονομάζεται *index* του `Cf`. Στο Σχήμα 5.4 ο chief είναι η τιμή 4 του μήκους του ελεύθερου διαστήματος μεταξύ των πρακτόρων C και D. Το index είναι η τιμή 2 της απόστασης μεταξύ των πρακτόρων A και E (ή A και B). Έστω $\texttt{Cf}'$ ο σχηματισμός που προκύπτει από τον `Cf` μετά την κίνηση του αξονικού πράκτορα προς κάποιον από τους γειτονικούς του κόμβους. Εάν ο σχηματισμός $\texttt{Cf}'$ δεν έχει πολλαπλότητες και είναι συμμετρικός τότε τον ονομάζουμε *special* σχηματισμό. Δηλαδή *special* ονομάζεται ένας συμμετρικός σχηματισμός χωρίς πολλαπλότητες που έχει προκύψει από κάποιον άλλον συμμετρικό σχηματισμό μετά την κίνηση του αξονικού πράκτορα.

Λήμμα 5.27 Έστω C ένας συμμετρικός μη-περιοδικός σχηματισμός ενός περιττού πλήθους πρακτόρων χωρίς πολλαπλότητες. Έστω z και f τα index και chief του C αντίστοιχα. Έστω C' ο σχηματισμός που προκύπτει από τον C μετά την κίνηση του αξονικού πράκτορα σε κάποιον από τους γειτονικούς του κόμβους. Εάν ο C' είναι special σχηματισμός τότε $z = f - 1$ ή $z = f + 1$ και ο chief του C' είναι είτε $z + 1$ είτε $z - 1$.

Απόδειξη. Αν τα weights των $z + 1$ και $z - 1$ είναι και τα δύο άρτιοι αριθμοί στον C', τότε και τα δύο πρέπει να ήταν περιττοί αριθμοί στον σχηματισμό C. Αυτό όμως αντιβαίνει στο Λήμμα 5.26. Για τον ίδιο λόγο, ακριβώς ένα από αυτά πρέπει να έχει περιττό weight στον C. Συνεπώς ο chief f του C είτε είναι $f = z - 1$ ή $f = z + 1$. Επιπλέον, ακριβώς ένα από τα $z - 1$ και $z + 1$ έχει άρτιο weight στον C και άρα θα έχει περιττό weight στον C' και συνεπώς αυτός είναι ο chief του C'. ■

Έστω ένας special σχηματισμός C. Ορίζουμε τα παρακάτω σύνολα:

- *White part*: το σύνολο των ακεραίων στο range του C με ομοτιμία ίδια με του chief.
- *Black part*: το σύνολο των ακεραίων στο range του C με διαφορετική ομοτιμία από εκείνη του chief.

Για παράδειγμα αν ο chief του C είναι περιττός αριθμός τότε το white part είναι το σύνολο των περιττών ακεραίων ενώ το black part είναι το σύνολο των άρτιων ακεραίων στο range του C.

Συμβολίζουμε με $b(C)$ το συνολικό πλήθος των εμφανίσεων στον C, ακεραίων από το black part του range του.

Λήμμα 5.28 Έστω ένας special σχηματισμός C. Έστω C' ο σχηματισμός που προκύπτει από τον C μετά την κίνηση του αξονικού πράκτορα σε κάποιον από τους γειτονικούς του κόμβους. Εάν ο C' είναι special σχηματισμός τότε $b(C') < b(C)$.

Απόδειξη. Έστω z το index του C, και f ο chief. Από το Λήμμα 5.27, προκύπτει ότι ισχύει είτε $z = f - 1$ ή $z = f + 1$. Ας υποθέσουμε ότι ισχύει η πρώτη περίπτωση, δηλαδή $f = z + 1$. Στον σχηματισμό C', το weight καθενός από τους ακεραίους $z + 1$ και $z - 1$ αυξάνει κατά 1 και το weight του ακεραίου z μειώνεται κατά 2. Αφού το weight του f ήταν περιττός αριθμός στον C, τώρα γίνεται άρτιος. Αφού το weight του $z - 1$ ήταν άρτιος αριθμός, τώρα γίνεται περιττός και ο αριθμός $z - 1$ είναι ο chief του C'. Η ομοτιμία του chief δεν αλλάζει σε σχέση με τον σχηματισμό C. Συνεπώς ο αριθμός z (αν ακόμη έχει θετικό weight στον C') βρίσκεται στο black part του range του C' διότι έχει ομοτιμία διαφορετική από εκείνη του chief. Οι ακέραιοι $z - 1$ και $z + 1$ ανήκουν στο white part του range. Κανένα άλλο weight δεν αλλάζει τιμή σε σχέση με την τιμή που είχε στον C. Έτσι λοιπόν προκύπτει ότι το άθροισμα των weights στο black part του C' είναι κατά 2 μικρότερο από το άθροισμα των weights στο black part του C. Η απόδειξη για τη δεύτερη περίπτωση, δηλαδή όταν $f = z - 1$, γίνεται ανάλογα. ■

Πόρισμα 5.29 Έστω μια ακολουθία $(C_1, C_2, \dots)$ από special σχηματισμούς, τέτοια ώστε ο C_{i+1} προκύπτει από τον C_i μετά την κίνηση του αξονικού πράκτορα σε κάποιον από τους γειτονικούς του κόμβους. Τότε για κάποιο πεπερασμένο i, ισχύει $b(C_i) = 0$. ■

Λήμμα 5.30 Έστω ένας special σχηματισμός C, με $b(C) = 0$. Έστω C' ο σχηματισμός που προκύπτει από τον C μετά την κίνηση του αξονικού πράκτορα σε κάποιον από τους γειτονικούς του κόμβους. Αν ο C' δεν έχει πολλαπλότητες τότε δεν είναι συμμετρικός.

Απόδειξη. Εάν ο C' ήταν συμμετρικός, τότε θα ήταν special σχηματισμός με $b(C') < b(C)$. Αυτό όμως αντιβαίνει στο Λήμμα 5.28 λόγω του ότι $b(C) = 0$. ■

Έτσι λοιπόν από τα παραπάνω συμπεραίνουμε ότι μετά από έναν πεπερασμένο αριθμό εμφανίσεων special σχηματισμών, ο σχηματισμός που προκύπτει είναι rigid. Τώρα μπορούμε να αποδείξουμε την ορθότητα του Αλγόριθμου 28.

Θεώρημα 5.31 Ο Αλγόριθμος 28 επιλύει το πρόβλημα της συνάντησης για οποιονδήποτε μη-περιοδικό σχηματισμό ενός περιττού πλήθους πρακτόρων.

Απόδειξη. Έστω ένας αρχικός μη-περιοδικός σχηματισμός C ενός περιττού πλήθους πρακτόρων χωρίς πολλαπλότητες. Αν ο σχηματισμός είναι rigid τότε ο αλγόριθμος είναι σωστός λόγω του Λήμματος 5.23. Διαφορετικά λόγω του Πορίσματος 5.16 και του Λήμματος 5.17, ο σχηματισμός πρέπει να είναι συμμετρικός με μοναδικό άξονα συμμετρίας. Έστω A ο μοναδικός αξονικός πράκτορας. Έστω C_1 ο σχηματισμός που προκύπτει από τον C μετά την κίνηση του πράκτορα A σε κάποιον από τους γειτονικούς του κόμβους. Αν ο C_1 έχει μια πολλαπλότητα τότε το πρόβλημα επιλύεται με την κλήση της διαδικασίας
`Single-Multiplicity-Gathering`. Αν ο C_1 είναι rigid τότε λόγω του Λήμματος 5.23 ο αλγόριθμος είναι σωστός. Διαφορετικά, ο C_1 είναι είτε περιοδικός είτε συμμετρικός, λόγω του Πορίσματος 5.16. Λόγω του Λήμματος 5.25, ο σχηματισμός δεν μπορεί να είναι περιοδικός, και συνεπώς πρέπει να είναι συμμετρικός και άρα special σχηματισμός. Έστω ο σχηματισμός C_2 που προκύπτει μετά την κίνηση του αξονικού πράκτορα στον σχηματισμό C_1 σε κάποιον από τους γειτονικούς του κόμβους. Ο C_2 είτε περιέχει μια πολλαπλότητα, είτε είναι rigid, είτε είναι special σχηματισμός. Στις δύο πρώτες περιπτώσεις είναι ξεκάθαρη η συνέχεια, ενώ στην τρίτη περίπτωση ο αξονικός πράκτορας κινείται ξανά. Με αυτόν τον τρόπο δημιουργείται μία σειρά $C_1, C_2, \ldots$ από special σχηματισμούς. Από το Πόρισμα 5.29, προκύπτει ότι υπάρχει ένας σχηματισμός C_i σε αυτήν τη σειρά, με $b(C_i) = 0$. Έστω C' ο σχηματισμός που προκύπτει από τον C_i μετά την κίνηση του αξονικού πράκτορα σε κάποιον από τους γειτονικούς του κόμβους. Τότε, με βάση το Λήμμα 5.30, ο σχηματισμός C' είτε έχει μια πολλαπλότητα, είτε δεν είναι συμμετρικός και άρα θα πρέπει να είναι rigid. Στην πρώτη περίπτωση γίνεται κλήση της διαδικασίας `Single-Multiplicity-Gathering` ενώ στη δεύτερη περίπτωση με βάση το Λήμμα 5.23 ο αλγόριθμος οδηγεί τους πράκτορες σε συνάντηση. ■

Συνεπώς από το Θεώρημα 5.20 και τον Αλγόριθμο 28 έχουμε ότι:

Θεώρημα 5.32 Η συνάντηση ενός περιττού πλήθους πρακτόρων είναι εφικτή αν και μόνο αν ο αρχικός σχηματισμός δεν είναι περιοδικός.

5.3 Σχόλια και βιβλιογραφικές παρατηρήσεις

Η ανάλυση για το πρόβλημα της συνάντησης πολλών πρακτόρων που παρουσιάστηκε στις πρώτες ενότητες αυτού του κεφαλαίου εμφανίστηκε αρχικά στο άρθρο [Flocchini et al., 2004] και στη διδακτορική διατριβή [Sawchuk, 2004]. Ένας άλλος αλγόριθμος για τη συνάντηση πολλών πρακτόρων μπορεί να βρεθεί στο άρθρο [Gasieniec et al., 2006] όπου αποδεικνύονται βέλτιστα όρια για τις απαιτήσεις μνήμης.

Η μελέτη του προβλήματος της συνάντησης ανώνυμων πανομοιότυπων πρακτόρων έχει ερευνηθεί αρκετά [Agmon and Peleg, 2006, Ando et al., 1999, Cieliebak et al., 2003, Cohen and Peleg, 2004, Flocchini et al., 2005, Prencipe, 2001, Prencipe, 2005, Suzuki and Yamashita, 1999] σε μοντέλα παρόμοια με το μοντέλο Look-Compute-Move που παρουσιάσθηκε σε αυτό το κεφάλαιο. Για παράδειγμα έχει αποδειχθεί στο [Flocchini et al., 2005] ότι η συνάντηση ασύγχρονων πρακτόρων στο *επίπεδο* είναι δυνατή αν οι πράκτορες συμφωνούν στον προσανατολισμό, ακόμη και αν έχουν περιορισμένη ορατότητα. Χωρίς συμφωνία στον προσανατολισμό, το πρόβλημα της συνάντησης επιλύθηκε στο [Cieliebak et al., 2003], θεωρώντας ότι οι πράκτορες έχουν την ικανότητα της ανίχνευσης πολλαπλότητας. Ένα συμπληρωματικό, αρνητικό αυτή τη φορά, αποτέλεσμα το οποίο αφορά σε ασύγχρονους πράκτορες αποδείχθηκε στο [Prencipe, 2005]: χωρίς την ικανότητα της ανίχνευσης πολλαπλότητας, η συνάντηση πρακτόρων που δεν έχουν κοινό προσανατολισμό είναι αδύνατη.

Μια παρουσίαση πολλών αποτελεσμάτων σχετικά με το πρόβλημα της συνάντησης πρακτόρων σε δίκτυα υπάρχει στο βιβλίο [Kranakis et al., 2010].

Το μοντέλο (Look-Compute-Move) που παρουσιάστηκε εδώ έχει μελετηθεί για το πρόβλημα της εξερεύνησης σε διάφορες τοπολογίες γραφημάτων. Στο άρθρο [Flocchini et al., 2013] οι συγγραφείς μελετούν την εξερεύνηση δακτυλίων, αποδεικνύοντας ότι η επίλυση αυτού του προβλήματος σε δακτύλιο n κόμβων από k πράκτορες είναι αδύνατη αν το k διαιρεί το n. Αποδεικνύουν επίσης ότι το ελάχιστο πλήθος πρακτόρων $p(n)$ που μπορεί να εξερευνήσει έναν δακτύλιο μεγέθους n είναι $O(\log n)$ και $p(n) = \Omega(\log n)$ για οσοδήποτε μεγάλα n.

Στο άρθρο [Flocchini et al., 2010] οι συγγραφείς έχουν χρησιμοποιήσει το ίδιο μοντέλο μελετώντας την εξερεύνηση δέντρων. Αποδεικνύουν ότι υπάρχουν δέντρα n κόμβων για τα οποία $\Omega(n)$ πράκτορες είναι αναγκαίοι, και αυτό ισχύει ακόμη και για δέντρα με μέγιστο βαθμό 4. Αποδεικνύουν επίσης πως για δέντρα μέγιστου βαθμού 3 $\Theta(\frac{\log n}{\log \log n})$ πράκτορες είναι αναγκαίοι και αρκούν για την εξερεύνηση.

Στο άρθρο [Chalopin et al., 2010] οι συγγραφείς θεωρούν ένα παρόμοιο μοντέλο. Η μόνη διαφορά από το μοντέλο που εξετάσαμε εδώ είναι ότι θεωρούν έναν τοπικό προσανατολισμό του γραφήματος. Με άλλα λόγια μοντελοποιούν το δίκτυο με ένα γράφημα που έχει ετικέττες στις ακμές. Μελετούν τα προβλήματα της συνάντησης και της εξερεύνησης και δείχνουν ότι $k = 2l + 1$ πράκτορες μπορούν να συναντηθούν σε οποιοδήποτε συνεκτικό γράφημα εάν η αρχική τοποθέτηση των πρακτόρων δεν είναι συμμετρική. Για το πρόβλημα της εξερεύνησης δείχνουν ότι ενώ 3 πράκτορες δεν μπορούν να εξερευνήσουν όλα τα γραφήματα, $k = 2l + 1$ πράκτορες μπορούν να εξερευνήσουν οποιοδήποτε συνεκτικό γράφημα εάν η αρχική τοποθέτηση των πρακτόρων δεν είναι συμμετρική. Για τις τοπολογίες δακτυλίων αποδεικνύουν ότι τα δύο προβλήματα είναι επιλύσιμα ανεξάρτητα από το πλήθος των πρακτόρων $k > 2$ αν και μόνο αν η αρχική τοποθέτηση των πρακτόρων δεν είναι συμμετρική. Το ίδιο ισχύει και για τοπολογίες δέντρων αναφορικά με το πρόβλημα της συνάντησης ενώ για το πρόβλημα της εξερεύνησης εάν η αρχική τοποθέτηση των πρακτόρων δεν είναι συμμετρική τότε $k \geq 4$ πράκτορες αρκούν.

Υπάρχουν επίσης αρκετά άρθρα που επεκτείνουν τα αποτελέσματα αυτού του κεφαλαίου. Στο άρθρο [Klasing et al., 2010] οι συγγραφείς μελετούν το πρόβλημα της συνάντησης πρακτόρων σε δακτύλιους στο ίδιο μοντέλο (Look-Compute-Move) με αυτό που παρουσιάστηκε εδώ, και δίνουν μια μέθοδο που βασίζεται στη διατήρηση των συμμετριών (σε αντίθεση με τη μέθοδο αυτού του κεφαλαίου). Σε αυτό το άρθρο αποδεικνύεται πως η συνάντηση ενός σχηματισμού από περισσότερους από 18 πράκτορες είναι δυνατή αν και μόνο αν ο σχηματισμός δεν είναι περιοδικός και δεν έχει συμμετρία ακμής-ακμής. Μερικές από τις υπόλοιπες περιπτώσεις, δηλαδή για λιγότερους από 18 πράκτορες μελετήθηκαν στα άρθρα [Haba et al., 2008] και [D'Angelo et al., 2011] και η μόνη περίπτωση που παραμένει ανοικτή είναι εκείνη ενός άρτιου πλήθους μεταξύ 8 και 18 πρακτόρων σε συμμετρικούς σχηματισμούς κόμβου-κόμβου ή κόμβου-ακμής χωρίς αξονικούς πράκτορες.

Μια ενδιαφέρουσα επέκταση του προβλήματος που παρουσιάστηκε εδώ αναφέρεται στο άρθρο [Dieudonne and Petit, 2009] όπου οι συγγραφείς μελετούν την επιλυσιμότητα του προβλήματος της συνάντησης από πολύ αδύναμους πράκτορες. Εισάγουν την έννοια της ισχυρής ανίχνευσης πολλαπλότητας η οποία ορίζεται ως η ικανότητα των πρακτόρων να ανιχνεύουν το ακριβές πλήθος των πρακτόρων πάνω σε οποιοδήποτε κόμβο.

Σημειώνουμε ότι το πρόβλημα της συνάντησης στο σενάριο που περιγράψαμε έχει σχέση με το πρόβλημα της εκλογής αρχηγού (βλ. e.g. [Lynch, 1996]), δεν είναι όμως κατ' ανάγκη ισοδύναμα. Εάν οι πράκτορες στον αρχικό σχηματισμό δεν μπορούν να εκλέξουν έναν κόμβο (αυτό για παράδειγμα συμβαίνει για όλους τους περιοδικούς σχηματισμούς καθώς και για κάποιους συμμετρικούς σχηματισμούς) τότε η συνάντηση είναι αδύνατη. Ακόμη όμως και αν είναι δυνατή η εκλογή ενός κόμβου στον αρχικό σχηματισμό, αυτό δεν εγγυάται υποχρεωτικά την επιλυσιμότητα του προβλήματος της συνάντησης. Παρ' όλο που ο κόμβος που επιλέγεται είναι φυσικός υποψήφιος για τον τόπο συνάντησης των πρακτόρων, δεν είναι καθόλου προφανές πώς θα διατηρηθεί ο ίδιος υποψήφιος κόμβος κατά τη διάρκεια οποιουδήποτε αλγόριθμου συνάντησης εξαιτίας του αδύναμου μοντέλου. (Θυμίζουμε ότι οι κόμβοι του δικτύου δεν έχουν ετικέττες, και οι σχηματισμοί που αντιλαμβάνονται οι πράκτορες (μέσω των εικόνων που παίρνουν κατά τη φάση Look) αλλάζουν όσο εξελίσσεται η διαδικασία, και συνεπώς ένας επιλεγμένος κόμβος από τους πράκτορες κάποια δεδομένη στιγμή μπορεί να μην αναγνωρίζεται σε μετέπειτα στιγμές).

Ένα σχετικό πρόβλημα είναι το πρόβλημα της συνάντησης πολλών ρομπότ στο επίπεδο (γνωστό επίσης σαν point formation) όπου η αποστολή των ρομπότ είναι να συναντηθούν σε ένα σημείο (όχι αναγκαστικά γνωστό από πριν). Τα ρομπότ μοντελοποιούνται σαν σημεία στο επίπεδο. Επίσης σχετικό είναι το πρόβλημα της σύγκλισης, όπου ο σκοπός είναι τα ρομπότ να συγκλίνουν σε ένα σημείο χωρίς να το φτάνουν αναγκαστικά. Η συνάντηση και η σύγκλιση έχουν ερευνηθεί σε μοντέλα περιορισμένης και απεριόριστης ορατότητας σε σχέση με το τί είναι ικανά τα ρομπότ να δουν στο χώρο στα άρθρα [Dieudonne and Petit, 2009, Flocchini et al., 2001, Klasing et al., 2008a, Klasing et al., 2008b, Prencipe and Cieliebak, 2002].

Η λύση πολλών προβλημάτων, όπως τα παιχνίδια pursuit-evasion (βλέπε τα άρθρα [Bonato et al., 2007, Bonato and Chiniforooshan, 2009]) έχει σχέση με την επίλυση του προβλήματος της συνάντησης. Το πρόβλημα της συνάντησης έχει μελετηθεί με στοιχεία της Θε-

ωρίας Παιγνίων (βλέπε για παράδειγμα τα άρθρα [Anderson and Essegaier, 1995, Baston, 1999, Baston and Gal, 1998, Baston and Gal, 2001]). Η συνάντηση σε γεωμετρικά περιβάλλοντα έχει μελετηθεί στα άρθρα [Anderson and Fekete, 2001, Anderson and Fekete, 1998].

Η συνάντηση πολλών πρακτόρων ενδιαφέρει επίσης τη Θεωρία Ελέγχου (Control Theory). Εκεί ενδιαφέρει η συλλογική συμπεριφορά μιας ομάδας από $n > 1$ κινητούς αυτόνομους πράκτορες (ή ρομπότ) που κινούνται στο επίπεδο με διαφορετικές ταυτότητες από 1 μέχρι n. Τα ρομπότ είναι ικανά να βλέπουν συνεχώς τις θέσεις εκείνων των ρομπότ που βρίσκονται 'κοντά' τους. Αυτή η περιοχή συνήθως ορίζεται σαν ένας δίσκος ακτίνας r με κέντρο την τρέχουσα θέση του ρομπότ. Όπως και στο μοντέλο που μελετήθηκε σε αυτό το κεφάλαιο, το πρόβλημα είναι η ανακάλυψη 'τοπικών' στρατηγικών ελέγχου για κάθε ρομπότ ξεχωριστά, έτσι ώστε όλα τα μέλη της ομάδας να μπορούν να συναντηθούν σε ένα σημείο που δεν είναι γνωστό από πριν, χωρίς να είναι απαραίτητη η επικοινωνία μεταξύ τους. Σχετική βιβλιογραφία για αυτό το πρόβλημα περιλαμβάνει τα άρθρα [Lin et al., 2003, Lin et al., 2007, Lin et al., 2004].

Βιβλιογραφία

[Agmon and Peleg, 2006] Agmon, N. and Peleg, D. (2006). Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82.

[Anderson and Fekete, 1998] Anderson, E. and Fekete, S. (1998). Asymmetric rendezvous on the plane. In *Proc. of the fourteenth annual Symp. on Computational geometry*, pages 365–373. ACM New York, NY, USA.

[Anderson and Essegaier, 1995] Anderson, E. J. and Essegaier, S. (1995). Rendezvous search on the line with indistinguishable players. *SIAM Journal of Control and Optimization*, 33:1637–1642.

[Anderson and Fekete, 2001] Anderson, E. J. and Fekete, S. (2001). Two-dimensional rendezvous search. *Operations Research*, 49:107–188.

[Ando et al., 1999] Ando, H., Oasa, Y., Suzuki, I., and Yamashita, M. (1999). A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *I.E.E.E. Transactions on Robotics and Animation*, 15(5):818–828.

[Barriere et al., 2007] Barriere, L., Flocchini, P., Fraigniaud, P., and Santoro, N. (2007). Rendezvous and election of mobile agents: impact of sense of direction. *Theory of Computing Systems*, 40(2):143–162.

[Baston, 1999] Baston, V. (1999). Two rendezvous search problems on the line. *Naval Research Logistics*, 46:335–340.

[Baston and Gal, 1998] Baston, V. and Gal, S. (1998). Rendezvous on the line when the players' initial distance is given by an unknown probability distribution. *SIAM Journal of Control and Optimization*, 36(6):1880–1889.

[Baston and Gal, 2001] Baston, V. and Gal, S. (2001). Rendezvous search when marks are left at the starting points. *Naval Research Logistics*, 47(6):722–731.

[Bonato and Chiniforooshan, 2009] Bonato, A. and Chiniforooshan, E. (2009). Pursuit and evasion from a distance: algorithms and bounds. In *Proc. of the Fifth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 1–10.

[Bonato et al., 2007] Bonato, A., Prałat, P., and Wang, C. (2007). Pursuit-Evasion in Models of Complex Networks. *Internet Mathematics*, 4(4):419–436.

[Chalopin et al., 2010] Chalopin, J., Flocchini, P., Mans, B., and Santoro, N. (2010). Network exploration by silent and oblivious robots. In *Proceedings of Graph Theoretic Concepts in Computer Science*, LNCS 6410, pages 208–219.

[Cieliebak et al., 2003] Cieliebak, M., Flocchini, P., Prencipe, G., and Santoro, N. (2003). Solving the robots gathering problem. In *Proceedings of 30th International Colloquium on Automata, Languages and Programming*, LNCS 2719, pages 1181–1196.

[Cohen and Peleg, 2004] Cohen, R. and Peleg, D. (2004). Robot convergence via center-of-gravity algorithms. In *Proceedings of 11th International Colloquium on Structural Information and Communication Complexity*, LNCS 3104, pages 79–88.

[D'Angelo et al., 2011] D'Angelo, G., Stefano, G. D., and Navarra, A. (2011). Gathering of six robots on anonymous symmetric rings. In *Proceedings of 18th International Colloquium on Structural Information and Communication Complexity*, LNCS 6796, pages 174–185.

[Dieudonne and Petit, 2009] Dieudonne, Y. and Petit, F. (2009). Self-stabilizing deterministic gathering. In *Proceedings of 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, LNCS 5804, pages 230–241.

[Flocchini et al., 2010] Flocchini, P., Ilcinkas, D., Pelc, A., and Santoro, N. (2010). Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theoretical Computer Science*, 411(14–15):1583 – 1598.

[Flocchini et al., 2013] Flocchini, P., Ilcinkas, D., Pelc, A., and Santoro, N. (2013). Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65(3):562–583.

[Flocchini et al., 2004] Flocchini, P., Kranakis, E., Krizanc, D., Santoro, N., and Sawchuk, C. (2004). Multiple mobile agent rendezvous in the ring. In *Proceedings of 6th Latin American Theoretical Informatics Symposium*, pages 599–608.

[Flocchini et al., 2001] Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. (2001). Gathering of asynchronous oblivious robots with limited visibility. In *Proceedings of Symposium on Theoretical Aspects of Computer Science*, LNCS 2010, pages 247–258.

[Flocchini et al., 2005] Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. (2005). Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168.

[Gasieniec et al., 2006] Gasieniec, L., Kranakis, E., Krizanc, D., and Zhang, X. (2006). Optimal memory rendezvous of anonymous mobile agents in a uni-directional ring. In *Proceedings of 32nd International Conference on Current Trends in Theory and Practice of Computer Science*, LNCS 3831, pages 282–292.

[Haba et al., 2008] Haba, K., Izumi, T., Katayama, Y., Inuzuka, N., and Wada, K. (2008). On gathering problem in a ring for 2n autonomous mobile robots. In *Proceedings of 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, poster.

[Klasing et al., 2008a] Klasing, R., Kosowski, A., and Navarra, A. (2008a). Taking advantage of symmetries: Gathering of asynchronous oblivious robots on a ring. In *Proceedings of 12th International Conference on Principles of Distributed Systems*, LNCS 5401, pages 446–462.

[Klasing et al., 2010] Klasing, R., Kosowski, A., and Navarra, A. (2010). Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theoretical Computer Science*, 411(34–36):3235 – 3246.

[Klasing et al., 2008b] Klasing, R., Markou, E., and Pelc, A. (2008b). Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science*, 390:27–39.

[Kranakis et al., 2010] Kranakis, E., Krizanc, D., and Markou, E. (2010). *The Mobile Agent Rendezvous Problem in the Ring*. Synthesis Lectures on Distributed Computing Theory, (Ed.) N. Lynch,. Morgan & Claypool Publishers.

[Kranakis et al., 2003] Kranakis, E., Krizanc, D., Santoro, N., and Sawchuk, C. (2003). Mobile agent rendezvous search problem in the ring. In *Proceedings of International Conference on Distributed Computing Systems*, pages 592–599.

[Lin et al., 2003] Lin, J., Morse, A. S., and Anderson, B. D. O. (2003). The multi-agent rendezvous problem. In *Proc. of the 42nd IEEE Conf. on Decision and Control*, volume 2, pages 1508–1513.

[Lin et al., 2004] Lin, J., Morse, A. S., and Anderson, B. D. O. (2004). The Multi-Agent Rendezvous Problem. An Extended Summary. *Cooperative Control*, 309:451–454.

[Lin et al., 2007] Lin, J., Morse, A. S., and Anderson, B. D. O. (2007). The Multi-Agent Rendezvous Problem. Part 2: The Asynchronous Case. *SIAM J. on Control and Optimization*, 46(6):2120–2147.

[Lynch, 1996] Lynch, N. (1996). *Distributed Algorithms*. Morgan Kaufman.

[Prencipe, 2001] Prencipe, G. (2001). Corda: Distributed coordination of a set of autonomous mobile robots. In *Proceedings of European Research Seminar on Advances in Distributed Systems*, pages 185–190.

[Prencipe, 2005] Prencipe, G. (2005). On the feasibility of gathering by autonomous mobile robots. In *Proceedings of 12th International Colloquium on Structural Information and Communication Complexity*, LNCS 3499, pages 246–261.

[Prencipe and Cieliebak, 2002] Prencipe, G. and Cieliebak, M. (2002). Gathering autonomous mobile robots. In *Proceedings of 9th International Colloquium on Structural Information and Communication Complexity*, pages 57–72.

[Sawchuk, 2004] Sawchuk, C. (2004). *Mobile Agent Rendezvous in the Ring*. PhD thesis, School of Computer Science, Carleton University, Ottawa, Canada.

[Suzuki and Yamashita, 1999] Suzuki, I. and Yamashita, M. (1999). Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363.

ΚΕΦΑΛΑΙΟ 6

Ανακάλυψη Εχθρικών Κόμβων σε Δακτύλιους και Δέντρα

Σε αυτό το κεφάλαιο περιγράφουμε και μελετούμε μοντέλα εχθρικών κόμβων σε δίκτυα τοπολογίας δακτυλίου και δέντρου. Εστιάζουμε ιδιαίτερα στο πρόβλημα της μαύρης τρύπας και περιγράφουμε αλγόριθμους για συγχρονισμένα και ασύγχρονα δίκτυα καθώς και αρνητικά αποτελέσματα.

6.1 Εξερεύνηση σε εχθρικά δίκτυα

Ένα σημαντικό ζήτημα στους κατανεμημένους υπολογισμούς με κινητούς πράκτορες είναι η ασφάλεια των πρακτόρων και των κόμβων του δικτύου [Chess, 1998, Greenberg et al., 1998, Oppliger, 1999, Schelderup and Ølnes, 1999, Borselius, 2002, Jansen, 2000]. Η περίπτωση εχθρικών κινητών πρακτόρων (που μπορεί να αναπαριστά ιούς που μολύνουν κάθε κόμβο του δικτύου που επισκέπτονται) έχει μελετηθεί στη διεθνή βιβλιογραφία. Ο καθαρισμός του μολυσμένου δικτύου μπορεί να πραγματοποιηθεί από μια ομάδα πρακτόρων που είναι ικανοί να καθαρίσουν τους κόμβους που επισκέπτονται και να εμποδίσουν νέα μόλυνση της καθαρής περιοχής. Το πρόβλημα αυτό είναι ισοδύναμο με το πρόβλημα της εύρεσης ενός φυγά που κινείται σε ένα δίκτυο. Αποτελέσματα σε αυτό και άλλα σχετικά προβλήματα έχουν εμφανιστεί στα άρθρα [Asaka et al., 1999, Barriere et al., 2002, Flocchini et al., 2005, Flocchini et al., 2006, Foukia et al., 2001, Luccio et al., 2006, Spafford and Zamboni, 2000].

Η προστασία των πρακτόρων από επιθέσεις των κόμβων, δηλαδή επικίνδυνων ιών που βρίσκονται σε κόμβους του δικτύου, είναι τόσο απαραίτητη όσο και η προστασία ενός κόμβου από επιθέσεις ενός εχθρικού πράκτορα. Μια σειρά από μεθόδους για την προστασία των πρακτόρων από επιθέσεις κόμβων έχουν προταθεί (π.χ., [Hohl, 1998, Hohl, 2000, Ng and Cheung, 1999, Sander and Tschudin, 1998, Schelderup and Ølnes, 1999, Vitek and Castagna, 1999]). Στο άρ-

θρο [Flocchini and Santoro, 2012] γίνεται μια επισκόπηση αποτελεσμάτων που αφορούν εχθρικούς κόμβους (ιδιαίτερα για ασύγχρονα δίκτυα) αλλά και εχθρικούς πράκτορες.

Τα τελευταία χρόνια, το πρόβλημα της εξερεύνησης έχει μελετηθεί σε μη-ασφαλή δίκτυα τα οποία περιέχουν εχθρικούς κόμβους που καλούνται *μαύρες τρύπες (black holes)*. Η μαύρη τρύπα είναι ένας κόμβος ο οποίος έχει την ικανότητα να καταστρέφει όλους τους πράκτορες που τον επισκέπτονται χωρίς να αφήνει κανένα ίχνος. Στο πρόβλημα της *Αναζήτησης της Μαύρης Τρύπας* ο σκοπός των πρακτόρων είναι η ανακάλυψη της μαύρης τρύπας μέσα σε πεπερασμένο χρόνο. Πιο συγκεκριμένα, τουλάχιστον ένας πράκτορας πρέπει να επιζήσει γνωρίζοντας όλες τις ακμές που οδηγούν στη μαύρη τρύπα. Το πρόβλημα αυτό αρχικά παρουσιάστηκε από τους S. Dobrev, P. Flocchini, G. Prencipe and N. Santoro το 2001 ([Dobrev et al., 2001]). Ο μοναδικός τρόπος της αναγνώρισης μιας μαύρης τρύπας είναι βάζοντας τουλάχιστον έναν πράκτορα να την επισκεφτεί. Φυσικά, εφόσον κάθε πράκτορας που επισκέπτεται μια μαύρη τρύπα εξαφανίζεται χωρίς να αφήσει ίχνη, η θέση της μαύρης τρύπας θα πρέπει να προκύψει μέσω κάποιου μηχανισμού επικοινωνίας που χρησιμοποιούν οι πράκτορες. Τέσσερις τέτοιοι μηχανισμοί έχουν προταθεί στη βιβλιογραφία: α) το μοντέλο του *πίνακα* στο οποίο υπάρχει ένας πίνακας σε κάθε κόμβο του δικτύου, όπου οι πράκτορες μπορούν να αφήνουν μηνύματα, β) το μοντέλο του *'αγνού' σημαδιού* κατά το οποίο οι πράκτορες μεταφέρουν σημάδια, τα οποία μπορούν να αφήσουν στους κόμβους, γ) το μοντέλο του *'ενισχυμένου' σημαδιού* στο οποίο οι πράκτορες μπορούν να αφήσουν σημάδια σε κόμβους ή ακμές, και δ) ο μηχανισμός χρονισμού (μόνο για συγχρονισμένα δίκτυα) στο οποίο ένας πράκτορας εξερευνά έναν νέο κόμβο και στη συνέχεια (μετά από ένα προκαθορισμένο χρόνο) πληροφορεί έναν άλλο πράκτορα που περιμένει σε ασφαλή κόμβο.

Ο πιο ισχυρός μηχανισμός επικοινωνίας μεταξύ πρακτόρων είναι οι πίνακες σε όλους τους κόμβους. Δεδομένου ότι η πρόσβαση σε έναν πίνακα γίνεται με αμοιβαίο αποκλεισμό, το μοντέλο αυτό παρέχει στους πράκτορες και έναν τρόπο για το σπάσιμο των συμμετριών: οι πράκτορες που ξεκινούν από τον ίδιο κόμβο, μπορούν να πάρουν διαφορετικές ταυτότητες και στη συνέχεια οι διακριτοί πλέον πράκτορες να αναθέσουν διαφορετικές ετικέτες σε όλους τους κόμβους του δικτύου. Ως εκ τούτου, σε αυτό το μοντέλο, αν οι πράκτορες ξεκινούν αρχικά στον ίδιο κόμβο, τόσο οι πράκτορες όσο και οι κόμβοι μπορεί να θεωρηθούν ότι έχουν διαφορετικές ταυτότητες χωρίς καμία απώλεια της γενικότητας. Ενώ το μοντέλο του πίνακα χρησιμοποιείται συνήθως σε μη ασφαλή δίκτυα, το μοντέλο του σημαδιού έχει χρησιμοποιηθεί ως επί το πλείστον στην εξερεύνηση ασφαλών δικτύων. Το μοντέλο των αγνών σημαδιών μπορεί να υλοποιηθεί με $O(1)$-bit πίνακες, για ένα σταθερό αριθμό πρακτόρων και σταθερό αριθμό σημαδιών, ενώ το μοντέλο του ενισχυμένου σημαδιού μπορεί να υλοποιηθεί με ένα $O(\log \Delta)$-bit πίνακα σε κάθε κόμβο, όπου Δ είναι ο μέγιστος βαθμός στο γράφημα. Στο μοντέλο του πίνακα, το μέγεθος του κάθε πίνακα συνήθως είναι τουλάχιστον $\Omega(\log n)$ bits, όπου n είναι ο αριθμός των κόμβων του δικτύου. Στα ασύγχρονα δίκτυα με δεδομένο ότι όλοι οι πράκτορες έχουν μνήμη και αρχικά ξεκινούν από τον ίδιο ασφαλή κόμβο, το πρόβλημα αναζήτησης μαύρης τρύπας έχει μελετηθεί στο μοντέλο του πίνακα (π.χ., [Dobrev et al., 2006a, Dobrev et al., 2006c, Dobrev et al., 2007a, Dobrev et al., 2004, Glaus, 2009, Balamohan et al., 2010, Balamohan et al., 2011b,

Balamohan et al., 2011a, Flocchini et al., 2012b]), στο μοντέλο του ενισχυμένου σημαδιού (π.χ., [Dobrev et al., 2006b, Dobrev et al., 2006d, Shi, 2009]) και στο μοντέλο του αγνού σημαδιού [Flocchini et al., 2008, Flocchini et al., 2012a]. Έχει αποδειχθεί ότι το πρόβλημα μπορεί να λυθεί από έναν ελάχιστο αριθμό πρακτόρων που εκτελούν έναν πολυωνυμικό αριθμό κινήσεων. Σε ένα ασύγχρονο δίκτυο, ο αριθμός των κόμβων του δικτύου πρέπει να είναι γνωστός στους πράκτορες διαφορετικά το πρόβλημα δεν μπορεί να λυθεί ([Dobrev et al., 2007a]). Εάν η τοπολογία του γραφήματος είναι άγνωστη, χρειάζονται τουλάχιστον $\Delta + 1$ πράκτορες, όπου Δ είναι ο μέγιστος βαθμός στο γράφημα ([Dobrev et al., 2006c]). Επιπλέον, το δίκτυο θα πρέπει να είναι 2-συνδεδεμένο, ενώ δεν είναι δυνατόν να απαντηθεί το ερώτημα της *ύπαρξης* μιας μαύρης τρύπας στο δίκτυο. Αν οι πράκτορες έχουν ένα χάρτη του δικτύου ή τουλάχιστον μια *αίσθηση της κατεύθυνσης* ([Flocchini et al., 1998, Flocchini et al., 2003]) και μπορούν να χρησιμοποιήσουν πίνακες, τότε δύο πράκτορες με μνήμη αρκούν για να λύσουν το πρόβλημα.

Σε ασύγχρονα δίκτυα με διάσπαρτους πράκτορες (δηλ. που δεν ξεκινούν στο ίδιο κόμβο), το πρόβλημα έχει διερευνηθεί για την τοπολογία δακτυλίου ([Dobrev et al., 2003, Dobrev et al., 2007a]) και για γενικές τοπολογίες ([Flocchini et al., 2009, Chalopin et al., 2007]) στο μοντέλο του πίνακα, ενώ στο μοντέλο του ενισχυμένου σημαδιού έχει μελετηθεί για δακτύλιους ([Dobrev et al., 2007b, Dobrev et al., 2008]) και για ορισμένα διασυνδεδεμένα δίκτυα ([Shi, 2009]).

Η περίπτωση των συγχρονισμένων δικτύων επιφέρει μερικές σοβαρές αλλαγές στις συνθήκες που επιτρέπουν τη λύση του προβλήματος. Τώρα δύο πράκτορες με μνήμη και διαφορετικές ταυτότητες που ξεκινούν στον ίδιο κόμβο μπορούν να ανακαλύψουν μια μαύρη τρύπα σε κάθε τοπολογία για την οποία έχουν ένα χάρτη με τη χρήση του μηχανισμού χρονισμού, χωρίς την ανάγκη πινάκων ή σημαδιών. Επιπλέον, το δίκτυο δεν απαιτείται να είναι 2-συνδεδεμένο, ενώ τώρα είναι δυνατό να δοθεί απάντηση στο ερώτημα της ύπαρξης ή όχι μιας μαύρης τρύπας στο δίκτυο. Έτσι το ζήτημα τώρα δεν είναι η ανακάλυψη, αλλά ο ελάχιστος χρόνος αναζήτησης της μαύρης τρύπας. Το πρόβλημα αυτό της αποτελεσματικής αναζήτησης της μαύρης τρύπας έχει μελετηθεί σε συγχρονισμένα δίκτυα χωρίς πίνακες ή σημάδια (μόνο με τη χρήση του μηχανισμού χρονισμού) στα άρθρα [Cooper et al., 2006, Cooper et al., 2010, Czyzowicz et al., 2006, Czyzowicz et al., 2007, Klasing et al., 2007, Klasing et al., 2008] υπό την προϋπόθεση ότι όλοι οι διαφορετικοί πράκτορες ξεκινούν στον ίδιο κόμβο. Ωστόσο, όταν οι πράκτορες είναι διάσπαρτοι στο δίκτυο, ο μηχανισμός χρονισμού, δεν είναι αρκετός για την επίλυση του προβλήματος.

Στις περισσότερες εργασίες που μελετάται το πρόβλημα της αναζήτησης μιας μαύρης τρύπας στο μοντέλο των σημαδιών εκτός από τα άρθρα [Flocchini et al., 2008, Flocchini et al., 2012a, Chalopin et al., 2011b, Chalopin et al., 2011a, Balamohan et al., 2012], οι συγγραφείς έχουν χρησιμοποιήσει το μοντέλο του ενισχυμένου σημαδιού για πράκτορες με διαφορετικές ταυτότητες και μη-σταθερή μνήμη. Το πιο περιορισμένο μοντέλο του αγνού σημαδιού έχει χρησιμοποιηθεί στα [Flocchini et al., 2008, Flocchini et al., 2012a, Balamohan et al., 2012] για πράκτορες που ξεκινούν στον ίδιο κόμβο και έχουν μη-σταθερή μνήμη σε ασύγχρονα δίκτυα. Τα πρώτα αποτελέσματα για διάσπαρτους πράκτορες με σταθερή μνήμη και αγνά σημάδια εμ-

φανίστηκαν στο [Chalopin et al., 2011b] για συγχρονισμένους δακτύλιους χωρίς προσανατολισμό και στο [Chalopin et al., 2011a] για συγχρονισμένα τορικά δίκτυα με προσανατολισμό. Στο [Chalopin et al., 2011a] αποδείχθηκε ότι τρεις διάσπαρτοι πράκτορες με σταθερή μνήμη και δύο σημάδια ο καθένας μπορούν να εντοπίσουν τη μαύρη τρύπα σε κάθε συγχρονισμένο και προσανατολισμένο τόρο.

Όπως συμβαίνει συχνά σε ζητήματα ανάλυσης αλγορίθμων για συγκεκριμένα προβλήματα, η απόδειξη της ορθότητας ενός αλγόριθμου για το πρόβλημα της αναζήτησης μιας μαύρης τρύπας, και η επίτευξη άνω φραγμάτων για το χρόνο που χρειάζεται καθώς και η απόδειξη της αδυναμίας λύσης για το πρόβλημα κάτω από ένα ορισμένο μοντέλο, χρησιμοποιούν συχνά την έννοια του *αντιπάλου*. Η ανάλυση του προβλήματος της αναζήτηση μιας μαύρης τρύπας κάτω από ένα μοντέλο μπορεί τότε να θεωρηθεί ως ένα παιχνίδι μεταξύ ενός προτεινόμενου αλγόριθμου και ενός *αντίπαλου* που ανταγωνίζεται τους πράκτορες και χρησιμοποιεί τη δύναμή του για να κάνει τον αλγόριθμο να αποτύχει. Όσο πιο αδύναμο είναι το μοντέλο, τόσο πιο ισχυρός είναι ο αντίπαλος ο οποίος μπορεί να επιλέξει όλες τις παραμέτρους του προβλήματος για τις οποίες ο αλγόριθμος δεν έχει καμία γνώση. Για παράδειγμα, ο αντίπαλος μπορεί να αποφασίσει τις αρχικές θέσεις των πρακτόρων, τη θέση της μαύρης τρύπας και, ανάλογα με το μοντέλο, την τοπολογία ή το μέγεθος του δικτύου, τον αριθμό των πρακτόρων, τις ετικέτες των ακμών, και (σε ασύγχρονα δίκτυα) το χρόνο που χρειάζεται ένας πράκτορας για να διασχίσει μια ακμή. Ένας σωστός αλγόριθμος θα πρέπει φυσικά να λειτουργεί για οποιεσδήποτε επιλογές του αντιπάλου. Η επίλυση του προβλήματος είναι αδύνατη όταν ο αντίπαλος μπορεί να οδηγήσει οποιονδήποτε αλγόριθμο σε αποτυχία.

Η συνέχεια του κεφαλαίου αυτού έχει ως εξής. Στην ενότητα 6.2 παρουσιάζουμε μερικά αποτελέσματα για δύο ή περισσότερους πράκτορες με μνήμη, που ξεκινούν στον ίδιο κόμβο ενός ασύγχρονου δακτυλίου με πίνακες στους κόμβους. Στη συνέχεια μελετούμε το πρόβλημα σε συγχρονισμένα δίκτυα για δύο πράκτορες με μνήμη που ξεκινούν στον ίδιο κόμβο, χρησιμοποιώντας μόνο το μηχανισμό χρονισμού. Ειδικότερα μελετούμε το πρόβλημα σε τοπολογίες δέντρων (ενότητα 6.3) και παρουσιάζουμε έναν αλγόριθμο βέλτιστου χρόνου για ειδικές τοπολογίες δέντρων και έναν προσεγγιστικό αλγόριθμο για γενικά δέντρα. Τέλος, στην ενότητα 6.4 αναφέρουμε μερικά αποτελέσματα για κατευθυνόμενα δίκτυα.

6.2 Η αναζήτηση μιας μαύρης τρύπας σε ασύγχρονους δακτύλιους

Το πρόβλημα της αναζήτησης μιας μαύρης τρύπας (Black Hole Search - BHS) προτάθηκε στο άρθρο [Dobrev et al., 2001] (βλέπε επίσης το πλήρες κείμενο του άρθρου στο [Dobrev et al., 2007a]) όπου μελετήθηκε για ασύγχρονους δακτύλιους. Παρουσιάζουμε εδώ κάποια αποτελέσματα από αυτή τη θεμελιώδη εργασία. Συζητούμε πρώτα το μοντέλο που χρησιμοποιήθηκε και δίνουμε κάτω όρια για τον αριθμό των πρακτόρων και του χρόνου αναζήτησης. Στη συνέχεια δίνουμε δύο αλγόριθμους που λύνουν το πρόβλημα χρησιμοποιώντας δύο ή περισσότερους κινητούς πράκτορες με μνήμη που ξεκινούν στον ίδιο κόμβο και χρησιμοποιούν πίνακες.

6.2.1 Το μοντέλο και μερικά βασικά κάτω όρια

Θεωρούμε δύο ανώνυμους πράκτορες με μνήμη που ξεκινούν στον ίδιο κόμβο. Ο δακτύλιος είναι ανώνυμος, ασύγχρονος και αποτελείται από n κόμβους. Σε κάθε κόμβο υπάρχει ένας *πίνακας*, όπου οι πράκτορες μπορούν να αφήσουν μηνύματα. Η πρόσβαση σε έναν πίνακα γίνεται με αμοιβαίο αποκλεισμό και συνεπώς, οι πράκτορες μπορούν να αποκτήσουν διαφορετικές ταυτότητες αμέσως μόλις ξεκινούν, με τη σειρά με την οποία θα έχουν πρόσβαση στο πίνακα του κόμβου εκκίνησης (π.χ., ο πρώτος πράκτορας που έχει πρόσβαση στον πίνακα γράφει την τιμή 1, και παίρνει την ταυτότητα 1, ενώ ο επόμενος πράκτορας αυξάνει κατά ένα την τιμή που διαβάζει και παίρνει το αποτέλεσμα ως ταυτότητά του). Παρά το γεγονός ότι ο δακτύλιος ήταν αρχικά ανώνυμος, οι διαφορετικές ταυτότητες που έχουν ανατεθεί στους πράκτορες, τους επιτρέπουν να βάλουν διακριτές ετικέτες σε όλους τους κόμβους και να συμφωνήσουν στον προσανατολισμό του δακτυλίου. Ο στόχος είναι η αναφορά στον κόμβο εκκίνησης της ακριβούς θέσης της μαύρης τρύπας μετά από ένα πεπερασμένο χρονικό διάστημα από τους πράκτορες που επιζούν.

Είναι προφανές ότι η επίλυση του προβλήματος είναι αδύνατη από έναν μόνο πράκτορα αφού ο πράκτορας αυτός απλά θα εξαφανιζόταν στη μαύρη τρύπα. Άρα απαιτούνται τουλάχιστον δύο πράκτορες για τον εντοπισμό της μαύρης τρύπας.

Εξαιτίας της έλλειψης συγχρονισμού στο δίκτυο, είναι αδύνατη η διάκριση μεταξύ ενός αργού συνδέσμου (ακμής) και ενός συνδέσμου που οδηγεί σε μια μαύρη τρύπα. Η παρατήρηση αυτή μας δίνει τα παρακάτω δύο λήμματα.

Λήμμα 6.1 ([Dobrev et al., 2007a]) Το πρόβλημα της ύπαρξης ή όχι μιας μαύρης τρύπας σε ένα ασύγχρονο δίκτυο δεν λύνεται.

Λήμμα 6.2 ([Dobrev et al., 2007a]) Ο εντοπισμός της θέσης της μαύρης τρύπας είναι αδύνατος αν το μέγεθος του δικτύου είναι άγνωστο.

Το παρακάτω θεώρημα μας δίνει ένα κάτω όριο για το χρόνο που χρειάζονται οι πράκτορες προκειμένου να ανακαλύψουν μια μαύρη τρύπα σε έναν δακτύλιο.

Θεώρημα 6.3 ([Dobrev et al., 2007a]) Οποιοσδήποτε αλγόριθμος που επιλύει το πρόβλημα της αναζήτησης μιας μαύρης τρύπας σε έναν δακτύλιο n κόμβων χρειάζεται τουλάχιστον $2n - 4$ χρόνο, ανεξάρτητα από το πλήθος των πρακτόρων.

Απόδειξη. Για την αναφορά της θέσης της μαύρης τρύπας πίσω στον κόμβο εκκίνησης h, οι πράκτορες πρέπει να επισκεφτούν τουλάχιστον $n - 1$ κόμβους. Σε αντίθετη περίπτωση, αν η μαύρη τρύπα είναι τοποθετημένη σε έναν από τους τουλάχιστον δύο κόμβους που δεν επισκέπτεται κανένας πράκτορας, είναι αδύνατο να προσδιοριστεί η ακριβής θέση της. Αυτό σημαίνει ότι κάθε κόμβος εκτός από το πολύ έναν πρέπει να προσπελαστεί από τουλάχιστον έναν πράκτορα. Ας υποθέσουμε ότι η μαύρη τρύπα βρίσκεται στον κόμβο $\mathcal{B}$ σε απόσταση $n - 1$ δεξιόστροφα από τον κόμβο εκκίνησης. Τυχόν πράκτορες που ταξιδεύουν σε αριστερόστροφη κατεύθυνση από τον κόμβο εκκίνησης

θα εξαφανίζονταν στη μαύρη τρύπα, αλλά (λόγω του ασύγχρονου δικτύου) οι υπόλοιποι πράκτορες δεν μπορούν να αποφασίσουν αν η μαύρη τρύπα είναι στο $\mathcal{B}$ ή εάν η σύνδεση με τον κόμβο $\mathcal{B}$ είναι αργή. Συνεπώς, τουλάχιστον ένας πράκτορας πρέπει να ταξιδέψει τουλάχιστον $n-2$ κόμβους μακρυά από τον κόμβο εκκίνησης και στη συνέχεια ο πράκτορας πρέπει να κάνει την αναφορά στον κόμβο h. Άρα ο συνολικός αριθμός των βημάτων είναι τουλάχιστον $2n-4$. ■

6.2.2 Ένας αλγόριθμος για δύο πράκτορες

Έστω U η ανεξερεύνητη και E η ασφαλής περιοχή αντίστοιχα. Συμβολίζουμε με U^L και U^R την συνεχή ανεξερεύνητη περιοχή αριστερόστροφα και δεξιόστροφα αντίστοιχα από την ασφαλή περιοχή. Μια βασική διαδικασία που χρησιμοποιείται στον αλγόριθμο είναι η διαδικασία *Προσεκτικής Ανίχνευσης* (*Cautious Walk*) (που προτάθηκε στα [Dobrev et al., 2001, Dobrev et al., 2007a]):

Θεωρήστε έναν πράκτορα που βρίσκεται σε έναν κόμβο v_0 γειτονικό με έναν ανεξερεύνητο κόμβο v_1. Ο πράκτορας εξερευνά την μέχρι τώρα μη-ασφαλή περιοχή $U_k =< v_1, v_2, ..., v_k >$ ως εξής:

Cautious Walk:

- πριν αφήσει τον κόμβο v_i για να πάει στον κόμβο v_{i+1}, ο πράκτορας σημειώνει την έξοδο (port) που οδηγεί από τον v_i στον v_{i+1} σαν *active*,
- αμέσως αφού επισκεφτεί τον κόμβο v_{i+1}, ο πράκτορας επιστρέφει στον v_i, και σημειώνει την έξοδο (port) που οδηγεί από τον v_i στον v_{i+1} σαν *safe*,
- ο πράκτορας ελέγχει για τυχόν μηνύματα στον κόμβο v_i και (αν υπάρχει κάποιο μήνυμα) αναθέτει στον εαυτό του μια μη-ασφαλή περιοχή U'_k προς εξερεύνηση και επαναλαμβάνει τη διαδικασία.

Οι πράκτορες γνωρίζουν το πλήθος των κόμβων n του δακτυλίου. Εκτελούν τον αλγόριθμο 29. Ακολουθεί η περιγραφή του αλγορίθμου:

Έστω ότι οι πράκτορες ξεκινούν στον κόμβο h. Χρησιμοποιώντας αμοιβαίο αποκλεισμό γράφουν στον πίνακα του κόμβου h και παίρνουν διαφορετικές ταυτότητες, όπως περιγράφεται στην αρχή της ενότητας. Μπορούν επίσης να συμφωνήσουν ως προς τη δεξιόστροφη κατεύθυνση. Στη συνέχεια, διαιρούν την ανεξερεύνητη περιοχή U με $|U| = n-1$ σε δυο ανεξάρτητα μονοπάτια U^L και U^R με $|U^L| = \lceil \frac{n-1}{2} \rceil$ και $|U^R| = \lfloor \frac{n-1}{2} \rfloor$. Ο πράκτορας 1 εξερευνά την περιοχή U^L και ο πράκτορας 2 εξερευνά την περιοχή U^R εκτελώντας τη διαδικασία *Cautious Walk*. Δεδομένου ότι υπάρχει ακριβώς μία μαύρη τρύπα, ενώ οι δύο περιοχές είναι συνεχείς και δεν έχουν κοινό κόμβο, μετά από ένα πεπερασμένο χρονικό διάστημα ακριβώς ένας από τους πράκτορες θα τελειώσει την εξερεύνηση. Ας υποθέσουμε, χωρίς βλάβη της γενικότητας ότι ο πράκτορας 1 τελειώνει. Στη συνέχεια ο πράκτορας 1 διασχίζει την ασφαλή περιοχή μέχρι να συναντήσει έναν κόμβο u του οποίου η έξοδος που οδηγεί σε έναν κόμβο v δεν έχει επισημανθεί ως ασφαλής. Τότε ο πράκτορας 1 ενημερώνει την ασφαλή περιοχή και χωρίζει τη νέα ανεξερεύνητη περιοχή στα νέα συνεχή και ξένα

μεταξύ τους μονοπάτια U^L και U^R. Αναθέτει την περιοχή U^L στον εαυτό του και την περιοχή U^R για εξερεύνηση από τον πράκτορα 2 και αφήνει αυτό το μήνυμα στον κόμβο u. Τώρα ο πράκτορας 1 διασχίζει την ασφαλή περιοχή και εξερευνά τη νέα περιοχή U^L που του έχει ανατεθεί. Ο πράκτορας 2, είτε ταξιδεύει μέσα από μια αργή σύνδεση (χωρίς να έχει εξερευνήσει την περιοχή που του είχε ανατεθεί) ή έχει εξαφανιστεί στη μαύρη τρύπα. Στην πρώτη περίπτωση ο πράκτορας 2 πρώτα θα επιστρέψει στον κόμβο u, θα ελέγξει για τα μηνύματα και θα ενημερωθεί για τη νέα περιοχή που του έχει ανατεθεί για εξερεύνηση. Οι πράκτορες επαναλαμβάνουν αυτήν τη διαδικασία έως ότου ακριβώς ένας από αυτούς εξαφανίζεται στη μαύρη τρύπα. Στη συνέχεια, ο άλλος πράκτορας που επαναλαμβάνει τη διαδικασία, τελικά θα καταλήξει σε μια ασφαλή περιοχή μεγέθους $n-1$. Εκείνη τη στιγμή γνωρίζει την ακριβή τοποθεσία της μαύρης τρύπας.

Αλγόριθμος 29 (2 *πράκτορες με μνήμη σε έναν ασύγχρονο δακτύλιο με πίνακες*)

1: Οι πράκτορες παίρνουν διαφορετικές ταυτότητες και συμφωνούν ως προς τον προσανατολισμό του δακτυλίου
2: Θέσε $X := h$
3: **while** $|E| < n-1$ **do**
4: Χώρισε την περιοχή U σε δύο συνεχή ανεξάρτητα μέρη U^L (που ξεκινά αριστερόστροφα του κόμβου X) και U^R (που ξεκινά δεξιόστροφα του κόμβου X) περίπου του ίδιου μεγέθους
5: Ο πράκτορας $1(2)$ αφήνει ένα μήνυμα στον κόμβο X που αναφέρει ότι θα εξερευνήσει την περιοχή $U^L(U^R)$
6: Ο πράκτορας $1(2)$ εξερευνά την περιοχή $U^L(U^R)$ εκτελώντας τη διαδικασία *Cautious Walk*
7: Ο πράκτορας $1(2)$ διασχίζει δεξιόστροφα (αριστερόστροφα) την ασφαλή περιοχή μέχρι να φτάσει σε ένα κόμβο u στον οποίο υπάρχει μια μη-ασφαλής έξοδος
8: Ενημέρωσε τις περιοχές E και U
9: Θέσε $X := u$
10: **end while**
11: Ανέφερε τη θέση της μαύρης τρύπας

Θεώρημα 6.4 ([Dobrev et al., 2007a]) Ο αλγόριθμος 29 ανακαλύπτει τη μαύρη τρύπα σε έναν ασύγχρονο δακτύλιο από n κόμβους χρησιμοποιώντας δύο πράκτορες που ξεκινούν στον ίδιο κόμβο μέσα σε χρόνο $2n \log n + O(n)$.

Μια εύλογη ερώτηση που προκύπτει είναι η εξής: Μπορούμε να πετύχουμε μείωση του χρόνου που απαιτείται για την ανακάλυψη της μαύρης τρύπας αν έχουμε στη διάθεσή μας περισσότερους από δύο πράκτορες που ξεκινούν στον ίδιο κόμβο;

6.2.3 Η περίπτωση των $n-1$ πρακτόρων

Ο Αλγόριθμος 30 δείχνει ότι $n-1$ πράκτορες που ξεκινούν στον ίδιο κόμβο μπορούν να ανακαλύψουν τη μαύρη τρύπα σε έναν ασύγχρονο δακτύλιο από n κόμβους, μέσα σε $2n-4$ κινήσεις.

Αλγόριθμος 30 ($n-1$ *πράκτορες με μνήμη σε έναν ασύγχρονο δακτύλιο με πίνακες*)

1: Οι πράκτορες παίρνουν διαφορετικές ταυτότητες από το σύνολο $\{1, 2, ..., n-1\}$ και συμφωνούν ως προς τον προσανατολισμό του δακτυλίου
2: Ο πράκτορας i διασχίζει $i-1$ ακμές δεξιόστροφα του κόμβου εκκίνησης
3: Ο πράκτορας i διασχίζει $n-2$ ακμές αριστερόστροφα του κόμβου εκκίνησης
4: Ο πράκτορας i επιστρέφει στον κόμβο εκκίνησης ταξιδεύοντας δεξιόστροφα
5: Ο πράκτορας i αναφέρει ότι η θέση της μαύρης τρύπας βρίσκεται σε απόσταση i δεξιόστροφα από τον κόμβο εκκίνησης

Είναι εύκολο να δει κανείς ότι από τους $n-1$ πράκτορες που εκτελούν τον Αλγόριθμο 30, ακριβώς ένας πράκτορας θα επιζήσει τερματίζοντας τον αλγόριθμο, ενώ οι υπόλοιποι $n-2$ πράκτορες θα εξαφανιστούν στη μαύρη τρύπα. Παρατηρήστε ότι οι πράκτορες χρησιμοποιούν μόνο τον πίνακα του κόμβου από τον οποίο ξεκινούν προκειμένου να πάρουν διαφορετικές ταυτότητες.

Θεώρημα 6.5 ([Dobrev et al., 2007a]) Ο αλγόριθμος 30 ανακαλύπτει τη μαύρη τρύπα σε έναν ασύγχρονο δακτύλιο απο n κόμβους μέσα σε $2n-4$ κινήσεις χρησιμοποιώντας $n-1$ πράκτορες που ξεκινούν στον ίδιο κόμβο.

Απόδειξη. Υποθέστε ότι η μαύρη τρύπα βρίσκεται σε έναν κόμβο σε απόσταση $\mathcal{B}$ δεξιόστροφα από τον κόμβο 0 (που ξεκινούν οι πράκτορες). Ο πράκτορας με ταυτότητα $\mathcal{B}$ ταξιδεύει $\mathcal{B}-1+n-2+n-(\mathcal{B}+1)=2n-4$.

6.3 Η Αναζήτηση της Μαύρης Τρύπας σε ένα Συγχρονισμένο Δέντρο

Τα πρώτα αποτελέσματα στο πρόβλημα της αναζήτησης μιας μαύρης τρύπας σε συγχρονισμένα δίκτυα αφορούσαν δύο πράκτορες που δρουν σε τοπολογίες δέντρων και παρουσιάστηκαν στο άρθρο [Czyzowicz et al., 2004] (δείτε και το πλήρες άρθρο [Czyzowicz et al., 2007]). Εδώ παρουσιάζουμε το μοντέλο που χρησιμοποιήθηκε, δίνουμε κάτω όρια για το χρόνο που χρειάζεται η αναζήτηση σε συγχρονισμένα δέντρα και περιγράφουμε έναν βέλτιστο αλγόριθμο για το πρόβλημα σε ειδικές τοπολογίες δέντρων.

6.3.1 Το μοντέλο και βασικά κάτω όρια

Θεωρούμε ένα επισημασμένο δέντρο T με ρίζα στον κόμβο s που είναι ο (ασφαλής) κόμβος που ξεκινούν οι δύο πράκτορες. Οι πράκτορες είναι συγχρονισμένοι, έχουν μνήμη, ένα χάρτη του δέντρου και διαφορετικές ταυτότητες. Μπορούν να επικοινωνούν μεταξύ τους (διαβάζοντας την κατάσταση που βρίσκεται ο κάθε πράκτορας) μόνο όταν συναντιούνται σε ένα κόμβο (και όχι, για παράδειγμα, αφήνοντας μηνύματα στους κόμβους). Υποθέτουμε ότι υπάρχει το πολύ μία μαύρη τρύπα στο δίκτυο. Ο στόχος είναι η σχεδίαση ενός αλγόριθμου που παράγει ένα σχήμα αναζήτησης της μαύρης τρύπας (*BHS-scheme*) για είσοδο (T, s) (δηλαδή, ένα ζεύγος ακολουθιών με τις κινήσεις του κάθε πράκτορα). Μετά την ολοκλήρωση της αναζήτησης θα πρέπει να υπάρχει τουλάχιστον ένας πράκτορας που έχει επιζήσει ο οποίος είτε γνωρίζει την ακριβή τοποθεσία της μαύρης τρύπας ή γνωρίζει ότι δεν υπάρχει μαύρη τρύπα στο δέντρο. Οι πράκτορες που έχουν επιζήσει πρέπει να επιστρέψουν πίσω στο s. Ένας πράκτορας μπορεί να κινείται κατά μήκος των ακμών του δέντρου (κάθε κίνηση διαρκεί μία μονάδα χρόνου) ή μπορεί να περιμένει σε ένα κόμβο.

Ο χρόνος της αναζήτησης μιας μαύρης τρύπας είναι ο αριθμός των μονάδων χρόνου μέχρι την ολοκλήρωση της αναζήτησης και την επιστροφή των πρακτόρων στον κόμβο εκκίνησης, λαμβάνοντας υπόψιν τη χειρότερη περίπτωση για τη θέση της μαύρης τρύπας (ή την απουσία της, όποιο από τα δύο είναι χειρότερο). Είναι εύκολο να δείτε ότι η χειρότερη περίπτωση για ένα δεδομένο σχήμα αναζήτησης, προκύπτει όταν δεν υπάρχει μαύρη τρύπα στο δέντρο ή όταν η μαύρη τρύπα βρίσκεται στον κόμβο που επισκέπτονται τελευταίο οι πράκτορες. Και στις δύο περιπτώσεις ο χρόνος είναι ο ίδιος. Ένα σχήμα αναζήτησης ονομάζεται *ταχύτατο* ή *βέλτιστο* για μια δεδομένη είσοδο, εάν ο χρόνος που διαρκεί είναι ο ελάχιστος δυνατός για την είσοδο αυτή. Τονίζουμε εδώ ότι ο χρόνος αναζήτησης μιας μαύρης τρύπας δεν θα πρέπει να συγχέεται με την χρονική πολυπλοκότητα ενός αλγορίθμου που παράγει μια τέτοια αναζήτηση.

Δεδομένου ότι υπάρχει το πολύ μία μαύρη τρύπα στο δέντρο, σε οποιοδήποτε BHS-σχήμα θα πρέπει να προσπελαστούν όλοι οι κόμβοι του δέντρου στη χειρότερη περίπτωση (π.χ., όταν δεν υπάρχει μαύρη τρύπα). Άρα από κάθε ακμή του δέντρου θα πρέπει να περάσει τουλάχιστον ένας πράκτορας.

Όταν οι πράκτορες συναντιούνται, ανταλλάσσουν πληροφορίες σχετικά με την περιοχή που έχει εξερευνηθεί (δηλαδή, μαθαίνουν το σύνολο των ακμών ή/και κόμβων που είναι ασφαλείς). Η ακολουθία των κινήσεων στο BHS-σχήμα μεταξύ δύο διαδοχικών συναντήσεων καλείται φάση. Δεδομένου ότι μία ανεξερεύνητη ακμή θα μπορούσε να καταλήγει στη μαύρη τρύπα, έχουμε:

Λήμμα 6.6 ([Czyzowicz et al., 2007]) Σε ένα BHS-σχήμα, μία ανεξερεύνητη ακμή δεν μπορεί να προσπελαστεί και από τους δύο πράκτορες.

Αν ένας από τους πράκτορες, a, προσπαθήσει να διασχίσει περισσότερες από μία ανεξερεύνητες ακμές πριν από τη συνάντηση με τον άλλο πράκτορα b τότε ο a μπορεί να εξαφανιστεί στη

μαύρη τρύπα (η οποία έχει τοποθετηθεί κάπου στο μονοπάτι του a) και ο πράκτορας b δεν έχει αρκετές πληροφορίες για να αποφασίσει για τη σωστή θέση της μαύρης τρύπας. Έτσι:

Λήμμα 6.7 ([Czyzowicz et al., 2007]) Κατά τη διάρκεια εκτέλεσης ενός BHS-σχήματος κάθε πράκτορας μπορεί να διασχίσει το πολύ μία ανεξερεύνητη ακμή.

Έτσι λοιπόν κατά τη διάρκεια εκτέλεσης ενός BHS-σχήματος, μια ακμή μπορεί να εξερευνηθεί μόνο με τον εξής τρόπο: ένας πράκτορας διασχίζει την ακμή και αμέσως μετά είναι προγραμματισμένη μια συνάντηση (που πρόκειται να γίνει σε κάποιον ήδη ασφαλή κόμβο) όπου οι πράκτορες ανταλλάσσουν πληροφορίες σχετικά με τη συνολικά εξερευνημένη περιοχή. Αν ένας από τους πράκτορες δεν εμφανιστεί στη συνάντηση (που σημαίνει ότι εξαφανίστηκε στη μαύρη τρύπα) τότε ο άλλος συμπεραίνει την ακριβή θέση της μαύρης τρύπας. Μια ανεξερεύνητη ακμή λοιπόν, θα μπορούσε να εξερευνηθεί στην επόμενη φάση μόνο αν είναι γειτονική της περιοχής που έχει ήδη εξερευνηθεί (η οποία είναι πάντα συνεχής).

Λήμμα 6.8 ([Czyzowicz et al., 2007]) Στο τέλος κάθε φάσης, η ασφαλής περιοχή αυξάνεται κατά μία ή δύο ακμές εκτός αν βρεθεί η τοποθεσία της μαύρης τρύπας.

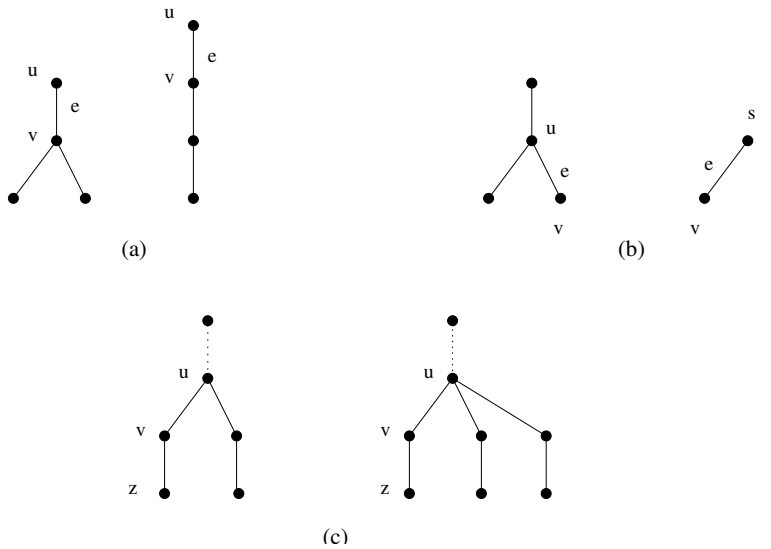
Καλούμε *1-phase* μια φάση στην οποία ακριβώς μία ακμή εξερευνάται. Ομοίως, καλούμε *2-phase* μια φάση στην οποία ακριβώς δύο ακμές εξερευνούνται. Σύμφωνα με το Λήμμα 6.8, κάθε φάση είναι είτε 1-phase είτε 2-phase.

Έστω (u, v) μια ανεξερεύνητη ακμή της οποίας ένα άκρο u είναι επίσης άκρο μιας ακμής που έχει ήδη εξερευνηθεί (ή $u \equiv s$). Θεωρήστε ότι οι δύο πράκτορες βρίσκονται στο u τη χρονική στιγμή t. Ένας τρόπος εξερεύνησης ακριβώς μιας ακμής σε μία φάση είναι ο εξής:

Διαδικασία `Probe(`v`)`: ο ένας πράκτορας διασχίζει την ακμή (u, v) και επιστρέφει στο u για να συναντηθεί με τον άλλο πράκτορα που περιμένει. Αν δεν συναντηθούν τη στιγμή $t + 2$ τότε η μαύρη τρύπα είναι στο v.

Ορίζουμε επίσης μια διαδικασία την οποία οι πράκτορες μπορούν να ακολουθήσουν για να εξερευνήσουν δύο νέες ακμές κατά τη διάρκεια μιας φάσης. Έστω $(u_1, v_1), (u_2, v_2)$ δύο ανεξερεύνητες ακμές με άκρα τα u_1, u_2 τα οποία είναι επίσης άκρα ήδη εξερευνημένων ακμών(είτε $u_1 \equiv u_2 \equiv s$). Έστω ότι ο ένας πράκτορας βρίσκεται στο u_1 και ο άλλος στο u_2 τη χρονική στιγμή t. Η διαδικασία ακολουθεί:

Διαδικασία `Split(`v_1, v_2`)`: Ο ένας από τους πράκτορες διασχίζει την ακμή (u_1, v_1) και μετά επιστρέφει στο u_1 ενώ ο άλλος διασχίζει την ακμή (u_2, v_2) και μετά επιστρέφει στο u_2. Έπειτα και οι δύο πράκτορες διατρέχουν το μονοπάτι $< u_1, u_2 >$ (το οποίο βρίσκεται εξ'ολοκλήρου στην ασφαλή περιοχή, αφού η ασφαλής περιοχή είναι συνεχής, από διαφορετικές κατευθύνσεις μέχρι να συναντηθούν όσο το δυνατόν γρηγορότερα σε κάποιο κόμβο. Έστω $dist(u_1, u_2)$ το πλήθος των ακμών στο μονοπάτι από τον κόμβο u_1 στον κόμβο u_2. Αν οι πράκτορες δεν συναντηθούν στο βήμα $t + \lceil \frac{dist(u_1,u_2)}{2} \rceil + 2$ τότε ο πράκτορας που έχει επιζήσει συμπεραίνει την ακριβή θέση της μαύρης τρύπας.



Σχήμα 6.1: (a) Η ακμή e είναι κόκκινη. (b) Η ακμή e είναι πράσινη. (c) Εκτός των διακεκομμένων ακμών, όλες οι υπόλοιπες είναι μπλε.

Και οι δύο παραπάνω διαδικασίες μοιάζουν με τη διαδικασία `Cautious-Walk` της ενότητας 6.2, με την έννοια ότι και στις δύο διαδικασίες οι πράκτορες εξερευνούν το πολύ μια ακμή ο κάθε ένας και στη συνέχεια επιστρέφουν για να επικοινωνήσουν. Δύο πράκτορες με ένα χάρτη μπορούν εύκολα να ανακαλύψουν την ακριβή θέση της μαύρης τρύπας (ή να αποφασίσουν ότι δεν υπάρχει μαύρη τρύπα) σε κάθε δέντρο μέσα σε πεπερασμένο χρόνο (π.χ., εξερευνώντας το δέντρο με Αναζήτηση κατά Βάθος χρησιμοποιώντας τη Διαδικασία `Probe`). Στην πραγματικότητα, αυτός ο απλός αλγόριθμος εγγυάται ένα λόγο προσέγγισης 4 ως προς ένα βέλτιστο BHS-σχήμα για οποιοδήποτε γενικευμένο δέντρο (βλέπε στην ενότητα 6.3.4 για λεπτομέρειες). Στη συνέχεια θα δώσουμε έναν αλγόριθμο (που εμφανίστηκε στο άρθρο [Czyzowicz et al., 2007]) ο οποίος χρησιμοποιεί και τις δύο διαδικασίες `Probe` και `Split` και λύνει το πρόβλημα BHS για μια ειδική οικογένεια δέντρων στον ελάχιστο δυνατό χρόνο. Πριν από την περιγραφή αυτού του αλγόριθμου δίνουμε ένα γενικό κάτω όριο για το χρόνο που απαιτείται από οποιονδήποτε αλγόριθμο για την επίλυση του προβλήματος σε ένα δέντρο.

6.3.2 Ένα κάτω όριο στο χρόνο αναζήτησης σε δέντρα

Έστω T ένα δέντρο με ρίζα s (την αρχική βάση των πρακτόρων) και $e = (u, v)$ μια ακμή του T (όπου v είναι ένα παιδί του u). Θεωρήστε τον εξής χρωματισμό που χωρίζει τις ακμές του δέντρου σε δύο σύνολα:

- μια ακμή e παίρνει κόκκινο χρώμα αν ο κόμβος v έχει τουλάχιστον δύο απογόνους,
- μια ακμή e παίρνει πράσινο χρώμα αν ο κόμβος v είναι φύλλο και ακριβώς ένα από τα εξής ισχύει: $u = s$ είτε η ακμή (w, u) είναι μια κόκκινη ακμή (όπου ο κόμβος w είναι γονιός του u),
- μια ακμή e παίρνει μπλέ χρώμα αν δεν έχει καμμία από τις παραπάνω ιδιότητες

Κόκκινες, πράσινες και μπλε ακμές φαίνονται στο Σχήμα 6.1.

Έστω $e = (u, v)$ και $e' = (v, z)$ δύο μπλε ακμές όπως φαίνεται στο Σχήμα 6.1(c), δηλαδή, ο κόμβος v είναι παιδί του u και ο z είναι φύλλο και μοναδικό παιδί του v. Καλούμε το σύνολο αυτών των δύο ακμών *κλαδί*. Το σύνολο όλων των κλαδιών από μπλε ακμές με πάνω κόμβο τον u καλείται *μπλοκ*. Με βάση τα Λήμματα 6.6 και 6.7 μπορεί εύκολα να αποδειχθεί ότι:

Λήμμα 6.9 ([Czyzowicz et al., 2007]) Σε οποιοδήποτε BHS-σχήμα, ισχύει το εξής: μια πράσινη ακμή πρέπει να προσπελαστεί από τους πράκτορες τουλάχιστον 2 φορές, μια κόκκινη ακμή πρέπει να προσπελαστεί τουλάχιστον 6 φορές και ένα κλαδί από μπλε ακμές απαιτεί συνολικά τουλάχιστον 6 διασχίσεις.

Αφού σε κάθε BHS-σχήμα, κάθε ένας από τους δύο πράκτορες μπορεί να διασχίσει το πολύ μία ακμή σε μία μονάδα χρόνου, ισχύει το ακόλουθο λήμμα:

Λήμμα 6.10 ([Czyzowicz et al., 2007]) Οποιοδήποτε BHS-σχήμα απαιτεί τουλάχιστον 3, 1 και $3r$ μονάδες χρόνου για τη διάσχιση μιας κόκκινης, μιας πράσινης και ενός μπλοκ από r κλαδιά μπλε ακμών αντίστοιχα.

6.3.3 Ένας βέλτιστος αλγόριθμος για 'θαμνώδη' δέντρα

Θεωρήστε την οικογένεια $\mathcal{T}$ των δέντρων με ρίζα τα οποία έχουν την εξής ιδιότητα: κάθε εσωτερικός κόμβος ενός δέντρου της οικογένειας $\mathcal{T}$ (συμπεριλαμβανομένης της ρίζας) έχει τουλάχιστον 2 παιδιά. Καλούμε τα δέντρα της οικογένειας $\mathcal{T}$ *θαμνώδη*. Σε τέτοια δέντρα, η αναζήτηση της μαύρης τρύπας μπορεί να γίνει με πολύ αποδοτικό τρόπο χρησιμοποιώντας κατάλληλα τη διαδικασία `Split`. Περιγράφουμε παρακάτω τον Αλγόριθμο 31 ο οποίος είναι βέλτιστος για αυτή την κλάση δέντρων.

Έστω T ένα θαμνώδες δέντρο με ρίζα s και έστω u ένας εσωτερικός κόμβος του T. Καλούμε *βαρύτερο παιδί* $v = H(u)$ (αντίστοιχα *ελαφρύτερο* παιδί $v = L(u)$) του u το παιδί v του u που είναι ρίζα του υποδέντρου $T(v)$ με το μέγιστο (αντίστοιχα ελάχιστο) ύψος ανάμεσα από όλα τα τα υποδέντρα με ρίζες τα παιδιά του u. Αν υπάρχουν πάνω από ένα τέτοια υποδέντρα με μέγιστο (αντίστοιχα ελάχιστο) ύψος τότε διαλέγουμε οποιοδήποτε από αυτά. Παρατηρήστε ότι τα $H(u)$ και $L(u)$ μπορούν να υπολογιστούν από τους πράκτορες για κάθε κόμβο u σε γραμμικό χρόνο.

Ακολουθεί η περιγραφή του Αλγόριθμου 31. Έστω m ο κόμβος στον οποίον έχει προγραμματιστεί συνάντηση μεταξύ των δύο πρακτόρων μετά από κάποια φάση (αρχικά $m \equiv s$).

- Εξερεύνησε κάθε ζευγάρι από ανεξερεύνητες ακμές (m, x), (m, y) με πάνω κόμβο τον m εκτελώντας τη διαδικασία `Split(x,y)`, αφήνοντας την ακμή $(m, L(m))$ τελευταία.

- Εάν υπάρχει ανεξερεύνητη ακμή με πάνω κόμβο τον m (η οποία πρέπει να είναι η $(m, L(m))$) τότε ένας από τους πράκτορες διασχίζει αυτήν την ακμή ενώ ο άλλος πράκτορας διασχίζει κάποια άλλη 'κοντινή' ανεξερεύνητη ακμή (αν υπάρχει) χρησιμοποιώντας πάλι τη διαδικασία `Split`. Εάν η ακμή $(m, L(m))$ είναι η τελευταία ανεξερεύνητη ακμή στο δέντρο, διέσχισέ τη εκτελώντας τη διαδικασία `Probe(L(m))`.

- Εάν έχουν εξερευνηθεί όλες οι ακμές των οποίων ο πάνω κόμβος είναι ο m, διέσχισε όπως προηγουμένως, τις ανεξερεύνητες ακμές που προσπίπτουν σε παιδιά του m και σε απόγονους του m.

Αλγόριθμος 31 (2 *πράκτορες με μνήμη και χάρτη σε ένα συγχρονισμένο 'θαμνώδες' δέντρο*)

1: $next := s$;
2: **repeat**
3: $v := next$;
4: **for** κάθε ζεύγος από ανεξερεύνητες ακμές (v, x), (v, y) με πάνω κόμβο τον v **do**
5: Split(x, y), έτσι ώστε η ακμή $(v, L(v))$ εξερευνάται τελευταία;
6: **end for**
7: **if** υπάρχουν ακόμη ανεξερεύνητες ακμές στο δέντρο **then**
8: **case 1**: όλες οι ακμές που προσπίπτουν στο v έχουν εξερευνηθεί:
9: **case 1.1**: υπάρχει τουλάχιστον μία ανεξερεύνητη ακμή που προσπίπτει σε ένα παιδί w του v:
10: $next := w$;
11: **case 1.2**: όλες οι ακμές που προσπίπτουν σε παιδιά του v έχουν εξερευνηθεί:
12: έστω t ο γονιός του v;
13: $next := t$;
14: πήγαινε στον κόμβο $next$;
15: **case 2**: υπάρχει μία ανεξερεύνητη ακμή (v, z) που προσπίπτει στο v:
16: *(* θα πρέπει να ισχύει $z = L(v)$ *)*
17: $next :=$ Explore-only-child(v);
18: **end if**
19: **until** όλες οι ακμές έχουν εξερευνηθεί
20: πήγαινε στον κόμβο s και ανέφερε τη θέση της μαύρης τρύπας;

Η συνάρτηση `Explore-only-child(`v`)` παίρνει σαν είσοδο τον τρέχοντα κόμβο v στον οποίο βρίσκονται και οι δύο πράκτορες και επιστρέφει το νέο σημείο συνάντησης μετά την εξερεύνηση της ακμής $(v, L(v))$. Ακολουθεί η περιγραφή της διαδικασίας:

- Αν υπάρχει ανεξερεύνητη ακμή που προσπίπτει σε ένα παιδί w του v, όπου $w \neq L(v)$, τότε οι πράκτορες εξερευνούν την ακμή $(w, H(w))$ μαζί με την ακμή $(v, L(v))$ ακολουθώντας τη διαδικασία `Split(H(w),L(v))`. Ο νέος κόμβος συνάντησης είναι ο w.

- Αν όλες οι ακμές που προσπίπτουν σε οποιοδήποτε παιδί w του v, διαφορετικό του $L(v)$, έχουν εξερευνηθεί ενώ η ακμή $(v, L(v))$ δεν είναι η τελευταία ανεξερεύνητη ακμή στο δέντρο, τότε βρες τον πιο μακρινό απόγονο a του v που έχει έναν απόγονο στον οποίο προσπίπτει ανεξερεύνητη ακμή (εκτός του $L(v)$). Οι πράκτορες εξερευνούν την ακμή $(D(a), H(D(a)))$ (όπου $D(a)$ είναι ο πιο κοντινός απόγονος του a στον οποίο προσπίπτει ανεξερεύνητη ακμή), μαζί με την ακμή $(v, L(v))$, χρησιμοποιώντας τη διαδικασία `Split(H(D(a)),L(v))`. Ο νέος κόμβος συνάντησης είναι ο $D(a)$.

- Αν η ακμή $(v, L(v))$ είναι η τελευταία ανεξερεύνητη ακμή στο δέντρο τότε εξερεύνησέ την χρησιμοποιώντας τη διαδικασία `Probe(L(v))`. Ο νέος κόμβος συνάντησης είναι ο v.

Παρατηρήστε ότι όλες οι ακμές του δέντρου (εκτός ίσως από την τελευταία αν ο συνολικός αριθμός των ακμών είναι περιττός αριθμός) εξερευνούνται με τη χρήση της διαδικασίας `Split`. Παρατηρήστε επίσης πως σε ένα θαμνώδες δέντρο υπάρχουν μόνο *κόκκινες* και *πράσινες* ακμές.

Μπορεί να αποδειχθεί ότι στο BHS σχήμα που παράγεται από τον Αλγόριθμο 31 κάθε κόκκινη ακμή διασχίζεται συνολικά 6 φορές ενώ κάθε πράσινη ακμή 2 φορές. Επιπλέον, κάθε φάση είναι 2-φάση (δηλαδή σε κάθε φάση διασχίζονται 2 ανεξερεύνητες ακμές από τους πράκτορες), εκτός ίσως από την τελευταία φάση, ενώ κανείς πράκτορας δεν περιμένει σε κάποια 2-φάση. Συνεπώς, με τη χρήση του Λήμματος 6.10 προκύπτει το ακόλουθο θεώρημα:

Θεώρημα 6.11 ([Czyzowicz et al., 2007]) Ο Αλγόριθμος 31 παράγει ένα βέλτιστο BHS-σχήμα για οποιοδήποτε θαμνώδες δέντρο.

6.3.4 Η περίπτωση των γενικών δέντρων

Για γενικά δέντρα υπάρχει ένα απλός αλγόριθμος, ο οποίος εγγυάται ένα λόγο προσέγγισης αυστηρά μικρότερο από 4, δηλαδή, ένας αλγόριθμος που παράγει ένα BHS-σχήμα για οποιοδήποτε δέντρο και κόμβο εκκίνησης του οποίου ο χρόνος είναι λιγότερος από 4 φορές το χρόνο του ταχύτερου BHS-σχήματος για το ίδιο δέντρο και κόμβο εκκίνησης. Ο αλγόριθμος είναι ο εξής: οι δύο πράκτορες διασχίζουν μαζί το δέντρο εκτελώντας αναζήτηση κατά βάθος και εξερευνούν κάθε νέο κόμβο χρησιμοποιώντας τη διαδικασία *probe*. Το BHS-σχήμα που παράγεται εξερευνά ένα δέντρο n κόμβων σε χρόνο (δηλαδή πλήθος διασχίσεων) το πολύ $4(n-1)-2l$, όπου l είναι ο αριθμός των φύλλων στο δέντρο. Σε ένα οποιοδήποτε BHS-σχήμα ενός δέντρου n κόμβων, κάθε ακμή πρέπει να προσπελαστεί τουλάχιστον δύο φορές από κάποιον πράκτορα (όχι κατ' ανάγκην τον ίδιο) το οποίο δίνει συνολικά $2(n-1)$ διασχίσεις. Συνεπώς, απαιτούνται τουλάχιστον $n-1$ βήματα, και αυτό οδηγεί σε έναν παράγοντα προσέγγισης αυστηρά μικρότερο από 4 για τον παραπάνω απλό αλγόριθμο.

Ένας καλύτερος (και επίσης απλός) προσεγγιστικός αλγόριθμος για γενικά δέντρα προτάθηκε στο [Czyzowicz et al., 2007] πετυχαίνοντας έναν προσεγγιστικό λόγο $\frac{5}{3}$. Ακολουθεί μια σύντομη περιγραφή του αλγόριθμου αυτού. Έστω v ο κόμβος συνάντησης των δύο πρακτόρων μετά από μια φάση (αρχικά $v \equiv s$). Οι ακμές με πάνω κόμβο τον v εξερευνούνται με τη χρήση της διαδικασίας

`Split` είτε μέχρι όλες αυτές οι ακμές να εξερευνηθούν, είτε μέχρι να υπάρχει το πολύ μία ανεξερεύνητη ακμή που προσπίπτει στο v, η οποία εξερευνάται καλώντας τη Διαδικασία `Probe`. Αυτό επαναλαμβάνεται για κάθε παιδί του v.

Το αν είναι δυνατή η παραγωγή ενός βέλτιστου BHS-σχήματος για οποιοδήποτε (γενικό) δέντρο σε πολυωνυμικό χρόνο ή αν το πρόβλημα είναι NP-hard, παραμένει ένα ανοιχτό ερώτημα.

6.4 Σχόλια και βιβλιογραφικές παρατηρήσεις

Οι περισσότεροι αλγόριθμοι που παρουσιάστηκαν σε αυτό το κεφάλαιο για το πρόβλημα της αναζήτησης μαύρης τρύπας βασίζονται στην τεχνική της προσεκτικής ανίχνευσης. Ωστόσο, όταν το δίκτυο είναι κατευθυνόμενο αυτή η τεχνική δεν μπορεί να εφαρμοστεί. Σε κατευθυνόμενα δίκτυα (συγχρονισμένα και ασύγχρονα) το πρόβλημα έχει μελετηθεί ([Czyzowicz et al., 2009, Kosowski et al., 2009]) για πράκτορες που ξεκινούν στον ίδιο κόμβο, έχουν μνήμη αλλά δεν έχουν χάρτη του δικτύου και μπορούν να γράφουν σε πίνακες που υπάρχουν στους κόμβους. Στο άρθρο [Czyzowicz et al., 2009] αποδεικνύεται πως σε κατευθυνόμενα ασύγχρονα δίκτυα με πίνακες στους κόμβους, απαιτούνται 2^{Δ} πράκτορες στη χειρότερη περίπτωση ενώ $\Delta + 1$ πράκτορες χωρίς χάρτη αρκούν σε μη-κατευθυνόμενα δίκτυα, όπου Δ είναι το πλήθος των εισερχόμενων ακμών στον κόμβο που βρίσκεται η μαύρη τρύπα. Έτσι βλέπουμε πως η αδυναμία εφαρμογής της προσεκτικής ανίχνευσης στα κατευθυνόμενα γραφήματα οδηγεί σε εκθετική αύξηση του απαιτούμενου αριθμού πρακτόρων. Το πρώτο κάτω φράγμα ισχύει ακόμη και στην περίπτωση συγχρονισμένων δικτύων. Σε πιο ειδικές περιπτώσεις κατευθυνόμενων δικτύων, όπως τα επίπεδα κατευθυνόμενα δίκτυα με μια εμφύτευση στο επίπεδο που είναι γνωστή στους πράκτορες, απαιτούνται 2Δ πράκτορες ενώ $2\Delta + 1$ πράκτορες αρκούν. Σε συγχρονισμένα κατευθυνόμενα δίκτυα με πίνακες στους κόμβους αποδείχθηκε στο άρθρο [Kosowski et al., 2009] ότι $O(\Delta \cdot 2^{\Delta})$ πράκτορες αρκούν για να λύσουν το πρόβλημα.

Βιβλιογραφία

[Asaka et al., 1999] Asaka, M., Okazawa, S., Taguchi, A., and Goto, S. (1999). A method of tracing intruders by use of mobile agents. In *Proceedings of 9th Annual Conference of the Internet Society*. (http://www.isoc.org/).

[Balamohan et al., 2012] Balamohan, B., Dobrev, S., Flocchini, P., and Santoro, N. (2012). Asynchronous exploration of an unknown anonymous dangerous graph with o(1) pebbles. In *Proc. of 19th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 7355, pages 279–290.

[Balamohan et al., 2010] Balamohan, B., Flocchini, P., Miri, A., and Santoro, N. (2010). Time optimal algorithms for black hole search in rings. In *4th Annual Int. Conference on Combinatorial Optimization and Applications*, LNCS 6509, pages 58–71.

[Balamohan et al., 2011a] Balamohan, B., Flocchini, P., Miri, A., and Santoro, N. (2011a). Improving the optimal bounds for black hole search in rings. In *Proc. of 18th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 6796, pages 198–209.

[Balamohan et al., 2011b] Balamohan, B., Flocchini, P., Miri, A., and Santoro, N. (2011b). Time optimal algorithms for black hole search in rings. *Discrete Mathematics, Algorithms and Applications*, 3(4):1–15.

[Barriere et al., 2002] Barriere, L., Flocchini, P., Fraigniaud, P., and Santoro, N. (2002). Capture of an intruder by mobile agents. In *Proceedings of 14th ACM Symposium on Parallel Algorithms and Architectures*, pages 200–209.

[Borselius, 2002] Borselius, N. (2002). Mobile agent security. *Electronics and Communication Engineering Journal*, 14(5):211–218.

[Chalopin et al., 2011a] Chalopin, J., Das, S., Labourel, A., and Markou, E. (2011a). Black hole search with finite automata scattered in a synchronous torus. In *Proc. of 25th Int. Symp. on Distributed Computing*, LNCS 6950, pages 432–446.

[Chalopin et al., 2011b] Chalopin, J., Das, S., Labourel, A., and Markou, E. (2011b). Tight bounds for scattered black hole search in a ring. In *Proc. of 18th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 6796, pages 186–197.

[Chalopin et al., 2007] Chalopin, J., Das, S., and Santoro, N. (2007). Rendezvous of mobile agents in unknown graphs with faulty links. In *Proceedings of 21st International Conference on Distributed Computing*, pages 108–122.

[Chess, 1998] Chess, D. M. (1998). Security issues in mobile code systems. In *Proceedings of Conference on Mobile Agent Security*, LNCS 1419, pages 1–14.

[Cooper et al., 2006] Cooper, C., Klasing, R., and Radzik, T. (2006). Searching for black-hole faults in a network using multiple agents. In *Proceedings of 10th International Conference on Principles of Distributed Systems*, LNCS 4305, pages 320–332.

[Cooper et al., 2010] Cooper, C., Klasing, R., and Radzik, T. (2010). Locating and repairing faults in a network with mobile agents. *Theoretical Computer Science*, 411(14-15):1638–1647.

[Czyzowicz et al., 2009] Czyzowicz, J., Dobrev, S., Kralovic, R., Miklik, S., and Pardubska, D. (2009). Black hole search in directed graphs. In *Proc. of 16th Int. Colloquium on Structural Information and Communication Complexity*, pages 182–194.

[Czyzowicz et al., 2004] Czyzowicz, J., Kowalski, D., Markou, E., and Pelc, A. (2004). Searching for a black hole in tree networks. In *Proc. of 8th International Conference on Principles of Distributed Systems*, LNCS 3544, pages 67–80.

[Czyzowicz et al., 2006] Czyzowicz, J., Kowalski, D., Markou, E., and Pelc, A. (2006). Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2-3):229–242.

[Czyzowicz et al., 2007] Czyzowicz, J., Kowalski, D., Markou, E., and Pelc, A. (2007). Searching for a black hole in synchronous tree networks. *Combinatorics, Probability & Computing*, 16(4):595–619.

[Dobrev et al., 2006a] Dobrev, S., Flocchini, P., Kralovic, R., Prencipe, G., Ruzicka, P., and Santoro, N. (2006a). Optimal search for a black hole in common interconnection networks. *Networks*, 47(2):61–71.

[Dobrev et al., 2006b] Dobrev, S., Flocchini, P., Kralovic, R., and Santoro, N. (2006b). Exploring a dangerous unknown graph using tokens. In *Proceedings of 5th IFIP International Conference on Theoretical Computer Science*, pages 131–150.

[Dobrev et al., 2001] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. (2001). Mobile agents search for a black-hole in an anonymous ring. In *Proceedings of 15th International Symposium on Distributed Computing*, pages 166–179.

[Dobrev et al., 2003] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. (2003). Multiple agents rendezvous in a ring in spite of a black hole. In *Proceedings of 6th International Conference on Principles of Distributed Systems*, pages 34–46.

[Dobrev et al., 2006c] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. (2006c). Searching for a black hole in arbitrary networks: Optimal mobile agents protocols. *Distributed Computing*, 19(1):1–19.

[Dobrev et al., 2007a] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. (2007a). Mobile search for a black hole in an anonymous ring. *Algorithmica*, 48:67–90.

[Dobrev et al., 2004] Dobrev, S., Flocchini, P., and Santoro, N. (2004). Improved bounds for optimal black hole search in a network with a map. In *Proceedings of 10th International Colloquium on Structural Information and Communication Complexity*, pages 111–122.

[Dobrev et al., 2006d] Dobrev, S., Kralovic, R., Santoro, N., and Shi, W. (2006d). Black hole search in asynchronous rings using tokens. In *Proceedings of 6th Conference on Algorithms and Complexity*, pages 139–150.

[Dobrev et al., 2007b] Dobrev, S., Santoro, N., and Shi, W. (2007b). Scattered black hole search in an oriented ring using tokens. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, pages 1–8.

[Dobrev et al., 2008] Dobrev, S., Santoro, N., and Shi, W. (2008). Using scattered mobile agents to locate a black hole in an un-oriened ring with tokens. *International Journal of Foundations of Computer Science*, 19(6):1355–1372.

[Flocchini et al., 2005] Flocchini, P., Huang, M. J., and Luccio, F. L. (2005). Contiguous search in the hypercube for capturing an intruder. In *Proceedings of 18th IEEE International Parallel and Distributed Processing Symposium*.

[Flocchini et al., 2006] Flocchini, P., Huang, M. J., and Luccio, F. L. (2006). Decontamination of chordal rings and tori. In *Proceedings of 8th Workshop on Advances in Parallel and Distributed Computational Models*.

[Flocchini et al., 2008] Flocchini, P., Ilcinkas, D., and Santoro, N. (2008). Ping pong in dangerous graphs: Optimal black hole search with pure tokens. In *Proceedings of 22nd International Symposium on Distributed Computing*, LNCS 5218, pages 227–241.

[Flocchini et al., 2012a] Flocchini, P., Ilcinkas, D., and Santoro, N. (2012a). Ping pong in dangerous graphs: Optimal black hole search with pebbles. *Algorithmica*, 62(3-4):1006–1033.

[Flocchini et al., 2009] Flocchini, P., Kellett, M., Mason, P., and Santoro, N. (2009). Map construction and exploration by mobile agents scattered in a dangerous network. In *Proceedings of IEEE International Symposium on Parallel & Distributed Processing*, pages 1–10.

[Flocchini et al., 2012b] Flocchini, P., Kellett, M., Mason, P., and Santoro, N. (2012b). Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184.

[Flocchini et al., 1998] Flocchini, P., Mans, B., and Santoro, N. (1998). Sense of direction: Definitions, properties, and classes. *Networks*, 32(3):165–180.

[Flocchini et al., 2003] Flocchini, P., Mans, B., and Santoro, N. (2003). Sense of direction in distributed computing. *Theoretical Computer Science*, 291:29–53.

[Flocchini and Santoro, 2012] Flocchini, P. and Santoro, N. (2012). *Mobile Agents in Networking and Distributed Computing (Eds. J. Sao and S. Das)*, chapter Distributed Security Algorithms for Mobile Agents. Wiley.

[Foukia et al., 2001] Foukia, N., Hulaas, J. G., and Harms, J. (2001). Intrusion detection with mobile agents. In *Proceedings of 11th Annual Conference of the Internet Society*.

[Glaus, 2009] Glaus, P. (2009). Locating a black hole without the knowledge of incoming link. In *Algorithmic Aspects of Wireless Sensor Networks*, pages 128–138.

[Greenberg et al., 1998] Greenberg, M. S., Byington, J. C., and Harper, D. G. (1998). Mobile agents and security. *IEEE Commun. Mag.*, 36(7):76–85.

[Hohl, 1998] Hohl, F. (1998). Time limited blackbox security: Protecting mobile agents from malicious hosts. In *Proceedings of Conference on Mobile Agent Security*, LNCS 1419, pages 92–113.

[Hohl, 2000] Hohl, F. (2000). A framework to protect mobile agents by using reference states. In *Proceedings of 20th International Conference on Distributed Computing Systems*, pages 410–417.

[Jansen, 2000] Jansen, W. (2000). Countermeasures for mobile agent security. *Computer Communications*, 23(17):1667–1676.

[Klasing et al., 2007] Klasing, R., Markou, E., Radzik, T., and Sarracco, F. (2007). Hardness and approximation results for black hole search in arbitrary graphs. *Theoretical Computer Science*, 384(2-3):201–221.

[Klasing et al., 2008] Klasing, R., Markou, E., Radzik, T., and Sarracco, F. (2008). Approximation bounds for black hole search problems. *Networks*, 52(4):216–226.

[Kosowski et al., 2009] Kosowski, A., Navarra, A., and Pinotti, C. (2009). Synchronization helps robots to detect black holes in directed graphs. In *Proc. of 13th Int. Conf. on Principles of Distributed Systems*, pages 86–98.

[Luccio et al., 2006] Luccio, F., Pagli, L., and Santoro, N. (2006). Network decontamination with local immunization. In *Proceedings of 8th Workshop on Advances in Parallel and Distributed Computational Models*.

[Ng and Cheung, 1999] Ng, S. and Cheung, K. (1999). Protecting mobile agents against malicious hosts by intention of spreading. In *Proceedings of International Conference on Parallel and Distributed Processing and Applications*, pages 725–729.

[Oppliger, 1999] Oppliger, R. (1999). Security issues related to mobile code and agent-based systems. *Computer Communications*, 22(12):1165–1170.

[Sander and Tschudin, 1998] Sander, T. and Tschudin, C. F. (1998). Protecting mobile agents against malicious hosts. In *Proceedings of Conference on Mobile Agent Security*, LNCS 1419, pages 44–60.

[Schelderup and Ølnes, 1999] Schelderup, K. and Ølnes, J. (1999). Mobile agent security — issues and directions. In *Proceedings of 6th International Conference on Intelligence and Services in Networks*, LNCS 1597, pages 155–167.

[Shi, 2009] Shi, W. (2009). Black hole search with tokens in interconnected networks. In *Proc. of 11th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems*, pages 670–682.

[Spafford and Zamboni, 2000] Spafford, E. H. and Zamboni, D. (2000). Intrusion detection using autonomous agents. *Computer Networks*, 34(4):547–570.

[Vitek and Castagna, 1999] Vitek, J. and Castagna, G. (1999). Mobile computations and hostile hosts. In Tsichritzis, D., editor, *Mobile Objects*, pages 241–261. University of Geneva.

ΚΕΦΑΛΑΙΟ 7

Εχθρικοί Κόμβοι σε Γραφήματα και Πράκτορες Χωρίς Μνήμη

Σε αυτό το κεφάλαιο συζητούμε το πρόβλημα της ανακάλυψης εχθρικών κόμβων σε γενικά γραφήματα. Περιγράφουμε επίσης αλγόριθμους για πεπερασμένα αυτόματα χωρίς μνήμη σε δακτύλιους και τορικά δίκτυα καθώς και προσεγγιστικούς αλγόριθμους για μηχανές Turing. Τέλος, συζητούμε μοντέλα εχθρικών κόμβων μεγάλης δύναμης (byzantine black holes) και κάνουμε μια κατηγοριοποίηση των δυνατοτήτων των εχθρικών κόμβων. Πιο συγκεκριμένα, στην ενότητα 7.1 μελετάμε το πρόβλημα της αναζήτησης μιας μαύρης τρύπας σε συγχρονισμένα γενικά γραφήματα. Δείχνουμε ότι το πρόβλημα της βέλτιστης αναζήτησης μιας μαύρης τρύπας είναι NP-hard σε γενικά γραφήματα. Δίνουμε προσεγγιστικούς αλγόριθμους για γενικά γραφήματα και αποδεικνύουμε ότι το πρόβλημα δεν μπορεί να προσεγγιστεί περισσότερο από έναν σταθερό παράγοντα σε πολυωνυμικό χρόνο, εκτός εάν $P = NP$ (δηλαδή είναι APX-hard). Στην ενότητα 7.2 μελετάμε το πρόβλημα των διάσπαρτων πρακτόρων που έχουν μόνο σταθερή μνήμη (DFAs) και μεταφέρουν αγνά σημάδια. Συζητάμε τα αποτελέσματα σε τοπολογίες δακτυλίου και τόρου. Τέλος, στην ενότητα 7.3 παρουσιάζουμε περιληπτικά κάποια άλλα μοντέλα προβλημάτων αναγνώρισης εχθρικών κόμβων.

7.1 Εχθρικοί Κόμβοι σε Συγχρονισμένα Γραφήματα

Η εύρεση ενός βέλτιστου BHS σχήματος για γενικά γραφήματα έχει αποδειχθεί ότι είναι NP-hard ([Klasing et al., 2005, Klasing et al., 2007]). Αυτό το αποτέλεσμα αποδείχθηκε επεκτείνοντας την αναγωγή που χρησιμοποιήθηκε για να δείξει ότι μια πιο γενική παραλλαγή του προβλήματος είναι NP-hard. Σε αυτήν την παραλλαγή αρχικά δίνεται ένα *σύνολο* ασφαλών κόμβων (αντί για τον κόμβο εκκίνησης) και παρουσιάστηκε στο άρθρο [Czyzowicz et al., 2006]. Αργότερα αποδείχθηκε ([Klasing et al., 2008]) ότι το πρόβλημα είναι APX-hard και στις δύο παραλλαγές για γενικά γραφήματα. Σε αυτήν την ενότητα περιγράφουμε αρχικά την αναγωγή που δείχνει ότι το πρόβλημα

στην παραλλαγή με το σύνολο ασφαλών κόμβων είναι NP-hard. Στη συνέχεια συζητούμε την περίπτωση του προβλήματος με τον ένα ασφαλή κόμβο. Τέλος περιγράφουμε τα αποτελέσματα μη-προσεγγισιμότητας για τις δύο παραλλαγές.

7.1.1 Η γενική περίπτωση του προβλήματος της Μαύρης Τρύπας

Στο άρθρο [Czyzowicz et al., 2006] μελετήθηκε το πρόβλημα της μαύρης τρύπας σε συγχρονισμένα γενικά γραφήματα στα οποία αρχικά δίνεται ένα σύνολο ασφαλών κόμβων το οποίο περιέχει τον κόμβο εκκίνησης, ενώ η μαύρη τρύπα βρίσκεται σε έναν από τους υπόλοιπους κόμβους. Ονομάζουμε αυτήν την παραλλαγή του προβλήματος *general Black Hole Search (gBHS)*. Αποδείχθηκε ότι η εύρεση ενός βέλτιστου σχήματος αναζήτησης για δύο πράκτορες σε ένα γενικό γράφημα είναι NP-hard. Παρουσιάστηκε επίσης ένας προσεγγιστικός αλγόριθμος ο οποίος, δοθέντος ενός γραφήματος, ενός συνόλου ασφαλών κόμβων και ενός κόμβου εκκίνησης, παράγει σε πολυωνυμικό χρόνο ένα σχήμα αναζήτησης για δύο πράκτορες το οποίο επιτυγχάνει την εύρεση της μαύρης τρύπας στο δοσμένο γράφημα σε χρόνο το πολύ 9.3 φορές περισσότερο από ότι ο χρόνος ενός βέλτιστου σχήματος για την ίδια είσοδο.

Μοντέλο και ορολογία

Το μοντέλο που χρησιμοποιήθηκε έχει τις εξής διαφορές σε σχέση με το μοντέλο του ενός ασφαλούς κόμβου. Αρχικά δίνονται:

- ένα γράφημα G και ένας κόμβος εκκίνησης s,
- ένα υποσύνολο S των κόμβων του γραφήματος στο οποίο περιλαμβάνεται ο κόμβος s. Οι κόμβοι του συνόλου S είναι ασφαλείς (δηλαδή το S δεν περιλαμβάνει τον κόμβο με τη μαύρη τρύπα).

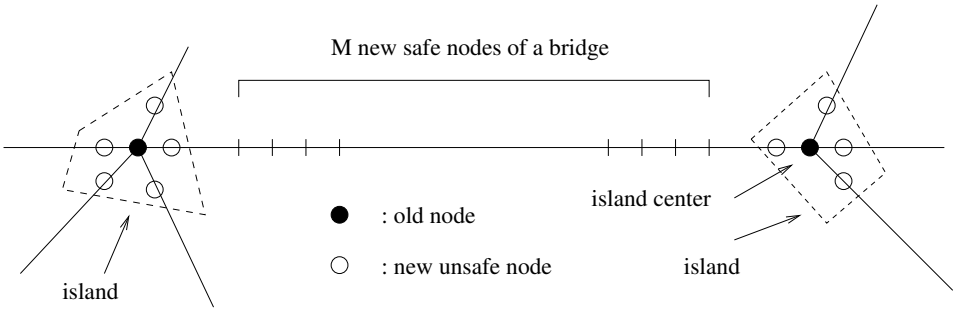
Θεωρούμε ότι υπάρχει το πολύ μία μαύρη τρύπα στο δίκτυο και ο σκοπός είναι να βρούμε ένα βέλτιστο gBHS σχήμα αναζήτησης για την είσοδο (G, S, s).

Το πρόβλημα general black hole search είναι NP-hard

Η απόδειξη χρησιμοποιεί μια αναγωγή από το πρόβλημα του κύκλου Hamilton (το οποίο είναι NP-complete) στο πρόβλημα απόφασης του προβλήματος gBHS (το οποίο καλείται dgBHS).

Το πρόβλημα του κύκλου Hamilton (HC):
Στιγμιότυπο: Ένα γράφημα G.
Ερώτηση: Υπάρχει κύκλος Hamilton στο G;

Το πρόβλημα dgBHS:
Στιγμιότυπο: Ένα γράφημα G', ένα υποσύνολο S ασφαλών κόμβων, ένας κόμβος εκκίνησης $s \in S$ και ένας θετικός ακέραιος X.



Σχήμα 7.1: Η κατασκευή ενός στιγμιότυπου του προβλήματος dgBHS

Ερώτηση: Υπάρχει ένα gBHS σχήμα για την είσοδο (G', S, s), το οποίο βρίσκει τη μαύρη τρύπα μετά από χρόνο το πολύ X;

Κατασκευή του στιγμιότυπου του προβλήματος dgBHS

Έστω ένα γράφημα G με n κόμβους και e ακμές το οποίο αποτελεί ένα στιγμιότυπο του προβλήματος HC. Κατασκευάζουμε ένα νέο γράφημα G' ως εξής. Καλούμε παλιούς κόμβους τους κόμβους του γραφήματος G. Σε κάθε ακμή του G προσθέτουμε 2 νέους μη-ασφαλείς κόμβους τους οποίους συνδέουμε με τα άκρα της ακμής και $M = 4e + 5n - 1$ νέους ασφαλείς κόμβους ανάμεσα στους νέους κόμβους, όπως στο Σχήμα 7.1. Επιλέγουμε ως κόμβο εκκίνησης s οποιονδήποτε από τους παλιούς n κόμβους. Όλοι οι παλιοί κόμβοι εκτός του s θεωρούνται μη-ασφαλείς. Συνεπώς το σύνολο όλων των μη-ασφαλών κόμβων αποτελείται από όλους τους παλιούς κόμβους εκτός του s και από όλους τους νέους κόμβους που γειτονεύουν με παλιούς κόμβους.

Το στιγμιότυπο του προβλήματος dgBHS είναι το γράφημα G' με $n' = n + (M + 2)e$ κόμβους, το σύνολο S με $Me + 1$ ασφαλείς κόμβους, ο κόμβος εκκίνησης s, και ο ακέραιος $X = M(n + 1) - 1$.

Η κατασκευή του στιγμιότυπου αυτού με δεδομένο το γράφημα G γίνεται σε πολυωνυμικό χρόνο. Αν υπάρχει ένας κύκλος Hamilton C στο αρχικό γράφημα G τότε οι δύο πράκτορες μπορούν να ακολουθήσουν αυτόν τον κύκλο ξεκινώντας στον κόμβο s και να ανακαλύψουν τη μαύρη τρύπα (ή να ανακαλύψουν ότι δεν υπάρχει μαύρη τρύπα) στο νέο γράφημα G' (το οποίο κατασκευάζεται όπως περιγράφηκε παραπάνω) σε χρόνο το πολύ $M(n + 1) - 1$ ως εξής:

Εξερευνούν με τη χρήση της διαδικασίας Probe που ορίστηκε στο προηγούμενο κεφάλαιο (δηλαδή ο ένας από τους πράκτορες πηγαίνει σε κάποιον γειτονικό κόμβο ενώ ο άλλος περιμένει) τους κόμβους που είναι γειτονικοί με το κέντρο ενός νησιού (βλέπε Σχήμα 7.1) εκτός από τους δύο κόμβους που είναι στο C και επιστρέφουν στο κέντρο του νησιού. Έπειτα εξερευνούν με τη χρήση της διαδικασίας Probe τον μη-ασφαλή κόμβο της γέφυρας στο C στην κατεύθυνση που επιλέγουν και προχωρούν στη γέφυρα, μέχρι να φτάσουν στον τελευταίο της κόμβο. Έπειτα εξερευνούν τον γειτονικό μη-ασφαλή κόμβο v (στο επόμενο νησί) με τη χρήση της διαδικασίας probe, φτάνουν

σε αυτόν, εξερευνούν το κέντρο του νησιού και πηγαίνουν εκεί. Επαναλαμβάνουν την παραπάνω διαδικασία σε κάθε νησί του C μέχρι που φτάνουν ξανά στον κόμβο s.

Αν δεν υπάρχει κύκλος Hamilton στο γράφημα G τότε οι πράκτορες χρειάζονται τουλάχιστον $M(n+1)$ χρόνο για να ανακαλύψουν τη μαύρη τρύπα (ή να ανακαλύψουν ότι δεν υπάρχει μαύρη τρύπα) στο G' (βλέπε [Czyzowicz et al., 2006] για περισσότερες λεπτομέρειες).

Θεώρημα 7.1 ([Czyzowicz et al., 2006]) Η κατασκευή ενός βέλτιστου BHS σχήματος για το πρόβλημα gBHS σε γενικά γραφήματα είναι NP-hard.

Ένας προσεγγιστικός αλγόριθμος για το πρόβλημα gBHS

Ένας προσεγγιστικός αλγόριθμος (Αλγόριθμος 32) για το πρόβλημα gBHS σε γενικά γραφήματα παρουσιάστηκε επίσης στο άρθρο [Czyzowicz et al., 2006]. Ο αλγόριθμος βασίζεται στην κατασκευή ενός *Steiner* δέντρου του γραφήματος εισόδου G, το οποίο δέντρο περιέχει (εκτός των άλλων κόμβων) τους μη-ασφαλείς κόμβους του G και τον κόμβο εκκίνησης s οι οποίοι καλούνται *απαιτούμενοι* κόμβοι. Θυμίζουμε ότι ένα δέντρο Steiner ενός γραφήματος $G=(V,E)$ με σύνολο απαιτούμενων κόμβων $R \subseteq V$ είναι οποιοδήποτε υποδέντρο του G που περιέχει το R. Μπορούμε να κατασκευάσουμε ένα τέτοιο δέντρο Steiner T σε πολυωνυμικό χρόνο με προσεγγιστικό παράγοντα α, όπου $\alpha = 1 + \frac{\ln 3}{2} < 1.55$ ([Kann,], [Robins and Zelikovsky, 2000]). Πιο συγκεκριμένα, αν x είναι το πλήθος των μη-ασφαλών κόμβων του G συν τον κόμβο s, και y είναι το πλήθος των ασφαλών κόμβων του T (εκτός του κόμβου s), ενώ y^* είναι το ελάχιστο πλήθος των ασφαλών κόμβων (εκτός του s) που απαιτούνται για το βέλτιστο δέντρο Steiner, τότε $(x+y) \leq 1.55(x+y^*)$.

Ακολουθεί η περιγραφή του Αλγόριθμου 32. Αρχικά κατασκευάζεται ένα δέντρο Steiner όπως περιγράφηκε παραπάνω. Έστω v ο κόμβος συνάντησης των δύο πρακτόρων μετά από μια φάση (αρχικά $v \equiv s$). Τα ανεξερεύνητα παιδιά του v εξερευνούνται με τη χρήση της διαδικασίας probe. Αν υπάρχει τουλάχιστον ένας ανεξερεύνητος κόμβος γειτονικός με ένα παιδί του v τότε οι πράκτορες πηγαίνουν εκεί και επαναλαμβάνουν τη διαδικασία. Αλλιώς οι πράκτορες πηγαίνουν στον πατέρα του v και επαναλαμβάνουν τη διαδικασία.

Η χρονική πολυπλοκότητα του Αλγόριθμου 32 είναι πολυωνυμική ως προς το μέγεθος του G και είναι ανάλογη του χρόνου κατασκευής του δέντρου Steiner. Ο αλγόριθμος 32 είναι στην ουσία ένας αλγόριθμος αναζήτησης κατά βάθος με μόνη διαφορά ότι η επίσκεψη κάθε μη-ασφαλούς κόμβου γίνεται προσεκτικά (με τη χρήση της διαδικασίας Probe). Ο χρόνος που δαπανάται στη διάσχιση οποιασδήποτε ακμής (u,v) (v είναι παιδί του u) του δέντρου T είναι το πολύ 4 βήματα: η χειρότερη περίπτωση εμφανίζεται όταν η ακμή (u,v) οδηγεί σε έναν ανεξερεύνητο κόμβο v ο οποίος δεν είναι φύλλο του T, συνεπώς οι πράκτορες δαπανούν 2 βήματα για να επισκεφτούν το v, 1 βήμα για να μετακινηθούν στο v και άλλο ένα βήμα για να επιστρέψουν στον κόμβο u - μετά την εξερεύνηση των απογόνων του v. Ο συνολικός χρόνος που απαιτείται από το gBHS-σχήμα που παράγεται από τον Αλγόριθμο 32 είναι λιγότερος από $4(x+y)$.

Λήμμα 7.2 ([Czyzowicz et al., 2006]) Οποιοδήποτε gBHS-σχήμα για το γράφημα G απαιτεί τουλάχιστον $\frac{4}{3}(x+y^*)$ διασχίσεις ακμών.

Αλγόριθμος 32 (*Ένας προσεγγιστικός αλγόριθμος για το πρόβλημα gBHS σε γενικά γραφήματα*)

1: κατασκεύασε ένα βέλτιστο δέντρο Steiner T που περιέχει όλους τους μη-ασφαλείς κόμβους και τον κόμβο s;
2: $next := s$;
3: **repeat**
4: $v := next$;
5: **for** κάθε ανεξερεύνητο κόμβο z γειτονικό του v **do**
6: probe(z);
7: **end for**
8: **if** υπάρχουν ακόμη ανεξερεύνητοι κόμβοι **then**
9: **case 1**: υπάρχει τουλάχιστον ένας ανεξερεύνητος κόμβος γειτονικός με ένα παιδί w του v:
10: $next := w$;
11: **case 2**: κάθε κόμβος γειτονικός με οποιοδήποτε παιδί του v έχει εξερευνηθεί:
12: έστω t ο πατέρας του v;
13: $next := t$;
14: μετακινήσου στον κόμβο $next$;
15: **end if**
16: **until** όλοι οι κόμβοι έχουν ερευνηθεί
17: μετακινήσου στον κόμβο s και ανέφερε τη θέση της μαύρης τρύπας;

Το κάτω φράγμα που προκύπτει από το Λήμμα 7.2 μαζί με το άνω φράγμα που επιτυγχάνεται από τον Αλγόριθμο 32 καθώς και ο προσεγγιστικός παράγοντας α (< 1.55) της εύρεσης ενός ελάχιστου δέντρου Steiner μας δίνουν το επόμενο θεώρημα:

Θεώρημα 7.3 ([Czyzowicz et al., 2006]) Ο Αλγόριθμος 32 είναι ένας προσεγγιστικός αλγόριθμος για το πρόβλημα gBHS με παράγοντα προσέγγισης $6\alpha < 9.3$.

Ο παράγοντας προσέγγισης για το πρόβλημα gBHS βελτιώθηκε στο άρθρο [Klasing et al., 2008] όπου δόθηκε ένας αλγόριθμος με παράγοντα προσέγγισης 6 χρησιμοποιώντας πάλι ελάχιστα συνδετικά δέντρα.

7.1.2 Το πρόβλημα BHS σε γενικά γραφήματα

Στο άρθρο [Klasing et al., 2005] (δες επίσης το πλήρες άρθρο στο [Klasing et al., 2007]) αποδείχθηκε πως το πρόβλημα της αναζήτησης μιας μαύρης τρύπας από δύο πράκτορες σε ελάχιστο χρόνο, όταν μόνο ο κόμβος εκκίνησης ειναι αρχικά ασφαλής, είναι NP-hard ακόμη και σε επίπεδα γραφήματα.

Το πρόβλημα της αναζήτησης μιας μαύρης τρύπας σε επίπεδα γραφήματα

Η απόδειξη πως το πρόβλημα σε επίπεδα γραφήματα (BHSp) είναι NP-hard έγινε με μια αναγωγή από μια παραλλαγή του προβλήματος του κύκλου Hamilton. Η αναγωγή έγινε από το πρόβλημα του κύκλου Hamilton για επίπεδα γραφήματα βαθμού 3 (cpHC)) στο πρόβλημα απόφασης BHSp.
Ακολουθεί ο τυπικός ορισμός των δύο προβλημάτων:

Το πρόβλημα cpHC

Στιγμιότυπο: ένα επίπεδο 2-συνδεδεμένο γράφημα $G = (V, E)$ βαθμού 3, και μια ακμή $(x, y) \in E$.

Ερώτηση: Υπάρχει ένας κύκλος Hamilton στο G ο οποίος περιλαμβάνει την ακμή (x, y);

Το πρόβλημα dBHSp

Στιγμιότυπο: ένα επίπεδο γράφημα $G' = (V', E')$, ένας κόμβος εκκίνησης $s \in V'$, και ένας θετικός ακέραιος X.

Ερώτηση: υπάρχει ένα BHS-σχήμα για το G' που ξεκινά από τον κόμβο s και χρειάζεται χρόνο το πολύ X;

Το πρόβλημα cpHC αποδείχθηκε πως είναι NP-complete στο άρθρο [Klasing et al., 2007] με μια απλή αναγωγή από το πρόβλημα του κύκλου Hamilton σε επίπεδα γραφήματα βαθμού 3 (το οποίο είχε αποδειχθεί ότι είναι NP-complete στο [Garey et al., 1976]). Στη συνέχεια οι συγγραφείς παρουσιάζουν μια αρκετά τεχνική κατασκευή η οποία ανάγει το πρόβλημα cpHC στο πρόβλημα dBHSp αποδεικνύοντας πως το τελευταίο είναι NP-complete.

Θεώρημα 7.4 ([Klasing et al., 2007]) Ένα γράφημα G με n κόμβους έχει έναν κύκλο Hamilton που περνά από μια ακμή (x, y) αν και μόνο αν υπάρχει ένα BHS-σχήμα για το γράφημα G' με κόμβο εκκίνησης s (ο οποίος κατασκευάζεται από το G σε πολυωνυμικό χρόνο) με χρόνο αναζήτησης το πολύ $5n + 2$ βήματα.

Ένας προσεγγιστικός αλγόριθμος για το πρόβλημα BHS σε γενικά γραφήματα

Μια μέθοδος για την σχεδίαση ενός προσεγγιστικού αλγόριθμου για το πρόβλημα BHS σε ένα γενικό γράφημα G, είναι η ακόλουθη: Βρες ένα συνδετικό δέντρο του G και εξερεύνησε το γράφημα διασχίζοντας τις ακμές του δέντρου. Αφού οποιοδήποτε BHS-σχήμα ενός γραφήματος n κόμβων απαιτεί χρόνο τουλάχιστον $n - 1$, αυτή η μέθοδος μαζί με τον απλό αλγόριθμο που περιγράφηκε στο τέλος της ενότητας 6.3 μας δίνει έναν προσεγγιστικό παράγοντα 4 για οποιοδήποτε γράφημα. Ακολουθώντας αυτή τη μέθοδο, για να πάρουμε καλύτερο αποτέλεσμα χρειαζόμαστε έναν αλγόριθμο που να κατασκευάζει καλά BHS-σχήματα για δέντρα και έναν αλγόριθμο που να κατασκευάζει συνδετικά δέντρα που είναι κατάλληλα για αυτά τα σχήματα. Ένας αλγόριθμος γραμμικού χρόνου ο οποίος επεκτείνει την κατασκευή του βέλτιστου BHS-σχήματος για *θαμνώδη* δέντρα (η οποία παρουσιάστηκε στην ενότητα 6.3) σε γενικά δέντρα προτάθηκε στο [Klasing et al., 2007]

και δίνει παράγοντα προσέγγισης $3\frac{3}{8}$. Ο αλγόριθμος δεν εγγυάται την εύρεση βέλτιστων σχημάτων αναζήτησης για δέντρα που δεν είναι θαμνώδη. Το πρόβλημα της εύρεσης σε πολυωνυμικό χρόνο, βέλτιστων σχημάτων αναζήτησης για γενικά δέντρα παραμένει ανοικτό.

7.1.3 Αποτελέσματα μη-Προσεγγισιμότητας

Στο άρθρο [Klasing et al., 2008] και οι δύο παραλλαγές του προβλήματος της αναζήτησης μαύρης τρύπας που παρουσιάστηκαν σε αυτό το κεφάλαιο αποδείχθηκε ότι είναι APX-hard. Αποδείχθηκε πως δεν μπορούμε να προσεγγίσουμε σε πολυωνυμικό χρόνο ένα βέλτιστο BHS-σχήμα για γενικά γραφήματα (στην παραλλαγή με το δεδομένο σύνολο ασφαλών κόμβων) με παράγοντα προσέγγισης $1+\varepsilon$ για οποιοδήποτε $\varepsilon < \frac{1}{388}$, εκτός αν $P = NP$. Αποδείχθηκε επίσης πως η παραλλαγή του προβλήματος στην οποία μόνο ένας κόμβος (ο κόμβος εκκίνησης) είναι αρχικά ασφαλής, είναι επίσης APX-hard.

Οι συγγραφείς δίνουν ένα κάτω όριο προσεγγισιμότητας του γενικού προβλήματος αναζήτησης μιας μαύρης τρύπας παρουσιάζοντας μια αναγωγή που διατηρεί την προσεγγισιμότητα από μια ειδική περίπτωση του προβλήματος του Πλανόδιου Πωλητή (Traveling Salesman Problem - TSP), η οποία έχει παρουσιαστεί στο [Engebretsen and Karpinski, 2006]. Σε αυτήν την ειδική περίπτωση του TSP, οι αποστάσεις μεταξύ των κόμβων είναι συμμετρικές και ικανοποιούν την τριγωνική ανισότητα, ενώ η μέγιστη απόσταση M είναι μια σταθερά.

Λήμμα 7.5 [Engebretsen and Karpinski, 2006] Η προσέγγιση του TSP(1,M) με παράγοντα $1+\varepsilon$ είναι NP-hard πρόβλημα για οποιοδήποτε $\varepsilon < \frac{1}{388}$.

Η μέθοδος της απόδειξης της μη-προσεγγισιμότητας του προβλήματος gBHS έχει ως εξής. Ένα οποιοδήποτε στιγμιότυπο (G, d) του TSP ανάγεται σε ένα στιγμιότυπο (G', S, s) του προβλήματος gBHS και αποδεικνύεται πως αν η βέλτιστη λύση του gBHS για το συγκεκριμένο κατασκευασμένο στιγμιότυπο μπορεί να προσεγγιστεί με παράγοντα $(1+\varepsilon)$, τότε η βέλτιστη λύση του αντίστοιχου στιγμιότυπου του TSP μπορεί να προσεγγιστεί με τον ίδιο παράγοντα.

Θεώρημα 7.6 ([Klasing et al., 2008]) Το πρόβλημα gBHS δεν είναι προσεγγίσιμο σε πολυωνυμικό χρόνο με παράγοντα $1+\varepsilon$ για οποιοδήποτε $\varepsilon < \frac{1}{388}$, εκτός αν $P = NP$.

Παρουσιάζουν επίσης μια αναγωγή από το TSP όπου οι αποστάσεις μεταξύ των κόμβων είναι 1 ή 2 στο πρόβλημα BHS με μόνο ένα αρχικά ασφαλή κόμβο:

Θεώρημα 7.7 ([Klasing et al., 2008]) Το πρόβλημα BHS με ένα αρχικά ασφαλή κόμβο δεν είναι προσεγγίσιμο με παράγοντα $1+\varepsilon$ για οποιοδήποτε $\varepsilon < \frac{1}{2258}$ εκτός αν $P = NP$.

7.2 Διασκορπισμένοι πράκτορες με σταθερή μνήμη

Στις προηγούμενες ενότητες του κεφαλαίου καθώς και στο προηγούμενο κεφάλαιο, το πρόβλημα της αναζήτησης μαύρης τρύπας μελετήθηκε για πράκτορες με μνήμη που ξεκινούν στον ίδιο κόμβο.

Η μνήμη είναι μια σημαντική δυνατότητα που επιτρέπει στους πράκτορες να φτιάχνουν και να αποθηκεύουν ένα χάρτη του δικτύου, να κρατούν πληροφορίες ή να μετρούν τους κόμβους του δικτύου που επισκέπτονται και έτσι τους βοηθά αποφασιστικά στην εξερεύνηση του δικτύου.

Σε ασύγχρονα δίκτυα, η αρχική τοποθέτηση των πρακτόρων στον ίδιο κόμβο εκκίνησης στο μοντέλο με πίνακες στους κόμβους τους δίνει τη δυνατότητα να παίρνουν διαφορετικές ταυτότητες και έτσι να αναθέτουν διαφορετικές ετικέτες στους κόμβους του δικτύου. Σε συγχρονισμένα δίκτυα τους δίνει τη δυνατότητα να τα εξερευνούν χρησιμοποιώντας τη διαδικασία *Probe* και να ανακαλύπτουν έτσι τη μαύρη τρύπα.

Έτσι είναι φυσιολογικό να ρωτήσει κανείς αν η επίλυση του προβλήματος είναι δυνατή σε πιο αδύναμα μοντέλα. Υπάρχουν αποτελέσματα για διασκορπισμένους πράκτορες (δηλαδή πράκτορες που δεν ξεκινούν στον ίδιο κόμβο) οι οποίοι μπορούν να αφήνουν σημάδια στους κόμβους (αντί να χρησιμοποιούν πίνακες) και δεν έχουν χάρτη του ασύγχρονου δικτύου. Η μνήμη των πρακτόρων όμως δεν μπορεί να περιοριστεί καθώς σε τέτοια δίκτυα οι πράκτορες χρειάζονται να ξέρουν και να μπορούν να αποθηκεύσουν το συνολικό πλήθος των κόμβων (ή έστω ένα πάνω φράγμα). Αντιθέτως, σε συγχρονισμένα δίκτυα, γενικώς δεν απαιτείται η γνώση τέτοιου φράγματος και συνεπώς μια ενδιαφέρουσα ερώτηση είναι αν είναι δυνατόν να επιλυθεί το πρόβλημα από *ντετερμινιστικά πεπερασμένα αυτόματα (DFAs)* (που έχουν δηλαδή μόνο σταθερή μνήμη), και είναι αρχικά διασκορπισμένα στο δίκτυο, χωρίς χάρτη και έχοντας μόνο ένα σταθερό αριθμό σημαδιών που μπορούν να αφήσουν στους κόμβους του δικτύου.

Σημειώνουμε εδώ πως ο *H. A. Rollik* ([Rollik, 1980]) έχει αποδείξει ότι δεν υπάρχει σταθερό πλήθος (οσοδήποτε μεγάλο και αν είναι) από πεπερασμένα αυτόματα που να μπορεί να εξερευνήσει όλα τα επίπεδα γραφήματα βαθμού 3. Δεδομένου ότι ένα πεπερασμένο αυτόματο είναι πιο ισχυρό από ένα σημάδι, αυτό σημαίνει ότι δεν υπάρχει σταθερό πλήθος από πεπερασμένα αυτόματα ακόμη και αν χρησιμοποιούν οποιοδήποτε σταθερό αριθμό σημαδιών που να μπορούν να εξερευνήσουν όλα τα επίπεδα γραφήματα βαθμού 3, ακόμη και όταν οι πράκτορες ξεκινούν από τον ίδιο κόμβο και μπορούν να ανταλλάσσουν πληροφορίες, όταν συναντιούνται σε κόμβους. Είναι σαφές ότι αν οι πράκτορες δεν μπορούν να εξερευνήσουν το γράφημα (δηλαδή, να επισκέπτονται όλους τους κόμβους) δεν μπορούν να λύσουν το πρόβλημα BHS. Ως εκ τούτου ένα ενδιαφέρον ερώτημα είναι αν υπάρχουν συγχρονισμένες τοπολογίες δικτύων για τις οποίες το πρόβλημα λύνεται σε ένα τέτοιο αδύναμο μοντέλο. Ακόμα και όταν οι πράκτορες μπορούν να επικοινωνούν όταν συναντιούνται σε έναν κόμβο, αφού αρχικά είναι διάσπαρτοι στο γράφημα χρειάζεται πρώτα να λύσουν το πρόβλημα της συνάντησης το οποίο δεν είναι καθόλου εύκολο για ανώνυμα DFAs που μεταφέρουν πανομοιότυπα σημάδια.

Το πρόβλημα BHS μελετήθηκε σχετικά πρόσφατα σε αυτό το μοντέλο για τοπολογίες δακτυλίου ([Chalopin et al., 2011b]) και τόρου ([Chalopin et al., 2011a]). Δηλαδή χρησιμοποιώντας αρχικά διασκορπισμένους ανώνυμους πράκτορες με σταθερή μνήμη που μπορούν να μεταφέρουν μόνο ένα σταθερό αριθμό από πανομοιότυπα σημάδια και έχουν τη δυνατότητα να επικοινωνούν μόνο όταν συναντηθούν στον ίδιο κόμβο.

Καθώς το μοντέλο τώρα είναι αρκετά διαφορετικό, δίνουμε παρακάτω περισσότερες λεπτομέρειες.

7.2.1 Το μοντέλο

Το μοντέλο αποτελείται από ένα ανώνυμο, συγχρονισμένο δίκτυο με $k \geq 2$ πανομοιότυπους κινητούς πράκτορες που αρχικά βρίσκονται σε διαφορετικούς κόμβους (οι οποίοι καλούνται *βάσεις*). Κάθε κινητός πράκτορας έχει ένα σταθερό αριθμό t πανομοιότυπων σημαδιών, τα οποία μπορούν να τοποθετηθούν σε κάθε κόμβο που επισκέπτεται ο πράκτορας. Όλα τα σημάδια είναι ίδια. Κάθε σημάδι ή πράκτορας σε ένα δεδομένο κόμβο είναι ορατός μόνο από τους πράκτορες που βρίσκονται σε εκείνον τον κόμβο. Οι πράκτορες εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο ξεκινώντας την ίδια στιγμή και βρίσκονται στην ίδια αρχική κατάσταση.

Ένα σημάδι καλείται *μετακινήσιμο* αν μπορεί να τοποθετηθεί σε κάποιον κόμβο και να μετακινηθεί αργότερα από κάποιον πράκτορα σε άλλον κόμβο. Αλλιώς το σημάδι καλείται *μη-μετακινήσιμο* ή *αμετακίνητο* εννοώντας ότι αν τοποθετηθεί κάπου δεν μπορεί να μετακινηθεί.

Τυπικά κάθε πράκτορας είναι ένα πεπερασμένο αυτόματο Moore. Όλοι οι υπολογισμοί που γίνονται από τους πράκτορες είναι ανεξάρτητοι (οι πράκτορες δεν έχουν καμία γνώση) του μεγέθους του δικτύου. Οι αλγόριθμοι για την τοπολογία δακτυλίου δουλεύουν, ακόμη και χωρίς τη γνώση του αριθμού των πρακτόρων. Υπάρχει ακριβώς μία μαύρη τρύπα στο δίκτυο. Ένας πράκτορας μπορεί να ξεκινήσει από οποιονδήποτε κόμβο, εκτός από τη μαύρη τρύπα και οι πράκτορες δεν ξεκινούν από τον ίδιο κόμβο εκκίνησης. Μόλις ένας πράκτορας ανιχνεύει μια ακμή που προσπίπτει στη μαύρη τρύπα, μαρκάρει μόνιμα αυτή την ακμή ως επικίνδυνη. Ο στόχος είναι, στο τέλος της αναζήτησης, όλες οι ακμές που προσπίπτουν στη μαύρη τρύπα (και μόνο αυτές) να είναι σημειωμένες ως επικίνδυνες και να υπάρχει τουλάχιστον ένας επιζών πράκτορας. Παρατηρήστε ότι αυτός ο ορισμός της επιτυχούς αναζήτησης είναι ελαφρώς διαφορετικός από τον συνηθισμένο ορισμό. Πράγματι, στο συνηθισμένο ορισμό, απαιτείται να υπάρχει τουλάχιστον ένας επιζών πράκτορας, και αυτός ο πράκτορας πρέπει να γνωρίζει όλες τις ακμές που προσπίπτουν στη μαύρη τρύπα. Ωστόσο, αυτό είναι αδύνατο σε αυτό το μοντέλο, δεδομένου ότι οι πράκτορες έχουν μόνο σταθερή μνήμη.

7.2.2 Η τοπολογία δακτυλίου

Για την τοπολογία δακτυλίου τέσσερα διαφορετικά σενάρια εξετάστηκαν στο άρθρο [Chalopin et al., 2011b] ανάλογα με το αν τα σημάδια είναι μετακινήσιμα ή όχι, και αν οι πράκτορες συμφωνούν στον προσανατολισμό του δακτυλίου. Αποτελεί έκπληξη το γεγονός ότι το θέμα

		Αναγκαία και ικανά αγαθά	
Σημάδια	**Δακτύλιος**	**# πράκτορες**	**# σημάδια**
Μετακινήσιμα	Προσανατολισμένος	3	1
	Χωρίς προσανατολισμό		
Μη-μετακινήσιμα	Προσανατολισμένος	4	2
	Χωρίς προσανατολισμό	5	2

Σχήμα 7.2: Αποτελέσματα για το πρόβλημα BHS με DFAs και σημάδια σε συγχρονισμένους δακτύλιους

της συμφωνίας στον προσανατολισμό του δακτυλίου δεν επηρεάζει τον αριθμό των πρακτόρων που απαιτούνται στην περίπτωση που τα σημάδια είναι μετακινήσιμα, αλλά είναι σημαντικό στην περίπτωση μη-μετακινήσιμων σημαδιών.

Τα κάτω όρια που παρουσιάζονται στο [Chalopin et al., 2011b] είναι πολύ ισχυρά, με την έννοια ότι δεν επιτρέπουν καμία αυξομείωση μεταξύ του αριθμού των πρακτόρων και του πλήθους των σημαδιών για την επίλυση του προβλήματος BHS. Ειδικότερα, αποδείχθηκε ότι:

- Ένας οποιοσδήποτε σταθερός αριθμός πρακτόρων, ακόμα και με απεριόριστη μνήμη, δεν μπορεί να λύσει το πρόβλημα BHS με λιγότερα σημάδια από εκείνα που απεικονίζονται στις περιπτώσεις του Πίνακα 7.2

- Ένας αριθμός πρακτόρων μικρότερος από εκείνον που απεικονίζεται στις περιπτώσεις του Πίνακα 7.2 δεν μπορεί να λύσει το πρόβλημα ακόμη και αν οι πράκτορες διαθέτουν οποιοδήποτε αριθμό από σημάδια και έχουν απεριόριστη μνήμη.

Τα πάνω όρια που δίνουν οι αλγόριθμοι ταιριάζουν με τα κάτω όρια για τον αριθμό των σημαδιών, είναι ασυμπτωτικά βέλτιστοι σε χρόνο και δεδομένου ότι δεν απαιτούν καμία γνώση του μεγέθους του δακτυλίου ή του αριθμού των πρακτόρων, δουλεύουν σε κάθε ανώνυμο συγχρονισμένο δακτύλιο, για οποιοδήποτε αριθμό ανώνυμων πανομοιότυπων πρακτόρων (που πληρούν τις ελάχιστες απαιτήσεις του Πίνακα 7.2).

Αποτελέσματα μη-επιλυσιμότητας

Περιγράφουμε εδώ κάποια αρνητικά αποτελέσματα, αρχικά για δακτύλιους με προσανατολισμό και στη συνέχεια για δακτύλιους χωρίς προσανατολισμό.

Θεωρήστε ότι οι πράκτορες συμφωνούν ως προς τον προσανατολισμό του δακτυλίου (δηλαδή, οι κατευθύνσεις στις ακμές είναι επισημασμένες ‘αριστερά’ και ‘δεξιά’, με συνέπεια). Όταν τα σημάδια δεν μπορούν να μετακινηθούν, μια ομάδα από οποιονδήποτε σταθερό αριθμό πρακτόρων χρειάζεται τουλάχιστον δύο σημάδια ανά πράκτορα για την επίλυση του προβλήματος BHS. Αυτό οφείλεται στο γεγονός ότι με ένα μόνο μη-μετακινήσιμο σημάδι οι πράκτορες είναι υποχρεωμένοι να παίρνουν πάντα τις ίδιες αποφάσεις και να κάνουν πάντα τις ίδιες ενέργειες συμπερι-

λαμβανομένης της σήμανσης (λανθασμένα σε διαφορετικού μεγέθους δαχτυλίδια) των ακμών ως επικίνδυνων.

Θεώρημα 7.8 ([Chalopin et al., 2011b]) Για οποιαδήποτε σταθερά k, δεν υπάρχει αλγόριθμος που λύνει το πρόβλημα BHS σε όλα τα προσανατολισμένα δακτυλίδια που περιέχουν μία μαύρη τρύπα και k ή περισσότερους διασκορπισμένους πράκτορες, όταν κάθε πράκτορας διαθέτει μόνο ένα μη-μετακινήσιμο σημάδι. Αυτό το αποτέλεσμα ισχύει ακόμη και αν οι πράκτορες έχουν απεριόριστη μνήμη.

Για τη λύση του προβλήματος BHS σε δακτύλιους, οι δύο ακμές που προσπίπτουν στη μαύρη τρύπα πρέπει να μαρκαριστούν ως επικίνδυνες. Συνεπώς προκύπτει άμεσα το ακόλουθο αποτέλεσμα.

Θεώρημα 7.9 ([Chalopin et al., 2011b]) Δύο πράκτορες που μεταφέρουν οποιονδήποτε αριθμό από μετακινήσιμα σημάδια δεν μπορούν να λύσουν το πρόβλημα BHS σε έναν προσανατολισμένο δακτύλιο, ακόμη και αν οι πράκτορες έχουν απεριόριστη μνήμη.

Όταν τα σημάδια είναι μη-μετακινήσιμα, ακόμη και τρεις πράκτορες δεν αρκούν για την επίλυση του προβλήματος BHS, όπως φαίνεται παρακάτω. Αυτό οφείλεται στα ακόλουθα γεγονότα: α) λόγω του σταθερού αριθμού σημαδιών οι πράκτορες δεν μπορούν να εγγυηθούν ότι θα αφήσουν ένα σημάδι σε έναν κόμβο που είναι γειτονικός με τη μαύρη τρύπα, και β) αφού ένας πράκτορας χαθεί στη μαύρη τρύπα, οι υπόλοιποι δύο πράκτορες θα μπορούσαν, ενδεχομένως, να συναντηθούν (αξιοποιώντας την ασυμμετρία που δημιούργησε ο πράκτορας που εξαφανίστηκε), αλλά δεν μπορούν να ανακαλύψουν τις δύο ακμές που προσπίπτουν στη μαύρη τρύπα.

Θεώρημα 7.10 ([Chalopin et al., 2011b]) Τρεις πράκτορες που μεταφέρουν οποιονδήποτε σταθερό αριθμό από μη-μετακινήσιμα σημάδια δεν μπορούν να λύσουν το πρόβλημα BHS σε έναν προσανατολισμένο δακτύλιο, ακόμη και αν οι πράκτορες έχουν απεριόριστη μνήμη.

Σε ένα μη-προσανατολισμένο δακτύλιο, ακόμα και τέσσερις πράκτορες δεν αρκούν για να λύσουν το πρόβλημα BHS με σημάδια που δεν μετακινούνται, δεδομένου ότι δύο από αυτούς μπορούν να εξαφανιστούν στη μαύρη τρύπα ταυτόχρονα αφήνοντας τα σημάδια τους περισσότερο από μια σταθερή απόσταση μακριά από τη μαύρη τρύπα και έτσι οι υπόλοιποι δύο πράκτορες δεν είναι δυνατόν να επισημάνουν σωστά τις ακμές που προσπίπτουν στη μαύρη τρύπα.

Θεώρημα 7.11 ([Chalopin et al., 2011b]) Σε έναν δακτύλιο χωρίς προσανατολισμό, τέσσερις πράκτορες που μεταφέρουν οποιονδήποτε σταθερό αριθμό από μη-μετακινήσιμα σημάδια, δεν μπορούν να επισημάνουν σωστά τις ακμές που προσπίπτουν στη μαύρη τρύπα, ακόμη και αν οι πράκτορες έχουν απεριόριστη μνήμη.

Ένας αλγόριθμος με μετακινήσιμα σημάδια

Αν κάθε πράκτορας έχει ένα μετακινήσιμο σημάδι μπορεί να επισκέπτεται προσεκτικά νέους ανεξερεύνητους κόμβους με την εξής διαδικασία, την οποία καλούμε *CautiousWalk()*. Τοποθετεί το σημάδι στον κόμβο που βρίσκεται, προχωρά ένα βήμα προς κάποια κατεύθυνση `dir` (η οποία δίνεται ως είσοδος στη διαδικασία) και αμέσως μετά επιστρέφει για να πάρει το σημάδι που είχε αφήσει. Τέλος προχωρά ένα βήμα προς την κατεύθυνση `dir` (μεταφέροντας το σημάδι).

Αποδεικνύουμε ότι τρεις πράκτορες αρκούν για την επίλυση του προβλήματος, όταν έχουν από ένα μετακινήσιμο σημάδι. Ο αλγόριθμος 33 λειτουργεί για δακτύλιους με ή χωρίς προσανατολισμό.

Αλγόριθμος 33 (*Τρία DFAs με ένα μετακινήσιμο σημάδι σε συγχρονισμένους δακτύλιους*)

```
1: repeat
2:     CautiousWalk(Left)
3: until συναντάς έναν κόμβο με σημάδι αλλά χωρίς πράκτορα OR η επόμενη ακμή είναι επισημασμένη σαν Dangerous
4: MarkLink(Left)
5: repeat
6:     CautiousWalk(Right)
7: until συναντάς έναν κόμβο με σημάδι αλλά χωρίς πράκτορα OR η επόμενη ακμή είναι επισημασμένη σαν Dangerous
8: MarkLink(Right)
```

Θεώρημα 7.12 ([Chalopin et al., 2011b]) Ο Αλγόριθμος 33 λύνει το πρόβλημα BHS σε δακτύλιους χωρίς προσανατολισμό με $k \geq 3$ πράκτορες που έχουν σταθερή μνήμη και από ένα μετακινήσιμο σημάδι.

Αλγόριθμοι με αμετακίνητα σημάδια

Όταν οι πράκτορες έχουν στη διάθεσή τους μόνο αμετακίνητα σημάδια, χρησιμοποιούμε τη διαδικασία *Probe* για την εξερεύνηση νέων κόμβων. Τώρα όμως για να χρησιμοποιηθεί αυτή η τεχνική, θα πρέπει δύο πράκτορες να συναντηθούν στον ίδιο κόμβο και μάλιστα με τρόπο που να υπάρχει κάποια ασυμμετρία μεταξύ τους, έτσι ώστε να μπορούν να τους αποδοθούν διαφορετικοί ρόλοι. Αυτή είναι η κύρια δυσκολία για την οποία φροντίζουν οι αλγόριθμοι που ακολουθούν. Η βασική ιδέα των αλγορίθμων που παρουσιάζονται είναι η εξής. Αρχικά εντοπίζονται οι δύο βάσεις πρακτόρων που βρίσκονται πιο κοντά στη μαύρη τρύπα (μία σε κάθε πλευρά του δακτυλίου). Καλούμε αυτές τις βάσεις *πύλες*. Οι πύλες χωρίζουν το δαχτυλίδι σε δύο τμήματα. Το ένα τμήμα περιέχει την μαύρη τρύπα (και συνεπώς είναι επικίνδυνο). Το άλλο τμήμα περιέχει όλες τις βάσεις (και είναι ασφαλές). Αρχικά όλοι οι πράκτορες είναι στο ασφαλές τμήμα και ένας πράκτορας μπορεί να κινηθεί προς το επικίνδυνο τμήμα μόνο περνώντας μέσα από κάποια πύλη. Ο αλγόριθμος

εξασφαλίζει ότι οποιοσδήποτε πράκτορας φτάσει σε έναν κόμβο πύλη, περιμένει για έναν ακόμη πράκτορα, προκειμένου να εκτελέσει την ανιχνευτική διαδικασία (Probe). Παρουσιάζουμε τώρα δύο αλγόριθμους, έναν για προσανατολισμένους δακτύλιους και έναν για δακτύλιους χωρίς προσανατολισμό.

Προσανατολισμένοι δακτύλιοι

Παρουσιάζουμε τον Αλγόριθμο `BHS-Ring-2` που χρησιμοποιεί τουλάχιστον τέσσερις πράκτορες με δύο αμετακίνητα σημάδια.

Περιγραφή του Αλγόριθμου `BHS-Ring-2`:

Κατά τη διάρκεια της πρώτης φάσης του αλγόριθμου κάθε πράκτορας τοποθετεί ένα σημάδι στη βάση του, κινείται αριστερά μέχρι την επόμενη βάση (δηλαδή, τον επόμενο κόμβο με σημάδι) και στη συνέχεια επιστρέφει στη βάση του και τοποθετεί το δεύτερο σημάδι. Κατά τη διάρκεια αυτής της φάσης ένας πράκτορας θα χαθεί στη μαύρη τρύπα και θα υπάρξει ακριβώς μία βάση με ένα μόνο σημάδι (καλούμε αυτόν τον κόμβο *πύλη*) ενώ όλες οι άλλες βάσεις θα περιέχουν τελικά ακριβώς δύο σημάδια. Ωστόσο, οι πράκτορες μπορεί να μην ολοκληρώσουν αυτήν τη φάση του αλγορίθμου ταυτόχρονα. Έτσι, κατά τη διάρκεια του αλγόριθμου, μπορεί να υπάρχουν πολλές βάσεις που περιέχουν ένα μόνο σημάδι. Κάθε φορά που ένας πράκτορας φτάνει σε έναν κόμβο που περιέχει μόνο ένα σημάδι, περιμένει έναν πράκτορα συνεργάτη και κατόπιν ανιχνεύουν (χρησιμοποιώντας τη διαδικασία Probe) προς την αριστερή κατεύθυνση. Ένας από τους πράκτορες ενός ζεύγους θα χαθεί τελικά στη μαύρη τρύπα και τότε ο άλλος πράκτορας σηματοδοτεί την ακμή που οδηγεί στη μαύρη τρύπα και επιστρέφει στον κόμβο πύλη, περιμένοντας για έναν άλλον πράκτορα. Όταν ένας άλλος πράκτορας φτάνει σε αυτόν τον κόμβο, αυτοί οι δύο πράκτορες ανιχνεύουν (χρησιμοποιώντας τη διαδικασία Probe) προς την αντίθετη κατεύθυνση για να βρουν την άλλη ακμή που προσπίπτει στη μαύρη τρύπα. ■

Θεώρημα 7.13 ([Chalopin et al., 2011b]) Ο Αλγόριθμος BHS-Ring-2 ανακαλύπτει τη μαύρη τρύπα σε οποιονδήποτε προσανατολισμένο δακτύλιο χρησιμοποιώντας 4 ή περισσότερους πράκτορες που έχουν σταθερή μνήμη και μεταφέρουν δύο αμετακίνητα σημάδια ο καθένας.

Δακτύλιοι χωρίς προσανατολισμό

Σε δακτύλιους χωρίς προσανατολισμό, απαιτούνται τουλάχιστον 5 πράκτορες με δύο αμετακίνητα σημάδια. Ο Αλγόριθμος `BHS-Ring-3` για μη-προσανατολισμένους δακτύλιους, είναι παρόμοιος με εκείνον για προσανατολισμένους δακτύλιους, με τη διαφορά ότι κάθε πράκτορας επιλέγει έναν προσανατολισμό. Όταν δύο πράκτορες συναντιούνται μπορούν να συμφωνήσουν για τον προσανατολισμό και να αναλάβουν διαφορετικούς ρόλους.

Περιγραφή του Αλγόριθμου `BHS-Ring-3`:

Κάθε πράκτορας τοποθετεί ένα σημάδι στη βάση του, και κινείται αριστερά της μέχρι να βρει κάποιο άλλο σημάδι. Στη συνέχεια, επιστρέφει στη βάση του. Τώρα ο πράκτορας κινείται

προς τα δεξιά μέχρι να βρει ένα σημάδι και στη συνέχεια επιστρέφει και πάλι στη βάση του, όπου τοποθετεί το δεύτερο σημάδι. Κατά τη διάρκεια αυτής της φάσης ακριβώς δύο πράκτορες θα χαθούν στη μαύρη τρύπα. Κάθε πράκτορας που επιζεί κινείται προς τα αριστερά μέχρι να βρει έναν κόμβο u με ένα μόνο σημάδι. Σε αυτό το σημείο ο πράκτορας πρέπει να περιμένει, δεδομένου ότι είτε υπάρχει μια μαύρη τρύπα μπροστά, είτε ο κόμβος u είναι η βάση ενός πράκτορα b, που δεν έχει ακόμη επιστρέψει για να τοποθετήσει το δεύτερό του σημάδι. Αφού υπάρχουν τουλάχιστον πέντε πράκτορες, τουλάχιστον δύο από αυτούς θα συναντηθούν σε μια πύλη. Εντοπίζουν μια από τις ακμές που προσπίπτει στη μαύρη τρύπα και στη συνέχεια ο πράκτορας που απομένει συναντά έναν άλλο πράκτορα που περιμένει και προσδιορίζουν μαζί την άλλη ακμή που προσπίπτει στη μαύρη τρύπα. ■

Θεώρημα 7.14 ([Chalopin et al., 2011b]) Ο Αλγόριθμος BHS-Ring-3 ανακαλύπτει τη μαύρη τρύπα σε οποιονδήποτε δακτύλιο χωρίς προσανατολισμό με 5 ή περισσότερους πράκτορες που έχουν σταθερή μνήμη και μεταφέρουν δύο αμετακίνητα σημάδια ο καθένας.

7.2.3 Η τοπολογία Τόρου

Ενώ η εξερεύνηση ενός δακτυλίου με ασφαλείς κόμβους (ακόμη και χωρίς προσανατολισμό) είναι δυνατή από ένα DFA με ένα αμετακίνητο σημάδι, η εξερεύνηση ενός ασφαλούς τόρου δεν είναι πάντα δυνατή από ένα DFA που μεταφέρει σημάδια.

Αποτελέσματα μη-επιλυσιμότητας σε προσανατολισμένο τόρο

Θεώρημα 7.15 ([Chalopin et al., 2011a]) Αν k, t είναι δύο σταθεροί αριθμοί, δεν υπάρχει αλγόριθμος που να ανακαλύπτει τη μαύρη τρύπα σε όλα τα προσανατολισμένα τορικά δίκτυα που περιέχουν μια μαύρη τρύπα και k διασκορπισμένους πράκτορες, όπου κάθε πράκτορας έχει σταθερή μνήμη και t αμετακίνητα σημάδια.

Η ιδέα της απόδειξης του Θεωρήματος 7.15 είναι η εξής: Αποδεικνύεται ότι (με δεδομένη τη συνάρτηση μετάβασης των πρακτόρων) είναι πάντα δυνατή η επιλογή ενός αρκετά μεγάλου τόρου και η αρχική τοποθέτηση των πρακτόρων έτσι ώστε κανένας πράκτορας να μην επισκέπτεται κόμβους που περιέχουν σημάδια τα οποία έχουν τοποθετηθεί από άλλους πράκτορες, και κανείς πράκτορας να μην συναντιέται με άλλον πράκτορα. Επιπλέον, υπάρχουν κόμβοι στον τόρο που δεν επισκέπτεται ποτέ κανείς πράκτορας. Συνεπώς σε περίπτωση που η μαύρη τρύπα βρίσκεται σε έναν κόμβο που δεν τον επισκέπτονται οι πράκτορες δεν είναι δυνατή η επίλυση του προβλήματος.

Όταν τα σημάδια είναι μετακινήσιμα, η κατάσταση είναι αρκετά διαφορετική αφού τώρα κάθε προσανατολισμένος (ασφαλής) τόρος μπορεί να εξερευνηθεί ακόμη και από ένα DFA που μεταφέρει ένα μετακινήσιμο σημάδι. Συνεπώς, οποιοδήποτε κάτω φράγμα με μετακινήσιμα σημάδια δεν μπορεί να βασίζεται στην αδυναμία της εξερεύνησης (δηλαδή της επίσκεψης όλων των κόμβων του τόρου). Μια σχετικά εύκολη παρατήρηση είναι ότι δύο πράκτορες που μεταφέρουν έναν οποιονδήποτε αριθμό από μετακινήσιμα σημάδια δεν μπορούν να λύσουν το πρόβλημα σε έναν

προσανατολισμένο τόρο, ακόμη και αν οι πράκτορες έχουν απεριόριστη μνήμη διότι είναι πάντα δυνατή η αρχική τοποθέτηση των πρακτόρων και της μαύρης τρύπας με τέτοιο τρόπο, ώστε ένας από τους πράκτορες να εξαφανίζεται στη μαύρη τρύπα ενώ αλλάζει κάθετο δακτύλιο και ο άλλος κατά την αλλαγή οριζόντιου δακτύλιου. Επιπλέον, αποδεικνύεται το ακόλουθο κάτω φράγμα:

Λήμμα 7.16 ([Chalopin et al., 2011a]) Δεν υπάρχει αλγόριθμος που να λύνει το πρόβλημα BHS σε όλους τους προσανατολισμένους τόρους με τρεις πράκτορες που έχουν σταθερή μνήμη και από ένα μετακινήσιμο σημάδι.

Η ιδέα της απόδειξης είναι ότι τουλάχιστον δύο από τους τρεις πράκτορες αναγκάζονται να τοποθετήσουν τα σημάδια τους περισσότερο από ένα σταθερό αριθμό κόμβων μακριά (αλλιώς δεν είναι σε θέση να εξερευνήσουν τον τόρο) από τη μαύρη τρύπα, πριν εξαφανιστούν και στη συνέχεια ο τρίτος πράκτορας δεν μπορεί να αποφασίσει για τη σωστή θέση της μαύρης τρύπας.

Συνεπώς προκύπτει το επόμενο θεώρημα:

Θεώρημα 7.17 ([Chalopin et al., 2011a]) Απαιτούνται τουλάχιστον τρεις πράκτορες για την επίλυση του προβλήματος BHS σε όλους τους προσανατολισμένους τόρους. Οποιοσδήποτε αλγόριθμος που λύνει το πρόβλημα χρησιμοποιώντας τρεις πράκτορες απαιτεί τουλάχιστον δύο μετακινήσιμα σημάδια ανά πράκτορα.

Αλγόριθμοι για προσανατολισμένους τόρους με μετακινήσιμα σημάδια

Με δεδομένα τα αποτελέσματα μη-επιλυσιμότητας που αναφέρθηκαν παραπάνω, οποιοσδήποτε αλγόριθμος για το πρόβλημα με DFAs θα πρέπει να χρησιμοποιεί μετακινήσιμα σημάδια. Παρακάτω περιγράφουμε έναν αλγόριθμο με τρεις πράκτορες που μεταφέρουν τρία μετακινήσιμα σημάδια ο καθένας.

Αλγόριθμος `BHS-torus-33`

Κάθε πράκτορας εξερευνά έναν οριζόντιο δακτύλιο πηγαίνοντας ανατολικά και στη συνέχεια κινείται νότια και εξερευνά τον επόμενο οριζόντιο δακτύλιο. Κατά την εξερεύνηση ενός οριζόντιου δακτύλιου, ο πράκτορας αφήνει ένα σημάδι στον κόμβο εκκίνησης. Αυτός ο κόμβος ονομάζεται *βάση* του πράκτορα και το σημάδι που αφήνει εκεί ονομάζεται σημάδι βάσης. Το σημάδι αυτό θα χρησιμοποιηθεί από τον πράκτορα για να αποφασίσει πότε να προχωρήσει στον επόμενο οριζόντιο δακτύλιο. Ο πράκτορας χρησιμοποιεί τα δύο υπόλοιπα σημάδια για να ανιχνεύσει τον οριζόντιο δακτύλιο μέχρι να συναντήσει δύο φορές έναν κόμβο που περιέχει ένα σημάδι. Κάθε κόμβος που περιέχει ακριβώς ένα σημάδι είναι βάση κάποιου πράκτορα. Ο πράκτορας μετακινείται στον επόμενο οριζόντιο δακτύλιο, αφού συναντήσει δύο βάσεις. Στη συνέχεια επαναλαμβάνεται η ίδια διαδικασία εξερεύνησης για αυτό το νέο δακτύλιο. Κάθε φορά που ο πράκτορας βλέπει δύο ή τρία σημάδια στο τέλος μιας ανίχνευσης, έχει εντοπιστεί η θέση της μαύρης τρύπας. Αν συναντήσει δύο (τρία) σημάδια σε κάποιον κόμβο, η μαύρη τρύπα βρίσκεται στο γειτονικό κόμβο w προς τα ανατολικά

(αντίστοιχα νότια). Τότε ο πράκτορας διασχίζει έναν κύκλο γύρω από τον κόμβο w, και επισημαίνει όλες τις ακμές που προσπίπτουν στο w ως επικίνδυνες. ■

Θεώρημα 7.18 ([Chalopin et al., 2011a]) Ο Αλγόριθμος BHS-torus-33 λύνει το πρόβλημα BHS χρησιμοποιώντας 3 ή περισσότερους πράκτορες με τρία σημάδια.

Ένας αλγόριθμος που χρησιμοποιεί μια παρόμοια τεχνική μπορεί να λύσει το πρόβλημα με τέσσερις πράκτορες και δύο σημάδια, ενώ ένας πιο τεχνικός αλγόριθμος (που χρησιμοποιεί μια τεχνική που οδηγεί τους πράκτορες σε συνάντηση όταν είναι αρκετά κοντά) επιτυγχάνει το κάτω φράγμα, δηλαδή τρεις πράκτορες και δύο σημάδια (περισσότερες λεπτομέρειες σχετικά με αυτούς τους αλγόριθμους μπορούν να βρεθούν στο [Chalopin et al., 2011a]).

Θεώρημα 7.19 ([Chalopin et al., 2011a]) Το πρόβλημα BHS μπορεί να λυθεί σε οποιονδήποτε προσανατολισμένο τόρο με ακριβώς τρεις πράκτορες που μεταφέρουν δύο μετακινήσιμα σημάδια.

Η περίπτωση του τόρου χωρίς προσανατολισμό

Η περίπτωση ενός μη-προσανατολισμένου τόρου είναι αρκετά διαφορετική σε σχέση με τον προσανατολισμένο τόρο. Όπως δείχνουν τα σχετικά πρόσφατα αποτελέσματα του άρθρου [Markou and Paquette, 2012], ένας οποιοσδήποτε σταθερός αριθμός από DFAs με ένα μετακινήσιμο σημάδι δεν μπορεί να εξερευνήσει όλους τους (ασφαλείς) τόρους χωρίς προσανατολισμό και έτσι το πρόβλημα BHS δεν μπορεί να λυθεί. Ωστόσο, αποτελεί έκπληξη[1] ότι, όπως παρουσιάζεται στο [Markou and Paquette, 2012], ένας πράκτορας με δύο μετακινήσιμα σημάδια μπορεί να εξερευνήσει οποιονδήποτε (ασφαλή) τόρο χωρίς προσανατολισμό. Αυτό το αποτέλεσμα δίνει την ελπίδα ότι ακόμη και σε έναν τόρο χωρίς προσανατολισμό το πρόβλημα BHS μπορεί να λυθεί από ένα μικρό αριθμό πρακτόρων με μετακινήσιμα σημάδια. Σε έναν μερικώς μη-προσανατολισμένο τόρο το πρόβλημα μπορεί να λυθεί από πέντε πράκτορες με σταθερή μνήμη και τρία μετακινήσιμα σημάδια ([Markou and Paquette, 2012]).

7.3 Σχόλια και βιβλιογραφικές παρατηρήσεις

Μοντέλα περισσότερο ή λιγότερο ισχυρών σφαλμάτων από μια μαύρη τρύπα, έχουν επίσης εμφανιστεί στη διεθνή βιβλιογραφία. Στο άρθρο [Cooper et al., 2010] οι συγγραφείς μελετούν το πρόβλημα της γρήγορης αναγνώρισης και επισκευής ειδικών σφαλμάτων τα οποία μπορούν να καταστρέψουν μόνο τον πρώτο πράκτορα που τα επισκέπτεται χρησιμοποιώντας πράκτορες με μνήμη που ξεκινούν στον ίδιο κόμβο ενός συγχρονισμένου γνωστού δικτύου.

Όπως έχει ήδη αναφερθεί, η μαύρη τρύπα είναι ένα ιδιαίτερο είδος ενός κακόβουλου κόμβου με μια πολύ απλή συμπεριφορά: καταστρέφει κάθε πράκτορα αμέσως μόλις επισκεφτεί τον κόμβο, χωρίς να αφήνει ίχνη. Στην πραγματικότητα όμως, ένας κακόβουλος κόμβος μπορεί να έχει πιο πολύπλοκη συμπεριφορά. Για παράδειγμα μπορεί να έχει τη δυνατότητα να αποφασίσει πότε

[1] Με δεδομένο το παλιό αποτέλεσμα του Rollik ([Rollik, 1980]), στο οποίο αποδεικνύεται ότι κανένα πεπερασμένο σύνολο από πεπερασμένα αυτόματα δεν μπορεί να εξερευνήσει όλα τα επίπεδα γραφήματα βαθμού 3.

θα καταστρέψει κάποιον πράκτορα που τον επισκέπτεται, να αναπαραγάγει ή να εισάγει ψεύτικους πράκτορες στο δίκτυο, να αλλοιώσει τις πληροφορίες που είναι αποθηκευμένες στον πίνακα του κόμβου ή να μην υπακούει στα πρωτόκολλα επικοινωνίας (π.χ., να εκτελεί τις οδηγίες των πρακτόρων με τη σειρά που αποφασίζει). Αυτές είναι όλες πολύ ενδιαφέρουσες κατευθύνσεις της έρευνας για την προστασία ενός δικτύου από εχθρικούς κόμβους. Στο άρθρο [Kràlovic and Miklìk, 2010] εμφανίζεται η πρώτη μελέτη του πώς οι ικανότητες ενός κακόβουλου κόμβου επηρεάζουν την επιλυσιμότητα των προβλημάτων εξερεύνησης. Εκεί, μελετάται μια διαφορετική επικίνδυνη συμπεριφορά για πράκτορες που ξεκινούν στον ίδιο κόμβο: οι συγγραφείς θεωρούν έναν ασύγχρονο δακτύλιο με πίνακες στους κόμβους στον οποίο υπάρχει μια μαύρη τρύπα με βυζαντινή συμπεριφορά: αυτή η μαύρη τρύπα δεν καταστρέφει πάντα έναν πράκτορα που επισκέπτεται τον κόμβο αλλά μόνο όταν εκείνη το αποφασίσει, ενώ μπορεί επίσης να αλλοιώσει τις πληροφορίες που είναι αποθηκευμένες στον πίνακα του κόμβου. Τα αποτελέσματα του άρθρου δείχνουν ότι, στην περίπτωση ενός δακτυλίου, όταν ο κακόβουλος κόμβος δεν μπορεί να αλλοιώσει τη μνήμη ενός πράκτορα που τον επισκέπτεται, το πρόβλημα της περιοδικής ανάκτησης δεδομένων είναι επιλύσιμο από ένα σταθερό αριθμό πρακτόρων.

Βιβλιογραφία

[Chalopin et al., 2011a] Chalopin, J., Das, S., Labourel, A., and Markou, E. (2011a). Black hole search with finite automata scattered in a synchronous torus. In *Proc. of 25th Int. Symp. on Distributed Computing*, LNCS 6950, pages 432–446.

[Chalopin et al., 2011b] Chalopin, J., Das, S., Labourel, A., and Markou, E. (2011b). Tight bounds for scattered black hole search in a ring. In *Proc. of 18th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 6796, pages 186–197.

[Cooper et al., 2010] Cooper, C., Klasing, R., and Radzik, T. (2010). Locating and repairing faults in a network with mobile agents. *Theoretical Computer Science*, 411(14-15):1638–1647.

[Czyzowicz et al., 2006] Czyzowicz, J., Kowalski, D., Markou, E., and Pelc, A. (2006). Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2-3):229–242.

[Engebretsen and Karpinski, 2006] Engebretsen, L. and Karpinski, M. (2006). TSP with bounded metrics. *Journal of Computer and System Sciences*, 72(4):509–546.

[Garey et al., 1976] Garey, M. R., Johnson, D. S., and Tarjan, R. E. (1976). The planar hamiltonian circuit problem is np-complete. *SIAM Journal on Computing*, 5(4):704–714.

[Kann,] Kann, V. Minimum steiner tree. http://www.nada.kth.se/∼viggo/wwwcompendium/node78.html.

[Klasing et al., 2005] Klasing, R., Markou, E., Radzik, T., and Sarracco, F. (2005). Hardness and approximation results for black hole search in arbitrary graphs. In *Proc. of 12th Int. Colloquium on Structural Information and Communication Complexity*, LNCS 3499, pages 200–215.

[Klasing et al., 2007] Klasing, R., Markou, E., Radzik, T., and Sarracco, F. (2007). Hardness and approximation results for black hole search in arbitrary graphs. *Theoretical Computer Science*, 384(2-3):201–221.

[Klasing et al., 2008] Klasing, R., Markou, E., Radzik, T., and Sarracco, F. (2008). Approximation bounds for black hole search problems. *Networks*, 52(4):216–226.

[Kràlovic and Miklìk, 2010] Kràlovic, R. and Miklìk, S. (2010). Periodic data retrieval problem in rings containing a malicious host. In *Proc. of 17th Int. Colloquium on Structural Information and Communication Complexity*, pages 156–167.

[Markou and Paquette, 2012] Markou, E. and Paquette, M. (2012). Black hole search and exploration in unoriented tori with synchronous scattered finite automata. In *Proc. of 16th International Conference on Principles of Distributed Systems (OPODIS)*, LNCS 7702, pages 239–253.

[Robins and Zelikovsky, 2000] Robins, G. and Zelikovsky, A. (2000). Improved steiner tree approximation in graphs. In *Proc. of 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779.

[Rollik, 1980] Rollik, H. (1980). Automaten in planaren graphen. *Acta Informatica*, 13:287–298.

ΚΕΦΑΛΑΙΟ 8

Αξιόπιστη Επικοινωνία σε Κατανεμημένα Περιβάλλοντα

Καθώς τα δίκτυα επικοινωνιών αυξάνονται σε μέγεθος, γίνονται ολοένα και πιο ευάλωτα σε βλάβες των στοιχείων που τα απαρτίζουν, ηθελημένες ή αθέλητες. Οι αυτόνομες οντότητες που απαρτίζουν το δίκτυο συνεργάζονται για να εκτελέσουν εργασίες όπως απαιτητικούς υπολογισμούς, διανομή ψηφιακού υλικού ή λήψη κοινών αποφάσεων, υπό την απουσία μιας κεντρικής αρχής συντονισμού. Για τον λόγο αυτό ακολουθούν ένα προσυμφωνημένο *πρωτόκολλο επικοινωνίας*. Συχνά, κάποιες οντότητες δρουν με τρόπο που δεν συμβαδίζει με το συμφωνημένο πρωτόκολλο, είτε λόγω αστοχίας του υλικού ή του λογισμικού, είτε λόγω της επίδρασης κάποιου κακόβουλου αντιπάλου που ελέγχει κάποιους από τους πράκτορες. Η έννοια του αντιπάλου μπορεί να μοντελοποιήσει ακόμη και τις μη ηθελημένες βλάβες, όπως εξηγούμε παρακάτω.

Η μελέτη της συμπεριφοράς κατανεμημένων συστημάτων υπό την παρουσία βλαβών, περιλαμβάνει το σχεδιασμό αλγορίθμων οι οποίοι επιτυγχάνουν το ζητούμενο στόχο, παρά την ύπαρξη *διεφθαρμένων* πρακτόρων στο δίκτυο, η ταυτότητα μάλιστα των οποίων θεωρείται (αρχικά τουλάχιστον) άγνωστη. Για να μοντελοποιήσουμε τη διαφθορά των παικτών, θεωρούμε έναν κεντρικό *αντίπαλο*, ο οποίος ελέγχει εν μέρει τη συμπεριφορά κάποιων παικτών, τους οποίους έχει καταφέρει να διαφθείρει. Οι πράκτορες οι οποίοι δεν βρίσκονται υπό την επιρροή του αντιπάλου θα ονομάζονται *έντιμοι*. Διάφοροι τύποι διαφθοράς έχουν προταθεί στη σχετική βιβλιογραφία. Ανάμεσα σε αυτούς τους τύπους αντιπάλου, ο *Βυζαντινός ή Ενεργός* (Byzantine or Active) αντίπαλος, μοντελοποιεί τη χειρότερη περίπτωση αποκλίνουσας συμπεριφοράς. Συγκεκριμένα, θεωρείται ότι οι παίκτες που ελέγχονται από τον αντίπαλο, μπορούν να συμπεριφερθούν με αυθαίρετο τρόπο, σταματώντας τη ροή ή αλλάζοντας τη δρομολόγηση και το περιεχόμενό της πληροφορίας που πρέπει να μεταδώσουν με τρόπο που μπορεί να αποβεί καταστροφικός για την ορθότητα της επικοινωνίας και κατ'επέκταση οποιασδήποτε διαδικασίας απαιτεί επικοινωνία μεταξύ των συμμετεχόντων.

Διάφορες προσεγγίσεις έχουν προταθεί στη βιβλιογραφία, προκειμένου να μοντελοποιηθεί το πρόβλημα της αξιόπιστης επικοινωνίας. Ένα θεμελιώδες πρόβλημα που φαίνεται να συμπυκνώνει

και να αντιπροσωπεύει τα περισσότερα σχετικά προβλήματα είναι το πρόβλημα της Αξιόπιστης Εκπομπής, που παρουσιάζεται αναλυτικά στην επόμενη ενότητα.

8.1 Το πρόβλημα της Αξιόπιστης Εκπομπής

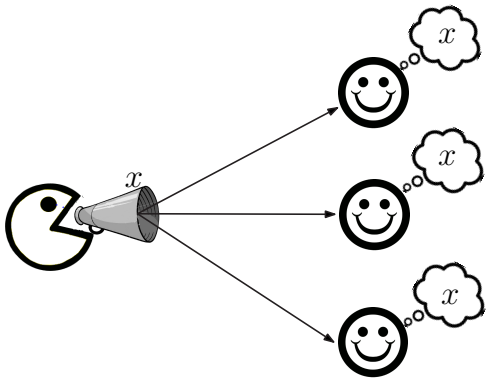
Ένα θεμελιώδες πρόβλημα στον τομέα των κατανεμημένων συστημάτων είναι αυτό της *Αξιόπιστης Εκπομπής* (Reliable Broadcast) ή αλλιώς το *πρόβλημα των Βυζαντινών Στρατηγών* (Byzantine Generals)· παρουσιάστηκε για πρώτη φορά από τους Lamport, Shostak and Pease [Lamport et al., 1982] το 1982. Στο πρόβλημα αυτό, ο στόχος είναι η ορθή μετάδοση ενός μηνύματος από έναν καθορισμένο *παίκτη-διανομέα* σε όλο το δίκτυο ανεξάρτητα από την ύπαρξη ενός Βυζαντινού αντιπάλου. Κατά κύριο λόγο, τα προβλήματα συμφωνίας έχουν μελετηθεί στη βιβλιογραφία στα πλαίσια του *μοντέλου φραγμένου αντιπάλου* (threshold adversary model), όπου υπάρχει η υπόθεση ύπαρξης ενός άνω ορίου t στον αριθμό των διεφθαρμένων παικτών: ο αντίπαλος μπορεί να διαφθείρει οποιουσδήποτε t παίκτες επιθυμεί. Σε αυτό το μοντέλο, έχει αποδειχθεί ήδη από την εργασία [Lamport et al., 1982], ότι αξιόπιστη εκπομπή μπορεί να επιτευχθεί αν και μόνον αν $t < n/3$, όπου n είναι ο συνολικός αριθμός των παικτών που συμμετέχουν.

Το πρόβλημα της αξιόπιστης εκπομπής, έχει μελετηθεί εκτενώς σε πλήρη δίκτυα υπό το μοντέλο φραγμένου αντιπάλου από το 1982 έως το 1998, όπου οι Garay και Moses [Garay and Moses, 1998] παρουσίασαν το πρώτο *πλήρως πολυωνυμικό* πρωτόκολλο το οποίο ήταν βέλτιστο ως προς την πολυπλοκότητα γύρων και ανεχόταν τον μέγιστο αριθμό διαφθορών $t = \lceil n/3 \rceil - 1$. Όταν υπάρχει πλήρης επικοινωνία, 'ένας-προς-έναν', μεταξύ των παικτών η δυσκολία του προβλήματος συνοψίζεται στο σενάριο όπου ο κόμβος-διανομέας είναι διεφθαρμένος. Όπως φαίνεται στο Σχήμα 8.1, σε πραγματικά δίκτυα όπου ο διανομέας συνδέεται μέσω διαφορετικών καναλιών επικοινωνίας με κάθε παίκτη, ο διανομέας μπορεί να στείλει αντικρουόμενες τιμές στους παίκτες. Σε αυτήν την περίπτωση, απαιτούμε τελικά οι παίκτες να συμφωνήσουν σε μια κοινή τιμή.

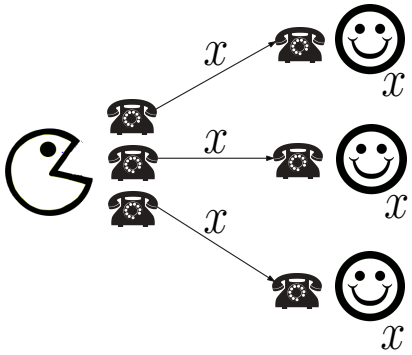
Ο σκοπός ενός πρωτοκόλλου αξιόπιστης εκπομπής είναι να προσομοιώσει μια ιδανική εκπομπή, όπου όλοι λαμβάνουν την ίδια τιμή από τον διανομέα την ίδια χρονική στιγμή, μέσω της επικοινωνίας των παικτών. Ουσιαστικά απαιτείται να παρακαμφθούν οι λανθασμένες συμπεριφορές, χωρίς να χαθεί η ομοφωνία στο σύστημα. Ακολουθεί ο τυπικός ορισμός του προβλήματος της Αξιόπιστης Εκπομπής.

Ορισμός 8.1 Πρόβλημα Αξιόπιστης Εκπομπής Έστω $V = \{p_1, p_2, \cdots, p_n\}$ ένα σύνολο n παικτών, X ο πεπερασμένος χώρος των μηνυμάτων και $D \in v$ ο κόμβος-διανομέας. Έστω επίσης ένα πρωτόκολλο Π μεταξύ των παικτών V με τιμές μηνυμάτων στο X, όπου ο D έχει αρχική τιμή $x_D \in X$ και κάθε παίκτης p_i τελικά αποφασίζει σε μια τιμή $y_i \in X$[1] Το πρωτόκολλο Π ονομάζεται πρωτόκολλο Αξιόπιστης Εκπομπής αν και μόνον αν ικανοποιεί τις ακόλουθες συνθήκες:

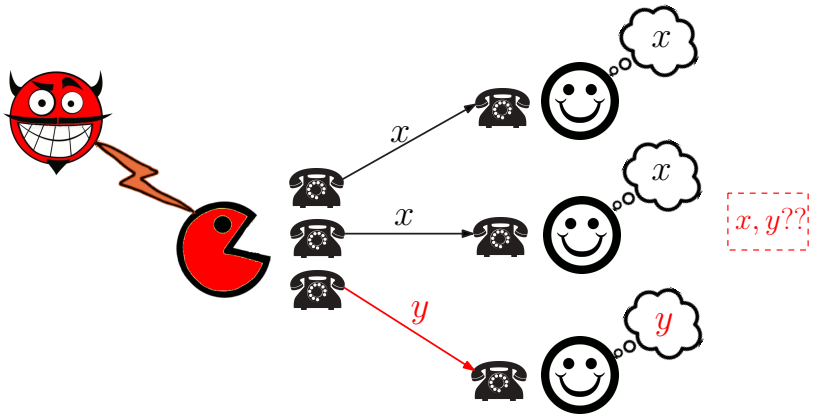
[1] Διαισθητικά με την έννοια της 'απόφασης' εννοούμε ότι ο παίκτης έχει καταλήξει οριστικά στο συμπέρασμα ότι η ορθή τιμή είναι η y_i. Τυπικά, ορίζουμε ότι ένας παίκτης *αποφασίζει* σε μια τιμή y_i όταν η αντίστοιχη μηχανή Turing φτάνει σε μια προκαθορισμένη κατάσταση απόφασης S_{y_i}, και δεν επιτρέπεται η μηχανή βρεθεί αργότερα σε διαφορετική κατάσταση απόφασης.



(α΄) Ιδανική Εκπομπη



(β΄) Πραγματική Εκπομπη



(γ΄) Πραγματική Εκπομπη με διεφθαρμένο διανομέα

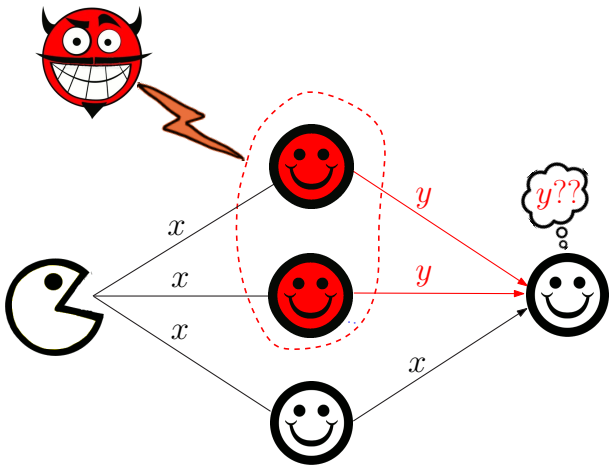
Σχήμα 8.1: Ιδανική και Πραγματική Αξιόπιστη Εκπομπή

1. *Συνέπεια:* Όλοι οι τίμιοι παίκτες τελικά αποφασίζουν στην ίδια τιμή, δηλαδή, $\forall p_i, p_j$ που είναι τίμιοι, ισχύει ότι $y_j = y_i$.

2. *Ορθότητα:* Αν ο κόμβος-διανομέας D είναι τίμιος, τότε όλοι οι τίμιοι παίκτες τελικά αποφασίζουν στην τιμή x_D.

Σε μη-πλήρη δίκτυα, όπου ο διανομέας δεν είναι άμεσα συνδεδεμένος με όλους τους παίκτες, το πρόβλημα παρουσιάζει μια επιπλέον δυσκολία: ακόμα και στην περίπτωση που ο κόμβος-διανομέας είναι ένας τίμιος παίκτης, ο αντίπαλος μπορεί να διαφθείρει ένα κρίσιμο μέρος του δι-

κτύου έτσι ώστε να καταστήσει αδύνατη την απόφαση στην σωστή τιμή για κάποιους τίμιους παίκτες. Το σενάριο αυτό απεικονίζεται στο Σχήμα 8.2.



Σχήμα 8.2: Αξιόπιστη Εκπομπή σε μη-πλήρη δίκτυα

Η περίπτωση της Αξιόπιστης Εκπομπής σε μη-πλήρη δίκτυα έχει μελετηθεί σε πολύ μικρότερο βαθμό. Μετά την αρχική μελέτη του Dolev [Dolev, 1982] η οποία αντιμετώπιζε κατευθείαν το συγκεκριμένο πρόβλημα, ακολούθησαν άλλες εργασίες [Dolev, 1982, Dolev et al., 1993, Kumar et al., 2002] όπου ουσιαστικά αντιμετωπίζουν το πρόβλημα κυρίως μέσω πρωτοκόλλων για *αξιόπιστη και ασφαλή μετάδοση μηνύματος*. Από τα τελευταία προκύπτουν και πρωτόκολλα Αξιόπιστης Εκπομπής για μη-πλήρη δίκτυα, αφού μπορούν να χρησιμοποιηθούν για την προσομοίωση αξιόπιστων καναλιών επικοινωνίας για κάθε ζεύγος παικτών. Όπως είναι φυσικό, εκτός από το φράγμα $t < n/3$, απαιτείται να τηρούνται επιπλέον περιορισμοί ως προς τη συνδεσιμότητα του δικτύου για να είναι το πρόβλημα επιλύσιμο. Για παράδειγμα, στην περίπτωση του φραγμένου αντιπάλου, το πολύ $t < c/2$ διαφθορές παικτών μπορούν να γίνουν ανεκτές, όπου c είναι η συνδεσιμότητα του δικτύου και αυτό το φράγμα είναι ακριβές, όπως αποδείχθηκε στο [Dolev, 1982], δηλαδή για $t \geq c/2$ ο αντίπαλος μπορεί να οδηγήσει κάποιους παίκτες σε λάθος απόφαση.

8.1.1 Αξιόπιστη Επικοινωνία με Τίμιο Διανομέα

Στην υποενότητα αυτή θα συζητήσουμε το πρόβλημα της *Αξιόπιστης Εκπομπής με τίμιο κόμβο-διανομέα* σε γενικά (μη-πλήρη) δίκτυα. Στη συνέχεια, για λόγους συντομίας, θα αναφερόμαστε σε αυτό το πρόβλημα πιο απλά ως *πρόβλημα Εκπομπής*. Όπως θα δούμε και στην Ενότητα 8.3, η περίπτωση αυτή αποτυπώνει ουσιαστικά τη δυσκολία του γενικού προβλήματος, όπου ακόμη και ο διανομέας μπορεί να διαφθαρεί. Αυτό προκύπτει, γιατί δοθείσας μίας λύσης στο απλούστερο

αυτό πρόβλημα, μπορούμε να με προσομοιάσουμε τις μεταδόσεις ενός πρωτοκόλλου Αξιόπιστης Εκπομπής για πλήρη δίκτυα. Ο ορισμός του προβλήματος ακολουθεί.

Πρόβλημα Εκπομπής με Τίμιο Διανομέα. Το δίκτυο μοντελοποιείται ως ένα γράφημα $G = (V, E)$, όπου V είναι το σύνολο των παικτών και το σύνολο ακμών E αντιπροσωπεύει κανάλια επικοινωνίας μεταξύ των παικτών. Υποθέτουμε την ύπαρξη ενός καθορισμένου, τίμιου κόμβου-διανομέα D, η αρχική τιμή του οποίου $x_D \in X$, όπου είναι ο χώρος των μηνυμάτων, πρέπει να διαδοθεί σε όλους τους παίκτες του δικτύου. Ένα κατανεμημένο πρωτόκολλο Π, ονομάζεται πρωτόκολλο Εκπομπής αν μετά το τέλος αυτού του πρωτοκόλλου, κάθε τίμιος παίκτης του δικτύου έχει αποφασίσει στην τιμή του διανομέα x_D, δηλαδή είναι σε θέση να εξάγει την τιμή x_D που στάλθηκε αρχικά από τον διανομέα.

Όπως έχουμε τονίσει προηγουμένως, το πρόβλημα έχει τετριμμένη λύση σε πλήρη δίκτυα· έτσι, εδώ θα εξετάσουμε την περίπτωση ελλιπών δικτύων επικοινωνίας.

Παρατηρούμε ότι η επίτευξη Εκπομπής, όπως ορίστηκε παραπάνω, είναι στην πραγματικότητα ισοδύναμη με την επίτευξη ορθής (αξιόπιστης) μετάδοσης μηνύματος από τον κόμβο-διανομέα D σε όλους τους παίκτες. Η υπόθεση του τίμιου κόμβου-διανομέα είναι ιδιαίτερης σημασίας στην περίπτωση των ασύρματων δικτύων. Σε αυτό το μοντέλο, λόγω της επικοινωνίας μέσω τοπικών εκπομπών, ο διανομέας ουσιαστικά δεσμεύεται να στείλει την ίδια αρχική τιμή σε όλους του γείτονες της και έτσι να δεσμευτεί σε αυτή.

Εδώ εξετάζουμε επίσης το συγγενές πρόβλημα της *Αξιόπιστης Μετάδοσης Μηνύματος* (Reliable Message Transmission–RMT), όπου ο κόμβος-διανομέας D επιθυμεί να μεταδώσει ένα μήνυμα σε έναν άλλον κόμβο-παραλήπτη. Στην πραγματικότητα, μια λύση του προβλήματος RMT για κάθε ζεύγος $(D, v), v \in V$, συνεπάγεται μια λύση του προβλήματος Εκπομπής. Ο τυπικός ορισμός του προβλήματος ακολουθεί.

Πρόβλημα Αξιόπιστης Μετάδοσης Μηνύματος (RMT). Υποθέτουμε την ύπαρξη ενός προκαθορισμένου πράκτορα-διανομέα D, η αρχική τιμή του οποίου $x_D \in X$, όπου είναι ο χώρος των μηνυμάτων, πρέπει να διαδοθεί σε έναν καθορισμένο πράκτορα R που ονομάζεται *παραλήπτης*. Ένα κατανεμημένο πρωτόκολλο, ονομάζεται πρωτόκολλο Αξιόπιστης Μετάδοσης Μηνύματος αν μετά το τέλος αυτού του πρωτοκόλλου, ο παραλήπτης R έχει αποφασίσει στην τιμή του διανομέα x_D, δηλαδή είναι σε θέση να εξάγει την τιμή x_D που του στάλθηκε αρχικά από τον διανομέας.

8.1.2 Το μοντέλο επικοινωνίας

Ένα δίκτυο επικοινωνίας αναπαρίσταται από ένα γράφημα $G = (V, E)$. Οι κόμβοι V του γραφήματος αντιπροσωπεύουν τους παίκτες και οι ακμές E τα μεταξύ τους κανάλια επικοινωνίας. Στη συνέχεια θα χρησιμοποιούμε τους όρους *παίκτης, κόμβος*, για να αναφερόμαστε στους συμμετέχοντες του συστήματος. Γενικά, ο κάθε παίκτης μπορεί να έχει κάποια αρχική πληροφορία/είσοδο

στην αρχή του πρωτοκόλλου. Η ανταλλαγή πληροφορίας μεταξύ των παικτών συντελείται μέσω των καναλιών επικοινωνίας. Τα κανάλια που χρησιμοποιούνται σε ένα τέτοιο δίκτυο μπορούν να ταξινομηθούν ανάλογα με τον βαθμό αξιοπιστίας και ασφάλειας που τα χαρακτηρίζει. Οι διάφορες κατηγορίες καναλιών επικοινωνίας φαίνονται παρακάτω.

- **Αξιόπιστα κανάλια (Authenticated):** Τα μηνύματα που μεταδίδονται μέσω τέτοιων καναλιών δεν μπορούν να παραποιηθούν από τον αντίπαλο αλλά ο αντίπαλος μπορεί να υποκλέψει το περιεχόμενο. Επιπλέον χρησιμοποιώντας αξιόπιστα κανάλια ο παραλήπτης ενός μηνύματος γνωρίζει πάντα την πραγματική ταυτότητα του αποστολέα.
- **Εμπιστευτικά κανάλια (Confidential):** Τα μηνύματα που μεταδίδονται μπορούν να παραποιηθούν από τον αντίπαλο αλλά δεν γίνεται να υποκλαπεί το περιεχόμενό τους.
- **Ασφαλή κανάλια (Secure):** Αξιόπιστα και εμπιστευτικά κανάλια.

Επίσης ως προς την καθυστέρηση μετάδοσης των μηνυμάτων έχουμε τις εξής κατηγορίες.

- **Συγχρονισμένα κανάλια (Synchronous):** Η καθυστέρηση της μετάδοσης των μηνυμάτων στο κανάλι φράσσεται από μια γνωστή σταθερά.
- **Ασύγχρονα κανάλια (Asynchronous):** Η καθυστέρηση, αν και πεπερασμένη, δεν φράσσεται από κάποια γνωστή σταθερά.

Η τοπολογία του δικτύου μπορεί να είναι *πλήρης* ή *μη-πλήρης*. Στη δεύτερη περίπτωση κάποια ζεύγη παικτών δεν μοιράζονται κάποιο κανάλι επικοινωνίας για να ανταλλάξουν μηνύματα μεταξύ τους.

Σε αυτό το κεφάλαιο θα εστιάσουμε κυρίως σε συγχρονισμένα μη-πλήρη δίκτυα με αξιόπιστα κανάλια.

8.1.3 Το μοντέλο του αντιπάλου

Όπως αναφέραμε και προηγουμένως, η αποκλίνουσα συμπεριφορά κάποιων παικτών μοντελοποιείται με έναν *κεντρικό αντίπαλο* ο οποίος τους διαφθείρει. Για παράδειγμα, ο αντίπαλος σε ένα πραγματικό σενάριο θα μπορούσε να αντιπροσωπεύει έναν hacker, ο οποίος έχει καταφέρει να αποκτήσει τον έλεγχο των μηχανημάτων που χειρίζονται οι παίκτες.

Είδος διαφθοράς

Διαφορετικά μοντέλα αντιπάλου μπορούν να προσδιοριστούν σε σχέση με την ικανότητα διαφθοράς του αντιπάλου· η χειρότερη περίπτωση αντιπάλου, την οποία μελετάμε εδώ, είναι ο *Βυζαντινός ή Ενεργός* (Byzantine or Active) αντίπαλος.

Βυζαντινός Αντίπαλος: Οι διεφθαρμένοι παίκτες βρίσκονται υπό τον πλήρη έλεγχο του αντιπάλου και η συμπεριφορά τους μπορεί να παρεκκλίνει από το πρωτόκολλο με οποιονδήποτε τρόπο επιθυμεί ο αντίπαλος.

Υπολογιστική δύναμη του αντιπάλου

Το μοντέλο αντιπάλου προσδιορίζει επίσης την υπολογιστική του ισχύ. Οι πιο συνηθισμένες υποθέσεις σε ότι αφορά το προηγούμενο είναι ότι ο αντίπαλος είτε είναι *υπολογιστικά περιορισμένος*, οπότε θεωρείται ότι περιορίζεται σε υπολογισμούς που διαρκούν πολυωνυμικό χρόνο ως προς κάποια παράμετρο ασφάλειας, είτε δεν έχει κανέναν περιορισμό ως προς την υπολογιστική του δύναμη και καλείται αντίπαλος *απεριόριστης υπολογιστικής ισχύος*.

Στην ενότητα αυτή, ασχολούμαστε με πρωτόκολλα που πετυχαίνουν τον στόχο τους ακόμα και υπό την παρουσία ενός Βυζαντινού αντιπάλου απεριόριστης υπολογιστικής ισχύος. Τα αποτελέσματα που παρουσιάζονται προφανώς ισχύουν και για άλλους τύπους αντιπάλου αφού το μοντέλο που χρησιμοποιούμε αποτελεί ένα σενάριο χειρότερης περίπτωσης ως προς επίπτωση της διαφθοράς στο σύστημα.

Ευάλωτα σύνολα πρακτόρων

Τέλος, τα είδη αντιπάλου μπορούν να διακριθούν σε σχέση με τα σύνολα των παικτών που μπορεί να διαφθείρουν. Το μοντέλο του *t-φραγμένου αντιπάλου* που έχει μελετηθεί εκτενώς στη βιβλιογραφία αντιστοιχεί στην κατάσταση όπου ο αντίπαλος μπορεί να διαφθείρει το πολύ t παίκτες στο δίκτυο. Στη συνέχεια εστιάζουμε στα ακόλουθα μοντέλα αντιπάλου ως προς το εύρος των παικτών που μπορούν να διαφθείρουν.

t-Τοπικά Φραγμένος Αντίπαλος. Ο αντίπαλος μπορεί να διαφθείρει το πολύ t κόμβους-παίκτες στην γειτονιά κάθε κόμβου. Το μοντέλο αυτό προτάθηκε από τον Koo [Koo, 2004] το 2004.

Η σημασία αυτού του μοντέλου προέρχεται, μεταξύ άλλων, από τους τοπικούς περιορισμούς που επιβάλλονται στον αντίπαλο, η οποία μπορεί να χρησιμοποιηθεί για τον σχεδιασμό κριτηρίων, τα οποία μπορούν να χρησιμοποιηθούν σε δίκτυα άγνωστης τοπολογίας όπου ο κάθε παίκτης γνωρίζει μόνο τους άμεσους γείτονές του με τους οποίους επικοινωνεί. Το μοντέλο τοπικά φραγμένου αντιπάλου είναι ιδιαίτερα σημαντικό στις σύγχρονες πρακτικές εφαρμογές. Για παράδειγμα, στα κοινωνικά δίκτυα είναι πιο πιθανό για έναν παίκτη να έχει μια αρκετά ακριβή εκτίμηση του μέγιστου αριθμού των κακόβουλων παικτών που μπορούν να εμφανιστούν στη γειτονιά του, παρά να έχει μια τέτοια εκτίμηση για ολόκληρο το δίκτυο. Στην πραγματικότητα, αυτό το σενάριο ισχύει για όλα τα είδη των δικτύων, όπου κάθε κόμβος θεωρείται ότι είναι είναι σε θέση να εκτιμήσει το μέγεθος της διαφθοράς στην άμεση γειτονιά του. Είναι επίσης φυσικό το φράγμα των διεφθαρμένων παικτών να ποικίλει σε διάφορα τμήματα του δικτύου. Έχει νόημα επομένως να ορίσουμε μια γενίκευση του t-τοπικά φραγμένου αντιπάλου όπου ο αντίπαλος έχει διαφορετικό, *μη ομοιόμορφο* φράγμα ως προς τους παίκτες που μπορεί να διαφθείρει σε κάθε γειτονιά. Σε αυτό το μοντέλο μη ομοιόμορφα φραγμένου αντιπάλου θα αναφερθούμε αργότερα. Παρακάτω ορίζουμε ένα ακόμη πιο γενικό μοντέλο που περιλαμβάνει το μοντέλο του t-τοπικά φραγμένου αντιπάλου και στην ομοιόμορφη και στην μη ομοιόμορφη εκδοχή του.

Μοντέλο Γενικού Αντιπάλου. Το μοντέλο γενικού αντιπάλου προτάθηκε και καθιερώθηκε από τους Hirt και Maurer στο [Hirt and Maurer, 1997], και εμπεριέχει όλα τα τα γνωστά μοντέλα αντιπάλου, συμπεριλαμβανομένων και των προαναφερθέντων. Ακολουθεί η τυπική περιγραφή του μοντέλου.

Ο αντίπαλος μπορεί να διαφθείρει οποιοδήποτε σύνολο παικτών σε μια δεδομένη οικογένεια συνόλων $\mathcal{Z}$ που ονομάζεται *δομή αντιπάλου του αντιπάλου*. Μια δομή $\mathcal{Z}$ για το σύνολο των παικτών V είναι μια *μονότονη* οικογένεια υποσυνόλων του V, δηλαδή μια οικογένεια συνόλων όπου κάθε υποσύνολο συνόλου της οικογένειας ανήκει σε αυτήν, δηλαδή:

$$\mathcal{Z} \subseteq 2^V, \text{ έτσι ώστε } \forall Z \in \mathcal{Z}, \forall Z' \subseteq Z, Z' \in \mathcal{Z}$$

8.1.4 Τοπολογική Γνώση

Όσον αφορά την αρχική γνώση που οι παίκτες διαθέτουν για την τοπολογία του δικτύου, επί το πλείστον έχουν μελετηθεί στη σχετική βιβλιογραφία οι επόμενες δύο περιπτώσεις.

- ***Ad Hoc* Δίκτυα (ή δίκτυα άγνωστης τοπολογίας) :** Κάθε παίκτης γνωρίζει μόνο τις ταυτότητες (ids) των άμεσων γειτόνων του.

- **Δίκτυα Γνωστής τοπολογίας:** Κάθε παίκτης γνωρίζει την τοπολογία ολόκληρου του δικτύου (ταυτότητες κόμβων και αντίστοιχες ακμές).

- **Δίκτυα Μερικώς Γνωστής τοπολογίας** (μοντέλο μερικής γνώσης – partial knowledge): Κάθε παίκτης γνωρίζει μόνο την τοπολογία ενός υπογραφήματος του δικτύου στο οποίο ανήκει και ο ίδιος.

Μοντέλο μερικής γνώσης. Το μοντέλο αυτό προτάθηκε πρόσφατα, στην εργασία [Pagourtzis et al., 2014]. Αποτελεί γενίκευση των προγενέστερων μοντέλων, ενώ καλύπτει και ένα μεγάλο πλήθος περιπτώσεων που δεν καλύπτονταν από τα προηγούμενα μοντέλα. Ανταποκρίνεται στο γεγονός ότι η γνώση ενός παίκτη ως προς τα πιθανά σύνολα διεφθαρμένων παικτών συσχετίζεται φυσιολογικά με την τοπολογική γνώση που αυτός κατέχει. Η ανάγκη μελέτης μοντέλων περιορισμένης γνώσης υπαγορεύεται από εφαρμογές σε δίκτυα ευρείας κλίμακας (π.χ. το διαδίκτυο), όπου η εκτίμηση του βαθμού δυσλειτουργίας μπορεί να γίνει με σχετική ακρίβεια από τον κάθε συμμετέχοντα στα πλαίσια της γειτονιάς του, ενώ μια συνολική εκτίμηση μπορεί να είναι δύσκολο να επιτευχθεί λόγω γεωγραφικών περιορισμών και περιορισμών δικαιοδοσίας. Επιπλέον, η εγγύτητα κόμβων σε κοινωνικά δίκτυα συχνά συσχετίζεται με αυξημένη ποσότητα διαθέσιμης πληροφορίας, γεγονός που δικαιολογεί περαιτέρω την ευστάθεια του μοντέλου.

8.1.5 Αποδοτικότητα Κατανεμημένων Πρωτοκόλλων

Η πολυπλοκότητα ενός κατανεμημένου πρωτοκόλλου μπορεί να αναλυθεί με βάση την μέγιστο χρόνο υπολογισμών που κάθε παίκτης θα χρειαστεί να εκτελέσει τοπικά (*πολυπλοκότητα τοπικών υπολογισμών, local computations complexity*) και με βάση τη συνολική ποσότητα πληροφορίας που απαιτείται να ανταλλαχθεί μεταξύ των παικτών κατά τη διάρκεια του πρωτοκόλλου (*πολυπλοκότητα επικοινωνίας ή μηνυμάτων, message complexity*). Επιπλέον, ο αριθμός των γύρων επικοινωνίας που απαιτούνται για την ολοκλήρωση του πρωτοκόλλου (*πολυπλοκότητα γύρων, round complexity*) είναι ένα άλλο μέτρο το οποίο λαμβάνεται υπόψη. Σε συγχρονισμένα δίκτυα, στα οποία κυρίως αναφερόμαστε στο κεφάλαιο αυτό, υποθέτουμε ότι κατά τη διάρκεια κάθε γύρου επικοινωνίας, όλοι οι κόμβοι παράλληλα λαμβάνουν τα τελευταία μηνύματα από τους γείτονές τους, εκτελούν τους τοπικούς υπολογισμούς που υπαγορεύονται από το πρωτόκολλο, και τελικά στέλνουν κάποια νέα μηνύματα στους γείτονές τους. οι παράγοντες που εξετάζονται προς βελτιστοποίηση σε ένα κατανεμημένο πρωτόκολλο είναι οι ακόλουθοι:

- **Πολυπλοκότητα Επικοινωνίας (Bit/Communication Complexity):** Ο μέγιστος συνολικός αριθμός bits που στέλνονται από όλους τους τίμιους παίκτες κατά τη διάρκεια του πρωτοκόλλου, μεταξύ όλων των πιθανών εκτελέσεων αυτού. Θα χρησιμοποιήσουμε επίσης τον όρο *Πολυπλοκότητα Μηνυμάτων* για τον συνολικό αριθμό των μηνυμάτων που αποστέλλονται από όλους τους τίμιους παίκτες.
- **Πολυπλοκότητα Γύρων (Round Complexity):** Ο μέγιστος αριθμός διαδοχικών γύρων επικοινωνίας, που χρειάζεται έτσι ώστε όλοι οι τίμιοι παίκτες να τερματίσουν το πρωτόκολλο, μεταξύ όλων των πιθανών εκτελέσεων αυτού.
- **Πολυπλοκότητα Τοπικών Υπολογισμών (Local Computations Complexity):** Η μέγιστη τοπική πολυπλοκότητα χειρότερης περίπτωσης μεταξύ όλων των παικτών και των πιθανών εκτελέσεων του πρωτοκόλλου.

Ένα κατανεμημένο πρωτόκολλο για το οποίο όλοι οι παραπάνω παράγοντες παραμένουν πολυωνυμικοί ονομάζεται *Πλήρως Πολυωνυμικό πρωτόκολλο* (Fully Polynomial protocol).

8.2 Αξιόπιστη Εκπομπή σε *Ad Hoc* Δίκτυα

Στην ενότητα αυτή θα εξετάσουμε το πρόβλημα της Αξιόπιστης Εκπομπής σε δίκτυα των οποίων η τοπολογία δεν είναι γνωστή στους συμμετέχοντες πράκτορες. Η περίπτωση αυτή έχει νόημα να μελετηθεί κυρίως σε μη-πλήρη δίκτυα. Θα αναλύσουμε την ανοχή (απέναντι σε σφάλματα, πιθανόν Βυζαντινού τύπου) του αλγορίθμου Certified Propagation Algorithm (CPA) [Koo, 2004], ο οποίος είναι ειδικά σχεδιασμένος για *ad hoc* δίκτυα, όπου κάθε πράκτορας v επικοινωνεί μόνο με ένα υποσύνολο των πρακτόρων, τους οποίους θεωρούμε *γείτονες (neighbors)* του v.

Ειδικότερα, θα αναλύσουμε το ζήτημα του προσδιορισμού του μέγιστου αριθμού των διεφθαρμένων παικτών $t_{\max}^{\text{CPA}}$ που ο αλγόριθμος CPA μπορεί να αντιμετωπίσει στο μοντέλο του t-τοπικά

φραγμένου αντιπάλου, στο οποίο ο αντίπαλος μπορεί να διαφθείρει το πολύ t παίκτες στην γειτονιά του κάθε παίκτη. Για κάθε γράφημα G και διανομέα D θα δούμε άνω και κάτω φράγματα για το $t^{\text{CPA}}_{\max}$, τα οποία μπορούν να υπολογιστούν αποδοτικά αποδοτικά μέσω μιας γραφοθεωρητικής παραμέτρου που παρουσιάζουμε σε αυτήν την ενότητα. Παράλληλα, θα παρουσιάσουμε έναν αποδοτικό 2-προσεγγιστικό αλγόριθμο για τον υπολογισμό του $t^{\text{CPA}}_{\max}$. Θα ορίσουμε γραφοθεωρητικές παραμέτρους, μία εκ των οποίων αντιστοιχεί ακριβώς στην μέγιστη ανοχή $t^{\text{CPA}}_{\max}$ του CPA. Τέλος, θα αποδείξουμε την αξιοσημείωτη ιδιότητα της *Μοναδικότητας του CPA*, όπως αυτή ορίστηκε στην εργασία [Pelc and Peleg, 2005], δηλαδή το γεγονός ότι ο CPA μπορεί να επιτύχει εκπομπή οποτεδήποτε αυτό είναι εφικτό από οποιοδήποτε πρωτόκολλο εκπομπής.

8.2.1 Το μοντέλο του τοπικά φραγμένου αντιπάλου

Στην περίπτωση ενός έντιμου διανομέα το αναγκαίο φράγμα $t < n/3$ για την επιλυσιμότητα του προβλήματος της Αξιόπιστης Εκπομπής δεν ισχύει· για παράδειγμα, σε πλήρη δίκτυα το πρόβλημα γίνεται τετριμμένο για οποιοδήποτε t. Ωστόσο, σε μη-πλήρη δίκτυα η κατάσταση είναι διαφορετική. Ένας μικρός αριθμός των προδοτών (διεφθαρμένοι παίκτες) μπορούν να καταφέρουν να αλλοιώσουν το αποτέλεσμα του πρωτοκόλλου, ελέγχοντας ένα κρίσιμο μέρος του δικτύου, π.χ. αν οι διεφθαρμένοι παίκτες απαρτίζουν έναν *διαχωριστή* του δικτύου, δηλαδή ένα σύνολο κόμβων που η αφαίρεσή τους από το γράφημα το καθιστά μη συνεκτικό. Είναι επομένως λογικό να οριστούν κριτήρια, βασισμένα στη δομή του γραφήματος (σε γραφοθεωρητικές παραμέτρους), ώστε να φράσσεται ο αριθμός ή να περιοριστεί η κατανομή προδοτών στο δίκτυο.

Μια προσέγγιση προς αυτή την κατεύθυνση είναι η εξέταση τοπολογικών περιορισμών ως προς την ικανότητα διαφθοράς του αντιπάλου. Η σημασία των τοπικών περιορισμών έγκειται, μεταξύ άλλων, στο γεγονός ότι αυτοί οι περιορισμοί μπορούν να χρησιμοποιηθούν για την εξαγωγή τοπικών κριτηρίων τα οποία οι παίκτες μπορούν να χρησιμοποιήσουν για την επίτευξη Εκπομπής σε *ad hoc* δίκτυα. Ένα τέτοιο παράδειγμα είναι το μοντέλο *t-τοπικά φραγμένου αντιπάλου* που προτάθηκε από τον Koo στο [Koo, 2004], στο οποίο το πολύ t διεφθαρμένοι πράκτορες μπορούν να εμφανιστούν στη γειτονιά κάθε πράκτορα.

Ο Koo [Koo, 2004] πρότεινε ένα απλό, αλλά με μεγάλες δυνατότητες πρωτόκολλο για την Εκπομπή στο μοντέλο τοπικά φραγμένου αντιπάλου. Το πρωτόκολλο αυτό, ονομάστηκε αργότερα από τους Pelc, Peleg [Pelc and Peleg, 2005] *Certified Propagation Algorithm*(CPA) και μελετήθηκε από τον Koo σε δίκτυα ειδικής τοπολογίας (δίκτυα πλέγματος). Το 2005, οι Pelc και Peleg [Pelc and Peleg, 2005] μελέτησαν το μοντέλο t-τοπικά φραγμένου αντιπάλου σε γραφήματα γενικής τοπολογίας και πρότειναν μια ικανή τοπολογική συνθήκη υπό την οποία ο CPA μπορεί να επιτύχει εκπομπή σε κάθε δίκτυο. Επίσης παρείχαν ένα άνω φράγμα, που περιγράφεται από μια γραφοθεωρητική παράμετρο, για τον αριθμό των διεφθαρμένων παικτών t που μπορεί να γίνει ανεκτός τοπικά από οποιοδήποτε πρωτόκολλο Εκπομπής. Η εξαγωγή ακριβέστερων φραγμάτων διατυπώθηκε από τους Pelc, Peleg σαν ανοικτό πρόβλημα. Σε αυτήν την κατεύθυνση, προτάθηκε από τους Ichimura and Shigeno [Ichimura and Shigeno, 2010] μια αποδοτικά υπολογίσιμη γραφοθεωρητική

παράμετρος η οποία δίνει τελικά ένα έναν καλύτερο, αλλά όχι ακριβή, χαρακτηρισμό της οικογένειας των γραφημάτων στην οποία ο CPA πετυχαίνει Εκπομπή. Έχει παραμείνει ανοικτό πρόβλημα από το 2005 η εξαγωγή μιας παραμέτρου η οποία αποκαλύπτει τον ακριβή αριθμό προδοτών t για τον οποίο ο CPA είναι t-*ανεκτικός*, δηλαδή ο αριθμός t ο οποίος μπορεί να γίνει τοπικά ανεκτός από τον CPA για οποιοδήποτε γράφημα G με διανομέα D.

8.2.2 Γραφοθεωρητική προσέγγιση

Στη συνέχεια θα συζητήσουμε μια αναγκαία και ικανή συνθήκη για την ορθότητα του CPA, που διατυπώθηκε στην εργασία [Litsas et al., 2013]. Η προσέγγισή αυτή επέτρεψε να δοθεί μια καταφατική απάντηση στο ανοιχτό πρόβλημα της *μοναδικότητας του CPA* [Pelc and Peleg, 2005]. Πιο αναλυτικά, τα αποτελέσματα που θα δούμε στηρίζονται σε τρεις νέες γραφοθεωρητικές παραμέτρους:

- Η παράμετρος $\mathcal{K}(G, D)$ προσδιορίζεται μέσω μιας κατάλληλης *διάταξης επιπέδων* των κόμβων του γραφήματος. Δείχνουμε ότι η συνθήκη $t < \mathcal{K}(G, D)/2$ είναι ικανή για να είναι ο CPA to be t-τοπικά ανεκτικός και ότι η $t < \mathcal{K}(G, D)$ αναγκαία συνθήκη, γεγονός που υποδηλώνει ότι $\lceil \mathcal{K}(G, D)/2 \rceil - 1 \leq t^{\text{CPA}}_{\text{max}} < \mathcal{K}(G, D)$. Αποδεικνύουμε ότι η παράμετρος αυτή συμπίπτει με την παράμετρο $\widetilde{\mathcal{X}}(G, D)$ που παρουσιάζεται στο [Ichimura and Shigeno, 2010]. Επιπλέον προτείνουμε έναν αποδοτικό αλγόριθμο για τον υπολογισμό της παραμέτρου $\mathcal{K}(G, D)$, ο οποίος είναι χαμηλότερης χρονικής πολυπλοκότητας από τον αλγόριθμο που προτείνεται στο [Ichimura and Shigeno, 2010] για τον υπολογισμό της $\widetilde{\mathcal{X}}(G, D)$. Σημειώνεται ότι αυτό παρέχει άμεσα έναν αποδοτικό 2-προσεγγιστικό αλγόριθμο για τον υπολογισμό του $t^{\text{CPA}}_{\text{max}}$ και παρέχουμε ένα παράδειγμα που δείχνει ότι ο λόγος προσέγγισης 2 είναι ακριβής για αυτόν τον αλγόριθμο.

- Η παράμετρος $\mathcal{M}(G, D, t)$, που εξαρτάται από την τιμή t, είναι μια παράμετρος που μπορεί να χρησιμοποιηθεί για να απαντήσουμε στο εάν ο CPA είναι t-τοπικά ανεκτικός για ένα γράφημα G με διανομέα D, ελέγχοντας απλά εάν ισχύει το $\mathcal{M}(G, D, t) \geq t + 1$. Επομένως, μέσω αυτής της παραμέτρου, παρέχουμε μία αναγκαία και ικανή συνθήκη έτσι ώστε ο CPA να είναι t-τοπικά ανεκτικός. Μια τέτοια συνθήκη δεν ήταν γνωστή στην βιβλιογραφία μέχρι πολύ πρόσφατα, όπου ανεξάρτητα από την εργασία [Litsas et al., 2013], παρουσιάστηκε μία αναγκαία και ικανή συνθήκη στην εργασία [Tseng et al., 2015]. Ωστόσο, ο τρόπος με τον οποίο ορίζεται η συνθήκη στο [Tseng et al., 2015], ορίζει έμμεσα έναν υπερεκθετικό αλγόριθμο για τον έλεγχο της (στην πραγματικότητα δεν παρουσιάζεται κάποιος αλγόριθμος στο [Tseng et al., 2015]). Από την άλλη πλευρά, θα δούμε ότι ακόμη και ένας αφελής αλγόριθμος για τον υπολογισμό της παραμέτρου $\mathcal{M}(G, D, t)$ δεν χρειάζεται περισσότερο εκθετικό χρόνο.

- Τέλος, η παράμετρος $\mathcal{T}(G, D) = \max\{t \in \mathrm{N} \mid \mathcal{M}(G, D, t) \geq t + 1\}$, μας δίνει ακριβώς το μέγιστο αριθμό διεφθαρμένων παικτών που μπορεί να ανεχθεί ο CPA σε κάθε γειτονιά, επομένως ταυτίζεται με την παράμετρο $t^{\mathrm{CPA}}_{\mathrm{max}}(G, D)$.

Επιπλέον, χρησιμοποιώντας την παράμετρο $\mathcal{M}(G, D, t)$ αποδεικνύεται πως ο CPA είναι τελικά *μοναδικός* μεταξύ *ad hoc* αλγορίθμων Εκπομπής. Δηλαδή, αν ένας *ad hoc* αλγόριθμος Εκπομπής είναι t-ανεκτικός για ένα γράφημα G με διανομέα D, τότε και ο CPA είναι t-ανεκτικός για τα G, D. Με αυτόν τον τρόπο, παρέχεται μια καταφατική απάντηση στο ανοιχτό ερώτημα της μοναδικότητας του CPA που είχε τεθεί στο [Pelc and Peleg, 2005].

8.2.3 Ο Αλγόριθμος CPA

Όπως εξηγήσαμε νωρίτερα, θεωρούμε ένα δίκτυο όπου μπορούν να εμφανιστούν το πολύ t διαφθορές στην γειτονιά κάθε κόμβου. Κάθε σύνολο με την παραπάνω ιδιότητα λοιπόν είναι ένα πιθανό σύνολο διεφθαρμένων παικτών. Στη συνέχεια θα χρησιμοποιούμε τον εξής συμβολισμό: δοθέντος ενός γραφήματος $G = (V, E)$ θα συμβολίζουμε με $\mathcal{N}(v)$ τη γειτονιά ενός κόμβου v στο G, επίσης θα συμβολίζουμε τους κόμβους ενός γραφήματος G με $V(G)$. Οι παρακάτω έννοιες παρουσιάστηκαν στο [Pelc and Peleg, 2005] και χρησιμοποιούνται στη συνέχεια.

Ορισμός 8.2 **t-τοπικό σύνολο** Ένα σύνολο A ονομάζεται t-τοπικό αν έχει την ιδιότητα,

$$\forall v \in V, |A \cap \mathcal{N}(v)| \leq t$$

Ορισμός 8.3 **t-τοπικά ανεκτικός αλγόριθμος** Δοθέντος ενός γραφήματος G και ενός διανομέα D, ο αλγόριθμος $\mathcal{A}$ ονομάζεται t-τοπικά ανεκτικός (t-locally resilient) για τα G, D εάν για οποιοδήποτε t-τοπικό σύνολο διεφθαρμένων παικτών και οποιαδήποτε συμπεριφορά αυτών, ο $\mathcal{A}$ πετυχαίνει Εκπομπή στα G, D.

Ορισμός 8.4 **ασφαλής αλγόριθμος** Ένας αλγόριθμος μέσω του οποίου κανένας παίκτης δεν αποφασίζει ποτέ σε λάθος τιμή λέγεται *ασφαλής αλγόριθμος* (safe algorithm).

Στον αλγόριθμο CPA, οι παίκτες χρησιμοποιούν μόνο τοπικές πληροφορίες. Αυτό καθιστά τον CPA ιδανικό για *ad hoc* δίκτυα, όπου η τοπολογική γνώση του κάθε παίκτη περιορίζεται στην γειτονιά του. Ο CPA είναι πιθανώς ο μόνος γνωστός αλγόριθμος Εκπομπής για το μοντέλου τοπικά φραγμένου αντιπάλου, ο οποίος δεν χρησιμοποιεί γνώση της τοπολογίας του δικτύου.

Πρωτόκολλο 1: ***Certified Propagation Algorithm (CPA)***

Είσοδος:
(Για κάθε παίκτη v) Ταυτότητα του διανομέα D, ταυτότητες γειτόνων του v, παράμετρος αντιπάλου t.
(Για τον διανομέα D) Αρχική τιμή x_D.
Μορφή μηνυμάτων: Τιμή $x \in X$, όπου X ο χώρος μηνυμάτων.

Κώδικας του D: Στείλε $x_D \in X$ σε όλους τους γείτονες, *αποφάσισε* στο x_D και τερμάτισε.

Κώδικας του $v \in \mathcal{N}(D)$: Αφού λάβεις την τιμή x_D από τον διανομέα, *αποφάσισε* στο x_D, στείλε το x_D σε όλους τους γείτονές σου και τερμάτισε.

(* *κανόνας πιστοποιημένης διάδοσης* *)

Κώδικας του $v \notin \mathcal{N}(D) \cup D$: Αν λάβεις $t(v)+1$ μηνύματα με τη ν ίδια τιμή x από $t(v)+1$ διαφορετικούς γείτονες, *αποφάσισε* στο x, στείλε το x σε όλους τους γείτονές σου και τερμάτισε.

Ορισμός 8.5 Μέγιστη ανοχή του CPA Για ένα γράφημα G και διανομέα D, $t^{\text{CPA}}_{\text{max}}(G, D)$ το μέγιστο t τέτοιο ώστε ο CPA είναι t-τοπικά ανεκτικός.

Όποτε τα G και D εννούνται από τα συμφραζόμενα, θα γράφουμε απλά $t^{\text{CPA}}_{\text{max}}$ αντί για $t^{\text{CPA}}_{\text{max}}(G, D)$.

Φράγματα και συνθήκες

Ας κάνουμε τώρα μια απλή αλλά χρήσιμη παρατήρηση: Έστω μια γραφοθεωρητική παράμετρο X, δείχνοντας ότι η συνθήκη $t < X$ είναι μια ικανή τοπολογική συνθήκη για να είναι ο CPA t-τοπικά ανεκτικός, παρέχει άμεσα ένα κάτω φράγμα $\lceil X \rceil - 1$ για το $t^{\text{CPA}}_{\text{max}}$. Αντίστοιχα, αναγκαίες συνθήκες παρόμοιας μορφής συνεπάγονται άνω φράγματα για το $t^{\text{CPA}}_{\text{max}}$. Θα χρησιμοποιηθεί συχνά αυτή η σχέση φραγμάτων και συνθηκών σε ολόκληρο το κεφάλαιο.

8.2.4 Κάτω φράγματα στην ανοχή του CPA

Οι Pelc και Peleg [Pelc and Peleg, 2005] ήταν οι πρώτοι που παρουσίασαν μια γραφοθεωρητική παράμετρο $\mathcal{X}(G, D)$, η οποία σχετίζει τον μέγιστο ανεκτό αριθμό από τοπικές διαφθορές με την τοπολογία του γραφήματος. Αυτή η παράμετρος αντιπροσωπεύει τον μέγιστο αριθμό b έτσι ώστε κάθε κόμβος v έχει τουλάχιστον b γείτονες με απόσταση από τον D μικρότερη από εκείνη του . Δίνουν μια ικανή συνθήκη σχετική με την ανοχή του CPA, συγκεκριμένα $\mathcal{X}(G, D) \geq 2t + 1$. Η συνθήκη αυτή υπονοεί ότι οι κόμβοι του γραφήματος G μπορούν να διαταχθούν σε επίπεδα ως προς την απόστασή τους από τον διανομέα· με το πρώτο επίπεδο να είναι οι γειτονιά του διανομέα, και κάθε κόμβο στο επίπεδο k να έχει τουλάχιστον $2t + 1$ γείτονες στο επίπεδο $k - 1$.

Αυτό με τη σειρά του συνεπάγεται ότι κάθε κόμβος σε απόσταση k από τον D (επίπεδο k) αποφασίζει στον k-οστό γύρο, επειδή σίγουρα θα λάβει τουλαχιστον $t+1$ ορθλές τιμές από τίμιους παίκτες του επιπέδου $k-1$. Ωστόσο, όπως αποδεικνύεται στην ίδια δουλειά αυτή η συνθήκη δεν είναι αναγκαία, επειδή ένας κόμβος στο επίπεδο k μπορεί επίσης να συλλέξει ορθές τιμές από γείτονές του στο επίπεδο k ή $k+1$, και έτσι να συμπληρώσει τον απαραίτητο αριθμό των $t+1$ αντιγράφων τις ίδιας τιμής. Με άλλα λόγια, η τιμή $\lceil \mathcal{X}/2 \rceil - 1$ είναι ένα κάτω φράγμα για την μέγιστη ανοχή του CPA αλλά όχι ακριβές.

Νέα παράμετρος-φράγμα για τη Μέγιστη Ανοχή του CPA

Σκοπεύοντας να εξάγουμε ακριβέστερα φράγματα για το $t_{\max}^{\text{CPA}}$ εισάγουμε την έννοια της *ελάχιστης k-διάταξης επιπέδων* ενός γραφήματος, η οποία οριζόταν έμμεσα στο [Pelc and Peleg, 2005]. Διαισθητικά μια ελάχιστη k-διάταξη επιπέδων είναι μια διαμέριση των κόμβων σε αριθμημένα επίπεδα-σύνολα, τέτοια ώστε κάθε κόμβος έχει τουλάχιστον k γείτονες σε προηγούμενα επίπεδα και ανήκει στο ελάχιστο επίπεδο για το οποίο ικανοποιείται αυτή η ιδιότητα. Τυπικά έχουμε:

Ορισμός 8.6 Μία *Ελάχιστη k-Διάταξη Επιπέδων* $\mathcal{L}_k(G, D)$ ενός γραφήματος $G = (V, E)$ με διανομέα D είναι μια διαμέριση $V \setminus \{D\} = \bigcup_{i=1}^{m} L_i,\ m \in \mathrm{N}$ τ.ώ.,

$$\begin{aligned} L_1 =& \mathcal{N}(D), \\ L_2 =& \{v \in V \setminus\ L_1 : |\mathcal{N}(v) \cap L_1| \geq k\} \\ & \vdots \\ L_m =& \{v \in V \setminus \bigcup_{j=1}^{m-1} L_j : |\mathcal{N}(v) \cap \bigcup_{j=1}^{m-1} L_j| \geq k\} \end{aligned}$$

Στη συνέχεια ορίζουμε την έννοια της *ασθενούς k-διάταξης επιπέδων* η οποία είναι χρήσιμη για την απλότητα των αποδείξεων, καταργώντας την απαίτηση να ανήκουν οι κόμβοι στο ελάχιστο επίπεδο που δίνει τις επιθυμητές ιδιότητες.

Ορισμός 8.7 Μία *Ασθενής k-Διάταξη Επιπέδων* ενός γραφήματος $G = (V, E)$ με διανομέα D είναι μια διαμέριση $V \setminus \{D\} = \bigcup_{i=1}^{m} L_i,\ m \in \mathrm{N}$ τ.ώ.,

$$L_1 = \mathcal{N}(D), \qquad \forall v \in L_i : |\mathcal{N}(v) \cap \bigcup_{j=1}^{i-1} L_j| \geq k$$

Ιδιότητες των k-διατάξεων επιπέδων

Σημειώνεται ότι ενώ μπορεί να υπάρχουν πολλές διαφορετικές ασθενείς k-διατάξεις επιπέδων ενός γραφήματος, η ελάχιστη k-διάταξη επιπέδων είναι μοναδική, όπως μπορεί να δειχθεί εύκολα με επαγωγή. Παρατηρούμε επίσης ότι μία ασθενής k-διάταξη επιπέδων μπορεί εύκολα να

μετατραπεί σε μια ελάχιστη· για να δείξουμε αυτό χρησιμοποιούμε την έννοια του *αργοπορημένου κόμβου*:

Έστω μία ασθενής k-διάταξη επιπέδων $\mathcal{L}$: $V = \bigcup_{i=1}^{m} L_i,\ m \in \mathrm{N}$, θα ονομάζουμε έναν κόμβο $u \in L_h \in \mathcal{L}$ *αργοπορημένο κόμβο* της $\mathcal{L}$ αν $\exists\ d$ with $1 < d < h \leq m$ τ.ώ. $|\mathcal{N}(u) \cap \bigcup_{j=1}^{d-1} L_j| \geq k$. Το παρακάτω πόρισμα συνεπάγεται άμεσα από τους προηγούμενους ορισμούς.

Πόρισμα 8.8 Μια ασθενής k-διάταξη επιπέδων χωρίς αργοπορημένους κόμβους είναι ελάχιστη k-διάταξη επιπέδων.

Λόγω του παραπάνω πορίσματος, για οποιαδήποτε ασθενή k-διάταξη επιπέδων $\mathcal{L}$ μπορούμε να κατασκευάσουμε μια ελάχιστη k-διάταξη επιπέδων $\mathcal{L}_k$ μετακινώντας επαναλαμβανόμενα κάθε αργοπορημένο κόμβο στο χαμηλότερο επίπεδο έτσι ώστε η διαμέριση να παραμένει ασθενής k-διάταξη επιπέδων. Επομένως ισχύει το παρακάτω.

Πρόταση 8.9 Έστω ένα γράφημα G με διανομέα D, για κάθε $k \in \mathrm{N}$, αν υπάρχει ασθενής k-διάταξη επιπέδων για τα G, D τότε υπάρχει μοναδική ελάχιστη k-διάταξη επιπέδων για τα G, D.

Απόδειξη. Αποδεικνύουμε ότι αν αλλάξουμε την διαμέριση με συγκεκριμένο τρόπο τότε αυτή παραμένει ασθενής k-διάταξη επιπέδων και τελικά κατασκευάζεται μία ελάχιστη k-διάταξη επιπέδων $\mathcal{L}_k(G, D)$. Θα χρησιμοποιήσουμε τον ακόλουθο ισχυρισμό,

Αρχικά παρατηρούμε ότι αν υπάρχει μαι ασθενής k-διάταξη επιπέδων $\mathcal{L}$: $V = \bigcup_{i=1}^{m} L_i, m \in \mathrm{N}$ με $1 < d < h \leq m$, τότε για τυχαίο $u \in L_h$ (αργοπορημένο κόμβο) με $|\mathcal{N}(u) \cap \bigcup_{j=1}^{d-1} L_j| \geq k$, η διαμέριση $\mathcal{L}'$:

$$V = L_1 \cup L_2 \cup \ldots \cup \{L_d \cup \{u\}\} \cup \cdots \cup \{L_h \setminus \{u\}\} \cup \cdots \cup L_m = \bigcup_{i=1}^{m} L'_i$$

είναι επίσης μία ασθενής k-διάταξη επιπέδων.

Στηριζόμενοι στην προηγούμενη παρατήρηση, δοθείσας μιας οποιασδήποτε ασθενούς k-διάταξης επιπέδων $\mathcal{L}$ μπορούμε να κατασκευάσουμε μία ελάχιστη k-διάταξη επιπέδων $\mathcal{L}_k$, μετακινώντας όλους τους αργοπορημένους κόμβους στο ελάχιστο επίπεδο για το οποίο η διαμέριση παραμένει ασθενής k-διάταξη επιπέδων. Συγκεκριμένα,

Δοθείσας μίας ασθενούς k-διάταξης επιπέδων $\mathcal{L} : V = \bigcup_{i=1}^{m} L_i$, για κάθε κόμβο v, μετακινούμε τον v στο σύνολο L_i τ.ώ.

$$i = \min \left\{ d \in \{1, \ldots, m\} \;\middle|\; |\mathcal{N}(v) \cap \bigcup_{j=1}^{d-1} L_j| \geq k \right\}$$

Επιπλέον, όποτε μετακινούμε έναν αργοπορημένο κόμβο πρέπει να ελέγξουμε όλους τους υπόλοιπους κόμβους που μπορεί να μετατράπηκαν σε αργοπορημένοι λόγω αυτής της κίνησης. Η διαδικασία αυτή τερματίζει σε πολυωνυμικό αριθμό βημάτων και η τελική διαμέριση δεν περιέχει αργοπορημένους κόμβους. Επομένως η διαμέριση που προκύπτει είναι μία ελάχιστη k-διάταξη επιπέδων σύμφωνα με το πόρισμα.

Ως προς την μοναδικότητα της ελάχιστης k-διάταξης επιπέδων $\mathcal{L}_k$, υποθέτουμε ότι για το γράφημα G με διανομέα D υπάρχουν δύο διαφορετικές ελάχιστες k-διατάξεις επιπέδων $\mathcal{L} = \{L_1, \cdots, L_m\}, \mathcal{L}' = \{L'_1, \cdots, L'_h\}$. Από τον ορισμό ισχύει ότι $L_1 = L'_1$. Έστω ότι i είναι ο ελάχιστος ακέραιος ώστε $L_i \neq L'_i$ και υποθέτουμε χωρίς βλάβη της γενικότητας ότι $\exists v$, τ.ώ. $v \in L_i$ και $v \notin L'_i$. Τότε είναι προφανές ότι ο v είναι ένας αργοπορημένος κόμβος της L'_i, και έτσι η L'_i δεν είναι ελάχιστη k-διάταξη επιπέδων, το οποίο είναι άτοπο.

Ορισμός 8.10 Παράμετρος $\mathcal{K}$ Για γράφημα G με διανομέα D,

$$\mathcal{K}(G, D) \stackrel{def.}{=} \max\{k \in \mathrm{N} \mid \exists \text{ ελάχιστη } k\text{-διάταξη επιπέδων } \mathcal{L}_k(G, D)\}$$

Θεώρημα 8.11 Ικανή συνθήκη Για κάθε γράφημα G με διανομέα D και $t \in \mathrm{N}$, αν $t < \mathcal{K}(G, D)/2$ τότε ο CPA είναι t-τοπικά ανεκτικός.

Απόδειξη. Παρατηρούμε ότι από το $2t < \mathcal{K}(G, D)$ συνεπάγεται η ύπαρξη μιας ελάχιστης $(2t + 1)$-διάταξης επιπέδων $\mathcal{L}_{2t+1}(G, D)$. Έστω $\mathcal{L}_{2t+1}(G, D)$ η διαμέριση $\{L_1, \ldots, L_m\}$ του V, δηλαδή, $V = \bigcup_{i=1}^{m} L_i$. Αρκεί να δείξουμε ότι για $1 \leq i \leq m$, κάθε τίμιος παίκτης $v \in L_i$ αποφασίζει στην τιμή του διανομέα x_D. Με ισχυρή επαγωγή στο i έχουμε :

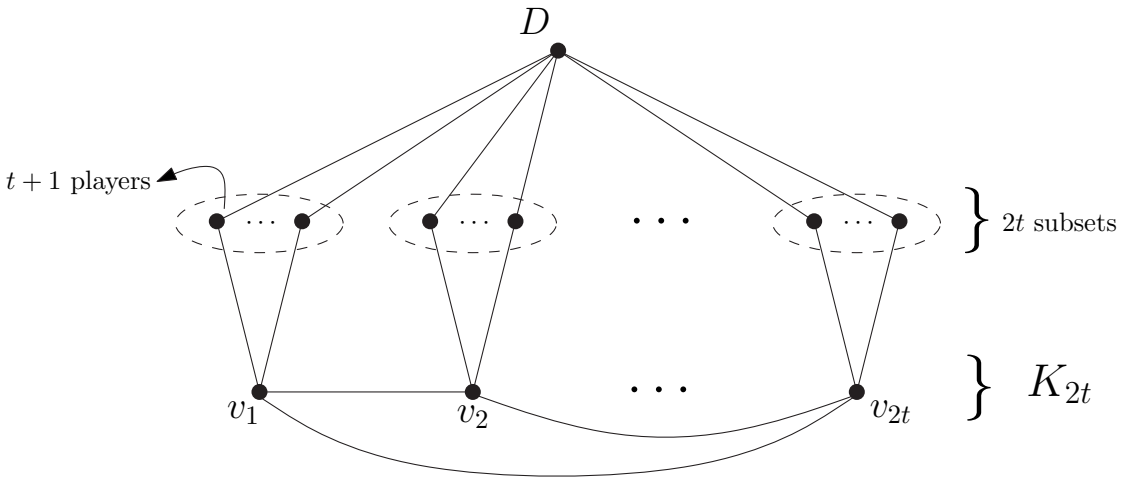
Κάθε τίμιος παίκτης $v \in L_1 = \mathcal{N}(D)$ αποφασίζει τετριμμένα στην τιμή του διανομέα x_D μόλις λάβει αυτή την τιμή. Αν όλοι οι τίμιοι παίκτες $u \in L_i, 1 \leq i \leq h$, αποφασίζουν στην x_D σε κάποιο γύρο, τότε κάθε τίμιος παίκτης $v \in L_{h+1}$ λαμβάνει $|\bigcup_{j=1}^{h} L_j \cap \mathcal{N}(v)| \geq 2t + 1$ μηνύματα από τους γείτονες του που έχουν ήδη αποφασίσει και ανήκουν σε προηγούμενα επίπεδα. Τουλάχιστον $t + 1$ από αυτούς είναι τίμιοι. Επομένως ο v αποφασίζει στην τιμή x_D.

Πόρισμα 8.12 Κάτω φράγμα Για κάθε γράφημα G με διανομέα D ισχύει $t^{\mathrm{CPA}}_{\max} \geq \lceil \mathcal{K}(G, D)/2 \rceil - 1$.

Ακρίβεια του κάτω φράγματος

Στο Θεώρημα 8.11 δείξαμε ότι η συνθήκη $t < \mathcal{K}(G, D)/2$ είνια ικανή για να είναι ο CPA t-τοπικά ανεκτικός· στη συνέχεια αποδεικνύουμε ότι η συνθήκη αυτή δεν είναι και αναγκαία. Διαισθητικά, ο λόγος είναι ότι η τοπολογία του γραφήματος μπορεί να είναι τέτοια ώστε να εμποδίζεται ο αντίπαλος να διαφθείρει ακριβώς t παίκτες σε *κάθε* γειτονιά του γραφήματος και επομένως μερικοί παίκτες θα αποφασίσουν σωστά εκτελώντας τον CPA ακόμα και αν έχουν μόνο $t + 1$ γείτονες σε προηγούμενα επίπεδα.

Πρόταση 8.13 Υπάρχει μια οικογένεια στιγμιοτύπων (G, D), τέτοια ώστε ο CPA είναι $(\mathcal{K}(G, D) - 1)$-τοπικά ανεκτικός για το (G, D).



Σχήμα 8.3: Γράφημα με $\mathcal{K}(G, D) = t + 1$, για το οποίο ο CPA είναι t-τοπικά ανεκτικός.

Απόδειξη. Στο Σχήμα 8.3 βλέπουμε μια τέτοια οικογένεια στιγμιοτύπων για κάθε τιμή του t. Σε αυτό το στιγμιότυπο, η γειτονιά του D αποτελείται από $2t^2 + 2t$ κόμβους, οι κόμβοι $v_1, \ldots, v_{2t}$ αποτελούν μια κλίκα μεγέθους $2t$ και είναι συνδεδεμένοι με κόμβους στο σύνολο $\mathcal{N}(D)$ όπως φαίνεται στο σχήμα. Εύκολα ελέγχουμε ότι $t = \mathcal{K}(G, D) - 1$. Αν τρέξουμε τον CPA στο G τότε κάθε παίκτης $v_i \in \{v_1, \ldots, v_{2t}\}$ λαμβάνει M ορθά μηνύματα, με

$$M = M_A + M_B \tag{1}$$

όπου, M_A : ο αριθμός των μηνυμάτων που έλαβε από το σύνολο $\mathcal{N}(D)$ και
M_B : ο αριθμός των μηνυμάτων που έλαβε από το σύνολο $B = \{v_1, \ldots, v_{2t}\} \setminus \{v_i\}$.

Έστω $T_i = T \cap \mathcal{N}(D) \cap \mathcal{N}(v_i)$ το σύνολο των προδοτών οι οποίοι είναι κοινοί γείτονες των D and v_i. Τότε,

$$M_A = |\mathcal{N}(D) \cap \mathcal{N}(v_i) \setminus T_i| = t + 1 - |T_i| \tag{2}$$

Για να υπολογίσουμε τον αριθμό των ορθών μηνυμάτων που λαμβάνει ο v_i από άλλους παίκτες στο B, ορίζουμε τα σύνολα:

$$C_{B_1} = \{v \in B \mid v \text{ έλαβε το πολύ } t \text{ μηνύματα από το } \mathcal{N}(D)\ \}$$
$$C_{B_2} = \{v \in B \mid v \text{ είναι διεφθαρμένος}\ \}$$
$$C_B = C_{B_1} \cup C_{B_2}$$

Παρατηρούμε ότι το C_{B_1} γίνεται μέγιστης πληθικότητας αν ο αντίπαλος διαφθείρει ακριβώς ένα παίκτη σε κάθε σύνολο $\mathcal{N}(v_j) \cap \mathcal{N}(D), \forall v_j \in B$. Επομένως,

$$\max_{T:t\text{-τοπικό}} |C_{B_1}| = \max_{T:t\text{-τοπικό}} |T \cap (\mathcal{N}(D) \setminus \mathcal{N}(v_i))| = t - |T_i|$$

Επίσης $|C_{B_2}| \leq t - |T_i|$ επειδή τα B και $\mathcal{N}(v_i) \cap \mathcal{N}(D)$ σχηματίζουν τη γειτονιά του v_i όπου οι διαφθορές μπορεί να είναι το πολύ t. Στη συνέχεια υπολογίζουμε ένα άνω φράγμα για το C_B.

$$|C_B| = |C_{B_1} \cup C_{B_2}| \leq |C_{B_1}| + |C_{B_2}| \leq (t - |T_i|) + (t - |T_i|) = 2t - 2|T_i|$$

άρα,

$$M_B = 2t - 1 - |C_B| = 2t - 1 - 2t + 2|T_i| = 2|T_i| - 1 \tag{3}$$

Τελικά υπολογίζουμε τον συνολικό αριθμό μηνυμάτων M,

$$(1), (2), (3) \Rightarrow M = M_A + M_B \geq t + 1 - |T_i| + 2|T_i| - 1 = \\ = t + |T_i|$$

Για οποιοδήποτε v_i, αν $|T_i| > 0$ τότε $M \geq t + 1$. Αλλιώς θα ισχύει $|T_i| = 0$ και ο v_i θα λάβει $t + 1$ ορθά μηνύματα από το $\mathcal{N}(D)$. Επομένως ο CPA πετυχαίνει Εκπομπή στο στιγμιότυπο (G, D).

8.2.5 Άνω φράγμα για την ανοχή του CPA

Στην προηγούμενη ενότητα δείξαμε ότι $t_{\max}^{\text{CPA}} \geq \lceil \mathcal{K}(G,D)/2 \rceil - 1$, επίσης παρουσιάσαμε περιπτώσεις όπου $\mathcal{K}(G,D) - 1$ προδότες μπορούν να γίνουν ανεκτοί από τον CPA. Σε αυτήν την ενότητα θα δείξουμε ότι ο αριθμός αυτός είναι και ο μέγιστος που μπορεί να γίνει ανεκτός, δηλαδή, δείχνουμε ότι η τιμή $\mathcal{K}(G,D) - 1$ είναι ένα άνω φράγμα στον τοπικό αριθμό προδοτών για κάθε G και D. Το πετυχαίνουμε το προηγούμενο αποδεικνύοντας μία αναγκαία συνθήκη για να είναι ο CPA t-τοπικά ανεκτικός.

Θεώρημα 8.14 Αναγκαία συνθήκη Για κάθε γράφημα G με διανομέα D και $t \geq \mathcal{K}(G,D)$, ο CPA δεν είναι t-τοπικά ανεκτικός.

Απόδειξη. Υποθέτουμε ότι ο CPA είναι t-τοπικά ανεκτικός, με $t \geq \mathcal{K}(G,D)$. Αφού ο CPA είναι t-τοπικά ανεκτικός πρέπει να υπάρχει θετικός ακέραιος s, τέτοιος ώστε ο αλγόριθμος τερματίζει μετά από s βήματα στο στιγμιότυπο (G,D). Περιγράφουμε την λειτουργία του CPA στο γράφημα G με σύνολα κόμβων που αποφασίζουν. Έστω L_i το σύνολο των κόμβων που αποφασίζουν στον i-στό γύρο. Αφού κάθε κόμβος $v \in L_i$ αποφασίζει στον i-στό γύρο ισχύει ότι ο v έχει τουλάχιστον $t+1$ γείτονες στα σύνολα $L_1, \ldots, L_{i-1}$. Δηλαδή,

$$\forall v \in L_i \Rightarrow |\mathcal{N}(v) \cap \bigcup_{j=1}^{i-1} L_j| \geq t+1.$$

Παρατηρούμε ότι η παραπάνω ακολουθία συνόλων είναι μία ασθενής $(t+1)$-διάταξη επιπέδων για τα G, D. Από αυτήν την παρατήρηση και την Πρόταση 8.9 έχουμε ότι πρέπει να υπάρχει μια ελάχιστη $(t+1)$-διάταξη επιπέδων για τα G, D. Αυτό όμως μας οδηγεί σε αντίφαση αφού υποθέσαμε ότι $t \geq \mathcal{K}(G,D)$.

Πόρισμα 8.15 Άνω φράγμα για το $t_{\max}^{\text{CPA}}$ Για κάθε γράφημα G με διανομέα D ισχύει ότι $t_{\max}^{\text{CPA}} < \mathcal{K}(G,D)$

Σύγκριση με την παράμετρο των Ichimura-Shigeno

Στην εργασία [Ichimura and Shigeno, 2010], οι Ichimura και Shigeno εισάγουν την γραφοθεωρητική παράμετρο $\widetilde{\mathcal{X}}(G,D)$ η οποία μπορεί να χρησιμοποιηθεί για την εξαγωγή μιας ικανής συνθήκης για την ανοχή του CPA. Για ένα γράφημα $G = (V,E)$ με διανομέα D, θεωρούν μια ολική διάταξη των κόμβων $\sigma = (v_1, v_2, \ldots)$ του συνόλου $V \setminus (\mathcal{N}(D) \cup D)$, και χρησιμοποιούν τη συνάρτηση $\delta(W_i, v)$ για τον αριθμό των γειτόνων του κόμβου v στο σύνολο $\mathcal{N}(D) \cup \{v_1, \ldots, v_{i-1}\}$. Η ολική διάταξη σ έχει την ιδιότητα $\forall i,j$, τ.ώ. $1 \leq i < j \leq |V \setminus (\mathcal{N}(D) \cup D)|$ ισχύει ότι $\delta(W_{i-1}, v_i) \geq \delta(W_{i-1}, v_j)$. Αυτή η διάταξη αναφέρεται επίσης στη βιβλιογραφία [Nagamochi and Ibaraki, 2008] σαν *max-back* διάταξη. Έπειτα ορίζουν την παράμετρο

$\widetilde{\mathcal{X}}(G,D) = \min\{\delta(W_{i-1}, v_i) \mid i = 1, 2, \ldots\}$. και αποδεικνύουν ότι είναι μοναδική , δηλαδή, είναι η ίδια για όλες τις max-back διατάξεις. Ουσιαστικά, με την ορολογία που χρησιμοποιούμε εδώ, αποδεικνύουν ότι [2]

$$\lceil \widetilde{\mathcal{X}}(G,D)/2 \rceil - 1 \leq t_{\max}^{\text{CPA}} < \widetilde{\mathcal{X}}(G,D). \tag{1}$$

Ως εκ τούτου, η παράμετρός τους δίνει παρόμοια φράγματα με τη δικιά μας. Στη συνέχεια δείχνουμε παρά τους διαφορετικούς ορισμούς των παραμέτρων $\mathcal{K}(G,D)$ και $\widetilde{\mathcal{X}}(G,D)$, αποδεικνύεται ότι αυτές είναι ίσες.

Πρόταση 8.16 $\mathcal{K}(G,D) = \widetilde{\mathcal{X}}(G,D)$

Απόδειξη. Θεωρούμε την max-back διάταξη $\sigma = (v_1, v_2, \ldots)$. Παρατηρούμε ότι η ακολουθία $\{L_1 = \mathcal{N}(D), L_2 = \{v_1\}, L_3 = \{v_2\}, \ldots\}$ είναι τετριμμένα μια ασθενής $\widetilde{\mathcal{X}}(G,D)$-διάταξη επιπέδων, γιατί η ελάχιστη συνδεσιμότητα μεταξύ ενός επιπέδου και των προηγούμενών του είναι $\widetilde{\mathcal{X}}(G,D)$. Άρα, λόγω της Πρότασης 8.9, υπάρχει μια ελάχιστη $\widetilde{\mathcal{X}}(G,D)$-διάταξη επιπέδων και επομένως $\mathcal{K}(G,D) \geq \widetilde{\mathcal{X}}(G,D)$. Έτσι, συνδυάζοντας την τελευταία ανισότητα με την ανισότητα (1) δείχνουμε το ακόλουθο:

$$t_{\max}^{\text{CPA}} < \widetilde{\mathcal{X}}(G,D) \leq \mathcal{K}(G,D)$$

Αφού η Πρόταση 8.13 υπονοεί ότι υπάρχει ένα γράφημα γαι το οποίο ο CPA είναι $(\mathcal{K}(G,D) - 1)$-τοπικά ανεκτικός η παραπάνω σχέση συνεπάγεται την ισότητα των παραμέτρων $\mathcal{K}(G,D)$ και $\widetilde{\mathcal{X}}(G,D)$. Αν αυτό δεν ίσχυε τότε το $\widetilde{\mathcal{X}}(G,D) < \mathcal{K}(G,D)$ θα σήμαινε ότι $t_{\max}^{\text{CPA}} < \mathcal{K}(G,D) - 1$, το οποίο είναι άτοπο.

Παρόλο που οι δύο παράμετροι $\mathcal{K}(G,D)$ και $\widetilde{\mathcal{X}}(G,D)$ αποδεικνύονται ίσες, το γεγονός ότι η $\mathcal{K}(G,D)$ είναι ορισμένη με διαφορετικό τρόπο οδηγεί σε βελτιωμένη πολυπλοκότητα υπολογισμού της, όπως θα δούμε στην επόμενη ενότητα.

8.2.6 Προσεγγιστικός υπολογισμός της μέγιστης ανοχής του CPA

Ας εξετάσουμε τώρα την προσεγγισιμότητα της *Μέγιστης ανοχής του CPA*· θα σχεδιάσουμε έναν αποδοτικό 2-προσεγγιστικό αλγόριθμο. Αρχικά δείχνουμε πως μπορούμε να ελέγξουμε την ύπαρξη μιας ελάχιστης m-διάταξης επιπέδων, για ένα γράφημα G με διανομέα D, χρησιμοποιώντας μια παραλλαγή του κλασικού αλγόριθμου BFS. Στη συνέχεια, πετυχαίνουμε την προσέγγιση υπολογίζοντας την παράμετρο $\mathcal{K}(G,D)$, μέσω επαναλήψεων του παραπάνω ελέγχου. Ο λόγος προσέγγισης αποδεικνύεται, συνδυάζοντας τα Πορίσματα 8.12 και 8.15.

[2] Παρατηρούμε ότι η συνθήκη $t \leq \widetilde{\mathcal{X}}(G,D)$ παρουσιάζεται σαν αναγκαία στο [Ichimura and Shigeno, 2010]· ωστόσο η απόδειξή τους μπορεί εύκολα να μετατραπεί για να αποδειχθεί το πιο ακριβές φράγμα $t < \widetilde{\mathcal{X}}(G,D)$, όπως φαίνεται στο δεξί μέλος της (1).

Έλεγχος ύπαρξης ελάχιστης m-διάταξης επιπέδων για τα (G, D)

Για είσοδο (G, D, m) κάνουμε τα ακόλουθα:

1. Αναθέτουμε ένα μετρητή σε κάθε κόμβο αρχικοποιημένο στο 0.
2. Εισάγουμε σε μία ουρά τον κόμβο-διανομέα και όλους τους γείτονές του.
3. Εξάγουμε έναν κόμβο από την ουρά και αυξάνουμε κατά 1 τους μετρητές όλων των γειτόνων του. Εισάγουμε έναν τέτοιο γείτονα στην ουρά μόνον αν ο μετρητής του είναι τουλάχιστον m.
4. Επαναλαμβάνουμε το βήμα 3 μέχρι να αδειάσει η ουρά.
5. Αν όλοι οι κόμβοι έχουν εισαχθεί στην ουρά τότε δίνουμε έξοδο '*Αληθές*' (υπάρχει μία ελάχιστη m-διάταξη επιπέδων), αλλιώς δίνουμε έξοδο '*Ψευδές*'.

Παρατηρούμε ότι ο παραπάνω αλγόριθμος μπορεί να τροποποιηθεί για να υπολογίζει την ελάχιστη m-διάταξη επιπέδων $\mathcal{L}_m(G, D)$.

2-Προσέγγιση της παραμέτρου $t_{\max}^{\text{CPA}}$

1. Υπολόγισε την παράμετρο $\mathcal{K}(G, D)$: Αφού $\mathcal{K}(G, D) < \min_{v \in V \setminus (\mathcal{N}(D) \cup D)} \deg(v) = \delta$, η ακριβής τιμή της παραμέτρου $\mathcal{K}(G, D)$ υπολογίζεται με $\log \delta$ επαναλήψεις του ελέγχου ύπαρξης, χρησιμοποιώντας δυαδική αναζήτηση.
2. Επέστρεψε $\lceil \mathcal{K}(G, D)/2 \rceil - 1$

Το $t \geq \mathcal{K}(G, D)$ σημαίνει ότι ο CPA δεν είναι t-τοπικά ανεκτικός, επομένως ισχύει ότι $t_{\max}^{\text{CPA}} < \mathcal{K}(G, D)$, συνεπώς, η επιστρεφόμενη τιμή είναι τουλάχιστον $\lceil t_{\max}^{\text{CPA}}/2 \rceil - 1$.

Ένα *ακριβές παράδειγμα* (tight example) για το λόγο προσέγγισης αυτού του αλγορίθμου στην πραγματικότητα δίνεται από την οικογένεια στιγμιοτύπων που παρουσιάστηκε νωρίτερα στο Σχήμα 8.3 στο οποίο παρουσιάσαμε ένα γράφημα G για το οποίο $\mathcal{K}(G, D) = t + 1$ και ο CPA είναι t-τοπικά ανεκτικός.

Η πολυπλοκότητα του παραπάνω προσεγγιστικού αλγορίθμου δίνεται προφανώς από την πολυπλοκότητα υπολογισμού του $\mathcal{K}(G, D)$. Όπως εξηγήσαμε παραπάνω, ο αλγόριθμος χρειάζεται το πολύ $\log \delta$ εκτελέσεις του ελέγχου ύπαρξης ελάχιστης διάταξης επιπέδων. Το τελευταίο απαιτεί πολυπλοκότητα χρόνου $O(|E|)$ (ίδια πολυπλοκότητα με τον αλγόριθμο BFS). Συνολικά, έχουμε ότι η πολυπλοκότητα χρόνου του αλγορίθμου είναι της τάξης $O(|E| \log \delta)$, η οποία είναι σημαντικά βελτιωμένη σε σχέση με την πολυπλοκότητα υπολογισμού της ισοδύναμης παραμέτρου $\widetilde{\mathcal{X}}(G, D)$, η οποία είναι $O(|V|(|V| + |E|))$ όπως αναφέρεται στο [Ichimura and Shigeno, 2010] .

8.2.7 Ακριβής προσδιορισμός της μέγιστης ανοχής του CPA

Σε αυτήν την ενότητα παρουσιάζουμε μια διαδικασία για τον ακριβή υπολογισμό του $t_{\max}^{\text{CPA}}$. Για τον σκοπό αυτό εισάγουμε δύο νέες γραφοθεωρητικές παραμέτρους. Για ένα πιθανό σύνολο διεφθαρμένων παικτών (t-τοπικό) T και γράφημα $G = (V, E)$ θα συμβολίζουμε με $G_{\bar{T}} = (V \setminus T, E')$ το επαγόμενο υπογράφημα (node induced) του G στο σύνολο κόμβων $V \setminus T$.

Ορισμός 8.17 Για γράφημα G με διανομέα D και θετικό ακέραιο t, το *t-όριο ασφαλείας* είναι η ποσότητα $\mathcal{M}(G, D, t) = \min_{T:\ t\text{-τοπικό}} \mathcal{K}(G_{\bar{T}}, D)$.

Θεώρημα 8.18 Αναγκαία και ικανή συνθήκη Για γράφημα $G = (V, E)$ και διανομέα D, ο CPA είναι t-τοπικά ανεκτικός αν και μόνον αν $\mathcal{M}(G, D, t) \geq t + 1$.

Απόδειξη. ($\Leftarrow$) Έστω $\mathcal{M}(G, D, t) \geq t + 1$ και $T \subseteq V \setminus D$ ένα οποιοδήποτε t-τοπικό σύνολο διεφθαρμένων κόμβων. Από τον ορισμό ισχύει $\mathcal{K}(G_{\bar{T}}, D) \geq t + 1$. Επομένως υπάρχει μία ελάχιστη $(t + 1)$-διάταξη επιπέδων $\mathcal{L}_{t+1}(G_{\bar{T}}, D) = \{L_1, \ldots, L_m\}$. Άρα οποιοσδήποτε τίμιος παίκτης v έχει τουλάχιστον $t + 1$ τίμιους γείτονες σε επίπεδα προηγούμενα του $\mathcal{L}_{t+1}(G_{\bar{T}}, D)$· με μια απλή επαγωγή δείχνουμε ότι ο v θα αποφασίσει στην τιμή του διανομέα x_D.

($\Rightarrow$) Εάν ο CPA είναι t-τοπικά ανεκτικός τότε για οποιοδήποτε t-τοπικό σύνολο διεφθαρμένων T ισχύει ότι κάθε τίμιος παίκτης στο υπογράφημα $G_{\bar{T}}$ αποφασίζει στο x_D. Έστω ότι ο συνολικός αριθμός γύρων μέχρι τον τερματισμό του πρωτοκόλλου είναι $m \in \mathrm{N}$. Ορίζουμε την ακολουθία συνόλων $L_i = \{v \in V \setminus T \mid v \text{ αποφασίζει στον γύρο } i \text{ του CPA }\}$, $i \in \{1, \ldots, m\}$. Δείχνουμε με επαγωγή ότι η ακολουθία $(L_i)_{i=1}^{m}$ είναι η (μοναδική) ελάχιστη $(t + 1)$-διάταξη επιπέδων του γραφήματος $G_{\bar{T}}$ με διανομέα D. Αρχικά σημειώνουμε ότι $L_1 = \mathcal{N}(D) \setminus T$ γιατί οι παίκτες που αποφασίζουν στον γύρο 1 είναι ακριβώς αυτοί που ανήκουν στη γειτονιά του διανομέα. Για τη βάση της επαγωγής, παρατηρούμε ότι $L_2 = \{v \in V \setminus T \mid \mathcal{N}(v) \cap L_1 \geq t + 1\}$ καθώς οι παικτες που αποφασίζουν στον γύρο 2 είναι ακριβώς αυτοί που θα λάβουν $t + 1$ όμοια μηνύματα από παίκτες που έχουν αποφασίσει στον γύρο 1. Υποθέτουμε ότι $L_k = \{v \in V \setminus \{T \cup \bigcup_{j=1}^{k-1} L_j\} : |\mathcal{N}(v) \cap \bigcup_{j=1}^{k-1} L_j| \geq t + 1\}$ και παρατηρούμε ότι ισχύει το

$$L_{k+1} = \{v \in V \setminus \{T \cup \bigcup_{j=1}^{k} L_j\} : |\mathcal{N}(v) \cap \bigcup_{j=1}^{k} L_j| \geq t + 1\}$$

λόγω του ότι οι παίκτες που αποφασίζουν στον γύρο $k + 1$ είναι ακριβώς οι παίκτες που λαμβάνουν τουλάχιστον $t + 1$ μηνύματα από παίκτες πυ έχουν αποφασίσει σε προηγούμενα επίπεδα. Αφού τα παραπάνω ισχύουν για κάθε T, ο ισχυρισμός αποδεικνύεται.

Για τον ακριβή προσδιορισμό της Μέγιστης Ανοχής του CPA $t_{\max}^{\text{CPA}}$ εισάγουμε την παράμετρο,

$$\mathcal{T}(G, D) = \max\{t \in \mathrm{N} \mid \mathcal{M}(G, D, t) \geq t + 1\}$$

Από την παραπάνω συζήτηση θα πρέπει να είναι σαφές ότι $\mathcal{T}(G, D)$ ταυτίζεται με την Μέγιστη Ανοχή του CPA.

Πόρισμα 8.19 $t_{\max}^{\text{CPA}}(G, D) = \mathcal{T}(G, D)$

Ένας απλός αλγόριθμος για να υπολογίσουμε το t-όριο ασφαλείας $\mathcal{M}(G, D, t)$ απαιτεί εκθετικό χρόνο καθώς πρέπει να λάβουμε υπόψιν όλα τα t-τοπικά σύνολα και να υπολογίσουμε το $\mathcal{K}(G_{\bar{T}}, D)$ όπως αυτό φαίνεται στην Ενότητα 8.2.6). Σημειώνεται ότι μια διαφορετική ικανή και αναγκαία συνθήκη για να είναι ο CPA t-τοπικά ανεκτικός δόθηκε ανεξάρτητα στο [Tseng et al., 2015]. Ωστόσο, μόνο ένας αλγόριθμος υπερεκθετικού χρόνου προκύπτει έμμεσα για τον έλεγχο αυτής της συνθήκης από την εργασία [Tseng et al., 2015] (δεν δίνεται αλγόριθμος στην εργασία αυτή).

Επιπλέον, για τον υπολογισμό του $t_{\max}^{\text{CPA}} = \mathcal{T}(G, D)$ αρκεί να εκτελεστούν το πολύ $\log \delta$ υπολογισμούς της παραμέτρου $\mathcal{M}(G, D, t)$ όπου, δ είναι ο ελάχιστος βαθμός μεταξύ όλων των κόμβων στο σύνολο $V \setminus (\mathcal{N}(D) \cup D)$.

8.2.8 Μοναδικότητα του CPA σε *Ad Hoc* δίκτυα

Βασιζόμενοι στην ικανή και αναγκαία συνθήκη για την t-τοπική ανεκτικότητα του CPA στο γράφημα Gμε διανομέα D μπορούμε τώρα να αποδείξουμε την *Εικασία της Μοναδικότητας του CPA* για *ad hoc* δίκτυα, που είχε τεθεί σαν ανοιχτό πρόβλημα στο [Pelc and Peleg, 2005]. Η εικασία αναφέρει ότι δεν υπάρχει αλγόριθμος Εκπομπής που να μπορεί να ανεχθεί περισσότερες τοπικές διαφθορές από τον CPA σε δίκτυα άγνωστης τοπολογίας (*ad hoc*).

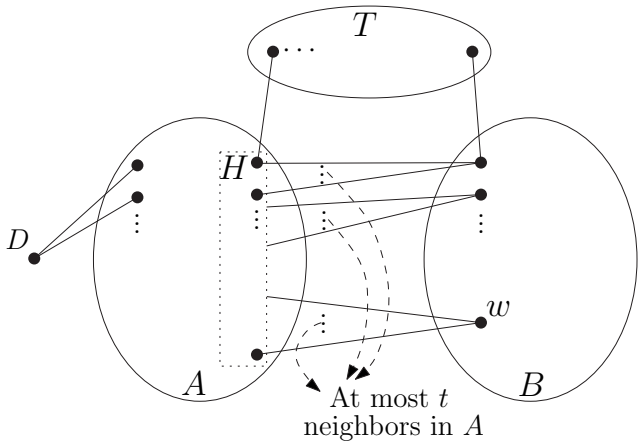
Λαμβάνουμε υπόψιν μόνο την κλάση των *t-τοπικά ασφαλών* αλγορίθμων Εκπομπής μέσω των οποίων κανένας παίκτης δεν αποφασίζει ποτέ σε λάθος τιμή υπό οποιοδήποτε t-τοπικό σύνολο διαφθοράς (βλέπε [Pelc and Peleg, 2005]).

Εργαζόμαστε στο μοντέλο *ad hoc* δικτύων, π.χ. [Pelc and Peleg, 2005]. Συγκεκριμένα υποθέτουμε ότι οι κόμβοι γνωρίζουν μόνο τις δικές τους ταυτότητες, τις ταυτότητες των γειτόνων τους και την ταυτότητα του διανομέα. Ένας κατανεμημένος αλγόριθμος Εκπομπής που λειτουργεί κάτω από αυτές τις παραδοχές ονομάζεται *ad hoc αλγόριθμος*.

Θεώρημα 8.20 Έστω $\mathcal{A}$ ένας *ad hoc* αλγόριθμος t-τοπικά ασφαλής. Αν ο $\mathcal{A}$ είναι t-τοπικά ανεκτικός για γράφημα G με διανομέα D τότε ο CPA είναι t-τοπικά ανεκτικός για τα G, D.

Απόδειξη. Από το Θεώρημα 8.18 έχουμε ότι, αν ο CPA δεν είναι t-τοπικά ανεκτικός στο (G, D) τότε, $\mathcal{M}(G, D, t) = \min_{T:\ t\text{-τοπικό}} \mathcal{K}(G_{\bar{T}}, D) \leq t$ το οποίο συνεπάγεται ότι υπάρχει ένα t-τοπικό σύνολο T τ.ώ. στο υπογράφημα $G_{\bar{T}}$ δεν υπάρχει ελάχιστη $(t+1)$-διάταξη επιπέδων. Από τον ορισμό της $(t+1)$-διάταξης επιπέδων έχουμε ότι δοθείσας μιας ακολουθίας υποσυνόλων του συνόλου κόμβων $V_{\bar{T}} = V \setminus (T \cup \{D\})$,

$$L_1 = \mathcal{N}_{G_{\bar{T}}}(D),\ \ L_i = \{v \in V_{\bar{T}} \setminus \bigcup_{j=1}^{i-1} L_j : |\mathcal{N}_{G_{\bar{T}}}(v) \cap \bigcup_{j=1}^{i-1} L_j| \geq t+1\}, 2 \leq i \leq m$$



Σχήμα 8.4: Διαμέριση του G στα υπογραφήματα A, B, T

υπάρχει $h \in \mathrm{N}$ s.t. $\forall j \geq h,\ L_j = \emptyset$ και $\bigcup_{i=1}^{h} L_i \subsetneq V_{\bar{T}}$. Συμβολίζουμε με $h_{\min}$ το ελάχιστο $h \in \mathrm{N}$ με την παραπάνω ιδιότητα. Χωρίς βλάβη της γενικότητας μπορούμε να υποθέσουμε ότι $h_{\min} \geq 2$, επειδή $h = 1$ συνεπάγεται ότι στο γράφημα $G_{\bar{T}}$ ο διανομέας D δεν είναι συνδεδεμένος με το υπόλοιπο γράφημα το οποίο με τη σειρά του τετριμμένα συνεπάγεται ότι κανένας αλγόριθμος δεν μπορεί να πετύχει Εκπομπή υπό την διαφθορά του συνόλου T.

Έστω $A = \bigcup_{i=1}^{h_{\min}} L_i$ και $B = V_{\bar{T}} \setminus A$. Είναι προφανές από τον ορισμό της ελάχιστης $(t+1)$-διάταξης επιπέδων ότι $\forall w \in B,\ |\mathcal{N}_{G_{\bar{T}}}(w) \cap A| \leq t$. Επιπλέον, το $\bigcup_{i=1}^{h_{\min}} L_i \subsetneq V_{\bar{T}}$ σημαίνει ότι $B \neq \emptyset$. Τελικά χρησιμοποιούμε τον συμβολισμό $H = \bigcup_{w \in B} (\mathcal{N}_{G_{\bar{T}}}(w) \cap A)$ και παρατηρούμε ότι το H συνιστά ένα διαχωριστή (τομή-κόμβων) στο γράφημα $G_{\bar{T}}$ που χωρίζει τον διανομέα D από το υπογράφημα B. Η διαμέριση του γραφήματος G στα τρία υπογραφήματα A, B, T απεικονίζεται στο Σχήμα 8.4.

Έστω το γράφημα G' το οποίο προκύπτει από το G αν αφαιρέσουμε ακμές (u, v) από το σύνολο $E' = \{(u, v) | u, v \in A \cup T\}$ τέτοιες ώστε το σύνολο H γίνεται t-τοπικό στο G' (π.χ. μπορούμε να αφαιρέσουμε όλες τις ακμές μεταξύ κόμβων του συνόλου $A \cup T$). Η ύπαρξη συνόλου ακμών που εγγυάται αυτήν την ιδιότητα προκύπτει από το γεγονός ότι $\forall w \in B,\ |\mathcal{N}_{G_{\bar{T}}}(w) \cap H| \leq t$.

Η συνέχεια της απόδειξης γίνεται με απαγωγή σε άτοπο. Υποθέτουμε ότι υπάρχει ένας t-τοπικά ασφαλής αλγόριθμος Εκπομπής . $\mathcal{A}$ ο οποίος είναι t-τοπικά ανεκτικός στο γράφημα G με διανομέα D. Θεωρούμε τις ακόλουθες εκτελέσεις σ και σ' του αλγορίθμου $\mathcal{A}$,

- Η εκτέλεση σ συμβαίνει στο γράφημα G με διανομέα D, για την τιμή του διανομέα έχουμε ότι $x_D = 0$, το σύνολο διεφθαρμένων παικτών είναι το T και σε κάθε γύρο, όλοι οι παίκτες

αυτού του συνόλου στέλνουν τα ίδια μηνύματα που θα στέλνανε στον αντίστοιχο γύρο της εκτέλεσης σ' όπου το σύνολο T είναι ένα σύνολο τίμιων παικτών.

- Η εκτέλεση σ' συμβαίνει στο γράφημα G' με διανομέα D, για την τιμή του διανομέα έχουμε ότι $x_D = 1$, το σύνολο διεφθαρμένων παικτών είναι το H και σε κάθε γύρο, όλοι οι παίκτες αυτού του συνόλου στέλνουν τα ίδια μηνύματα που θα στέλνανε στον αντίστοιχο γύρο της εκτέλεσης σ όπου το σύνολο H είναι ένα σύνολο τίμιων παικτών.

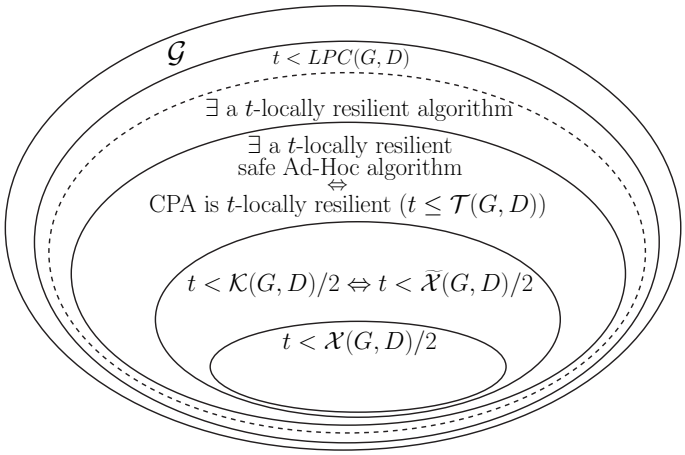
Σημειώνουμε ότι τα σύνολα T, H είναι πιθανά σύνολα διαφθοράς στα γραφήματα G, G' αντίστοιχα καθώς είναι t-τοπικά. Παρατηρούμε ότι το σύνολο $H \cup T$ είναι ένας διαχωριστής που αποσυνδέει τον διανομέα D από το υπογράφημα B και στα δύο γραφήματα G και G' και οι ενέργειες που εκτελούν όλοι οι κόμβοι αυτού του συνόλου-διαχωριστή είναι ακριβώς οι ίδιες και στις δύο εκτελέσεις σ, σ'. Συνεπώς, οι ενέργειες κάθε τίμιου παίκτη $w \in B$ θα είναι πανομοιότυπες και στις δύο εκτελέσεις. Από την υπόθεση μας έχουμε ότι ο αλγόριθμος $\mathcal{A}$ είναι t-τοπικά ανεκτικός στο G με διανομέα D, άρα ο w θα αποφασίσει στην τιμή του διανομέα 0 κατά την εκτέλεση σ στο G. Από τα προηγούμενα, πρέπει να αποφασίσει στην ίδια τιμή και στην εκτέλεση σ' στο γράφημα G'. Ωστόσο, στην τελευταία εκτέλεση η τιμή του διανομέα είναι 1. Αυτό μας δίνει μια αντίφαση αφού υποθέσαμε ότι ο $\mathcal{A}$ είναι t-τοπικά ασφαλής αλγόριθμος.

Παρατηρούμε ότι αν παραλείψουμε την απαίτηση να είναι ο αλγόριθμος t-τοπικά ασφαλής, τότε το θεώρημα δεν ισχύει. Διαισθητικά, μπορούμε να χρησιμοποιήσουμε ένα πρωτόκολλο το οποίο χρησιμοποιεί παραδοχές για της τοπολογικές ιδιότητες του δικτύου έτσι ώστε το πρωτόκολλο αυτό να είναι t-τοπικά ανεκτικό σε μια οικογένεια γραφημάτων που έχουν τις συγκεκριμένες τοπολογικές ιδιότητες.

Πιο τυπικά, στο [Pelc and Peleg, 2005], οι Pelc και Peleg εισήγαγαν έναν άλλον αλγόριθμο, με την ονομασία *Relaxed Propagation Algorithm*(RPA) ο οποίος χρησιμοποιεί γνώση της τοπολογίας του δικτύου και απέδειξαν ότι υπάρχει ένα γράφημα G με διανομέα D για το οποίο ο RPA είναι 1-τοπικά ανεκτικός ενώ ο CPA δεν είναι. Έτσι, αν χρησιμοποιήσουμε τον RPA στο *ad hoc* μοντέλο υποθέτοντας ότι το δίκτυο είναι στην πραγματικότητα το G τότε αυτός ο αλγόριθμος θα είναι 1-τοπικά ανεκτικός για το G με διανομέα D ενώ ο CPA δεν θα είναι. Το γεγονός ότι ο RPA δεν είναι t-τοπικά ασφαλής αλγόριθμος είναι εύκολο να αποδειχθεί. Αυτή η απλή παρατήρηση μας δείχνει ότι το θεώρημα δεν ισχύει αν λάβουμε υπόψιν αλγορίθμους οι οποίοι δεν είναι t-τοπικά ασφαλείς.

8.3 Συμπεράσματα

Από τη στιγμή που η ύπαρξη ενός t-τοπικά ανεκτικού αλγόριθμου Εκπομπής σε ένα γράφημα G με διανομέα D προφανώς εξαρτάται από την τοπολογία του G, για δοθέν αριθμό t μπορούμε να ορίσουμε και να συγκρίνουμε οικογένειες γραφημάτων (με καθορισμένο κόμβο-διανομέα) που καθορίζονται από τις ιδιότητες και τις τοπολογικές συνθήκες που έχουν εμφανιστεί έως τώρα στην βιβλιογραφία συμπεριλαμβανομένων αυτών που ορίστηκαν σε αυτό το κεφάλαιο. Μια επισκόπηση των αντίστοιχων οικογενειών και της μεταξύ τους σχέσης απεικονίζεται στο Σχήμα 8.5.



Σχήμα 8.5: Επισκόπηση συνθηκών σχετιζόμενες με την ύπαρξη t-τοπικά ανεκτικών αλγορίθμων. Οι παράμετροι $LPC(G, D)$ και $\mathcal{X}(G, D)$ ορίζονται στο [Pelc and Peleg, 2005] και η $\widetilde{\mathcal{X}}(G, D)$ στο [Ichimura and Shigeno, 2010]. Οι συνεχείς γραμμές δείχνουν γνήσια υποσύνολα. Οι αντίστοιχοι αγγλικοί χρησιμοποιούνται ως εξής· t-τοπικά ανεκτικός αλγόριθμος: t-locally resilient, ασφαλής αλγόριθμος: safe algorithm

Το γενικό πρόβλημα (διεφθαρμένος κόμβος-διανομέας)

Όπως αναφέραμε, η ορθότητα του CPA στηρίζεται στην υπόθεση ότι ο διανομέας είναι τίμιος. Προκειμένου να αντιμετωπιστεί η περίπτωση στην οποία ο διανομέας είναι διεφθαρμένος μπορούμε να παρατηρήσουμε ότι εάν ο συνολικός αριθμός των προδοτών είναι αυστηρά μικρότερος του $n/3, n = |V|$, και ο αριθμός των προδοτών σε κάθε γειτονιά είναι φραγμένος από το $\min_{D \in V} \mathcal{T}(G, D)$ τότε μπορούμε να πετύχουμε Εκπομπή προσομοιώνοντας οποιοδήποτε πρωτόκολλο για πλήρη γραφήματα ως εξής: Κάθε μετάδοση ενός παίκτη σε πολλούς (ή ακόμη και ενός παίκτη σε έναν άλλον) αντικαθίσταται από μαι εκτέλεση του CPA. Παρατηρούμε ότι η συνθήκη $\min_{D \in V} \mathcal{T}(G, D)$ μπορεί να μην είναι ακριβής σε αυτήν την περίπτωση. Μπορούμε να πάρουμε ένα καλύτερο φράγμα αν ορίσουμε την παράμετρο $\mathcal{M}(G, D, t)$ λαμβάνοντας υπόψιν μόνο πιθανά σύνολα διεφθαρμένων παικτών με πληθικότητα αυστηρά μικρότερη του $n/3$. Συνεπώς, εξάγουμε ένα άνω φράγμα για Εκπομπή με διεφθαρμένο κόμβο-διανομέα, δηλαδή $t \leq \min\left(\lceil n/3 \rceil - 1, \min_{D \in V} \mathcal{T}(G, D)\right)$. Η εξαγωγή ενός ακριβούς φράγματος για τον αριθμό των διαφθορών όπως και η μελέτη για πιο αποδοτικούς αλγορίθμους για το γενικό πρόβλημα είναι ενδιαφέροντα ανοιχτά ερωτήματα.

Άλλα ανοιχτά ερωτήματα που προκύπτουν από τη μελέτη που παρουσιάζεται σε αυτό το κεφάλαιο είναι τα εξής: ο ορισμός μιας αποδοτικά υπολογίσιμης παραμέτρου μέσω της οποίας μπορούν να εξαχθούν πιο ακριβή φράγματα από αυτά της παραμέτρου $\mathcal{K}(G, D)$ που θα οδηγήσουν σε μια προσέγγιση του $t_{\max}^{\text{CPA}}$ με λόγο μικρότερο του 2. Ένα άλλο θέμα που δεν έχει μελετηθεί αρκετά είναι η ανάπτυξη τοπικά ανεκτικών πρωτοκόλλων για ασύρματα δίκτυα, όπου πρέπει να αντιμετωπιστεί και το ζήτημα της παρεμβολής συχνοτήτων όταν δύο κόμβοι εκπέμπουν ταυτόχρονα σε κάποιον άλλον (σύγκρουση). Σε αυτήν την κατεύθυνση θα πρέπει να ληφθούν υπόψιν μοντέλα στα οποία ο αντίπαλος δεν μπορεί να προκαλέσει απεριόριστο αριθμό συγκρούσεων, σε διαφορετική περίπτωση θα μπορούσε να εμποδίσει την παράδοση κάποιων μηνυμάτων επ'αόριστον. Η τελευταία περίπτωση έχει μελετηθεί εν μέρει στις εργασίες [Koo, 2004, Kranakis et al., 2001, Koo et al., 2006].

8.4 Σχόλια και βιβλιογραφικές παρατηρήσεις

Το πρόβλημα της Αξιόπιστης Εκπομπής, υπό το όνομα *πρόβλημα των Βυζαντινών Στρατηγών* (Byzantine Generals) διατυπώθηκε από τους Lamport, Shostak and Pease [Lamport et al., 1982] το 1982 και μελετήθηκε αρχικά σε πλήρη δίκτυα, υπό την παρουσία γενικά φραγμένου αντιπάλου. Στο μοντέλο αυτό, αποδείχθηκε στην εργασία [Lamport et al., 1982] ότι αξιόπιστη εκπομπή μπορεί να επιτευχθεί αν και μόνον αν $t < n/3$, όπου n είναι ο συνολικός αριθμός των παικτών που συμμετέχουν. Μετά από μια σειρά συνεχών βελτιώσεων της πολυπλοκότητας ή/και της ανεκτικότητας των πρωτοκόλλων, το 1998 οι Garay και Moses [Garay and Moses, 1998] παρουσίασαν το πρώτο *πλήρως πολυωνυμικό* πρωτόκολλο το οποίο ήταν βέλτιστο ως προς την πολυπλοκότητα γύρων και μπορούσε να ανεχθεί τον μέγιστο αριθμό διαφθορών $t = \lceil n/3 \rceil - 1$.

Το πρόβλημα της Αξιόπιστης Εκπομπής σε μη-πλήρη δίκτυα έχει μελετηθεί σε πολύ μικρότερο βαθμό. Η αρχική μελέτη του Dolev [Dolev, 1982] απέδειξε ότι πρέπει να ισχύει επιπλέον η συνθήκη $t < c/2$, όπου c είναι η συνδεσιμότητα του δικτύου και αυτό το φράγμα είναι ακριβές, δηλαδή η συνθήκη είναι αναγκαία και ικανή. Στη συνέχεια, ακολούθησαν άλλες εργασίες [Dolev et al., 1993, Kumar et al., 2002] όπου ουσιαστικά αντιμετωπίζουν το πρόβλημα κυρίως μέσω πρωτοκόλλων για *αξιόπιστη και ασφαλή μετάδοση μηνύματος*. Από τα τελευταία προκύπτουν και πρωτόκολλα Αξιόπιστης Εκπομπής για μη-πλήρη δίκτυα, αφού μπορούν να χρησιμοποιηθούν για την προσομοίωση αξιόπιστων καναλιών επικοινωνίας για κάθε ζεύγος παικτών.

Ο Koo [Koo, 2004] εισήγαγε το μοντέλο του t-τοπικά φραγμένου αντιπάλου, που έχει νόημα μόνο σε σε μη-πλήρη δίκτυα. Μελέτησε το πρόβλημα της Αξιόπιστης Εκμπομπής σε δίκτυα άγνωστης τοπολογίας (*ad hoc*) και πρότεινε ένα απλό, αλλά με μεγάλες δυνατότητες πρωτόκολλο που ονομάστηκε αργότερα από τους Pelc, Peleg [Pelc and Peleg, 2005] *Certified Propagation Algorithm*(CPA). Οι Pelc και Peleg [Pelc and Peleg, 2005] μελέτησαν το μοντέλο t-τοπικά φραγμένου αντιπάλου σε γραφήματα γενικής τοπολογίας και πρότειναν μια ικανή τοπολογική συνθήκη υπό την οποία ο CPA μπορεί να επιτύχει εκπομπή σε κάθε δίκτυο. Επίσης όρισαν μια γραφοθεωρητική παράμετρο που αποτελεί άνω φράγμα για τον αριθμό των διεφθαρμένων παικτών t που μπορεί να γίνει ανεκτός τοπικά από οποιοδήποτε πρωτόκολλο Εκπομπής.

Η εξαγωγή ακριβέστερων φραγμάτων για την μέγιστη ανοχή του CPA αφέθηκε από τους Pelc, Peleg σαν ανοικτό πρόβλημα. Σε αυτήν την κατεύθυνση, προτάθηκε από τους Ichimura and Shigeno [Ichimura and Shigeno, 2010] μια αποδοτικά υπολογίσιμη γραφοθεωρητική παράμετρος η οποία δίνει τελικά ένα έναν καλύτερο, αλλά όχι ακριβή, χαρακτηρισμό της οικογένειας των γραφημάτων στην οποία ο CPA πετυχαίνει Εκπομπή. Στην εργασία [Litsas et al., 2013] δόθηκε τελικά μια γραφοθεωρητική παράμετρος που αποκαλύπτει το ακριβές πλήθος των τοπικών διαφθορών που μπορεί να ανεχθεί ο CPA. Ανεξάρτητα, μία ισοδύναμη αναγκαία και ικανή συνθήκη δόθηκε στην εργασία [Tseng et al., 2015]. Επιπλέον, στην εργασία [Pagourtzis et al., 2014] αποδείχθηκε ότι κανείς ασφαλής αλγόριθμος δεν μπορεί να πετύχει Εκπομπή με περισσότερες τοπικές διαφθορές απ'ότι μπορεί να ανεχθεί ο CPA, αποδεικνύοντας έτσι την μοναδικότητα του CPA, που είχε επίσης τεθεί σαν ανοιχτό πρόβλημα στην εργασία [Pelc and Peleg, 2005].

Βιβλιογραφία

[Dolev, 1982] Dolev, D. (1982). The byzantine generals strike again. *J. Algorithms*, 3(1):14–30.

[Dolev et al., 1993] Dolev, D., Dwork, C., Waarts, O., and Yung, M. (1993). Perfectly secure message transmission. *J. ACM*, 40(1):17–47.

[Garay and Moses, 1998] Garay, J. A. and Moses, Y. (1998). Fully polynomial byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Comput.*, 27(1):247–290.

[Hirt and Maurer, 1997] Hirt, M. and Maurer, U. M. (1997). Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In Burns, J. E. and Attiya, H., editors, *PODC*, pages 25–34. ACM.

[Ichimura and Shigeno, 2010] Ichimura, A. and Shigeno, M. (2010). A new parameter for a broadcast algorithm with locally bounded byzantine faults. *Inf. Process. Lett.*, 110(12-13):514–517.

[Koo et al., 2006] Koo, C., Bhandari, V., Katz, J., and Vaidya, N. H. (2006). Reliable broadcast in radio networks: the bounded collision case. In Ruppert, E. and Malkhi, D., editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 258–264. ACM.

[Koo, 2004] Koo, C.-Y. (2004). Broadcast in radio networks tolerating byzantine adversarial behavior. In Chaudhuri, S. and Kutten, S., editors, *PODC*, pages 275–282. ACM.

[Kranakis et al., 2001] Kranakis, E., Krizanc, D., and Pelc, A. (2001). Fault-tolerant broadcasting in radio networks. *Journal of Algorithms*, 39(1):47 – 67.

[Kumar et al., 2002] Kumar, M. V. N. A., Goundan, P. R., Srinathan, K., and Rangan, C. P. (2002). On perfectly secure communication over arbitrary networks. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, PODC '02, pages 193–202, New York, NY, USA. ACM.

[Lamport et al., 1982] Lamport, L., Shostak, R. E., and Pease, M. C. (1982). The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401.

[Litsas et al., 2013] Litsas, C., Pagourtzis, A., and Sakavalas, D. (2013). A graph parameter that matches the resilience of the certified propagation algorithm. In Cichon, J., Gebala, M., and Klonowski, M., editors, *ADHOC-NOW*, volume 7960 of *Lecture Notes in Computer Science*, pages 269–280. Springer.

[Nagamochi and Ibaraki, 2008] Nagamochi, H. and Ibaraki, T. (2008). *Algorithmic Aspects of Graph Connectivity*. Cambridge University Press. Cambridge Books Online.

[Pagourtzis et al., 2014] Pagourtzis, A., Panagiotakos, G., and Sakavalas, D. (2014). Reliable broadcast with respect to topology knowledge. In Kuhn, F., editor, *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, volume 8784 of *Lecture Notes in Computer Science*, pages 107–121. Springer.

[Pelc and Peleg, 2005] Pelc, A. and Peleg, D. (2005). Broadcasting with locally bounded byzantine faults. *Inf. Process. Lett.*, 93(3):109–115.

[Tseng et al., 2015] Tseng, L., Vaidya, N., and Bhandari, V. (2015). Broadcast using certified propagation algorithm in presence of byzantine faults. *Information Processing Letters*, 115(4):512 – 514.

ΚΕΦΑΛΑΙΟ 9

Το Πρόβλημα της Συνάντησης σε Άλλα Σενάρια

Σε αυτό το κεφάλαιο μελετούμε άλλα ενδιαφέροντα προβλήματα ξεκινώντας με τη σχέση που έχει το πρόβλημα εκλογής αρχηγού με το πρόβλημα της συνάντησης σε έναν προσανατολισμένο δακτύλιο. Στη συνέχεια παρουσιάζουμε πιθανοτικούς αλγόριθμους για το πρόβλημα της συνάντησης δύο πρακτόρων σε δακτύλιο. Έπειτα παρουσιάζουμε αλγόριθμους για το πρόβλημα της συνάντησης με πράκτορες που χρησιμοποιούν σημάδια τα οποία μπορεί να εξαφανιστούν από τους κόμβους. Τέλος, γίνονται σύντομες αναφορές για την επίλυση του προβλήματος σε δέντρα και γενικευμένα γραφήματα.

9.1 Η εκλογή αρχηγού και το πρόβλημα της συνάντησης

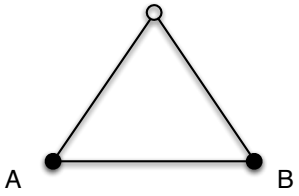
Σε αυτήν την ενότητα εξετάζουμε τη σχέση μεταξύ του προβλήματος της συνάντησης και του προβλήματος της εκλογής αρχηγού ανάμεσα από k ανώνυμους κινητούς πράκτορες σε έναν συγχρονισμένο δακτύλιο. Οι πράκτορες έχουν από ένα αμετακίνητο σημάδι και αρκετή μνήμη για να μπορούν να μετρούν αποστάσεις ανάμεσα στα σημάδια. Το πρόβλημα εκλογής αρχηγού αναφέρεται στην διαδικασία με την οποία μπορεί να εκλεγεί ένας συγκεκριμένος πράκτορας με σκοπό να διευθύνει μια ιδιαίτερη αλγοριθμική διαδικασία που αφορά το δίκτυο και την ομάδα των πρακτόρων. Όπως θα φανεί η σχέση μεταξύ των δύο αυτών προβλημάτων εξαρτάται από το εάν οι πράκτορες συμφωνούν ως προς τον προσανατολισμό του δικτύου. Μπορούμε να αποδείξουμε το επόμενο θεώρημα.

Θεώρημα 9.1 ([Flocchini et al., 2004b]) Έστω $k \geq 2$ πανομοιότυποι και ανώνυμοι πράκτορες σε έναν ανώνυμο και συγχρονισμένο δακτύλιο. Οι πράκτορες έχουν από ένα αμετακίνητο σημάδι και αρκετή μνήμη για να μπορούν να μετρούν τις αποστάσεις μεταξύ των σημαδιών. Αν οι πράκτορες συμφωνούν ως προς τον προσανατολισμό του δακτυλίου, τότε το πρόβλημα της εκλογής αρχηγού είναι ισοδύναμο με το πρόβλημα της συνάντησης. Αν όμως οι πράκτορες δεν συμφωνούν

ως προς τον προσανατολισμό του δακτυλίου, τότε το πρόβλημα της εκλογής αρχηγού είναι αυστηρά δυσκολότερο από το πρόβλημα της συνάντησης.

Απόδειξη. **(Ιδέα.)** Αν το πρόβλημα της εκλογής αρχηγού μπορεί να λυθεί τότε ο εκλεγμένος πράκτορας μπορεί να ταξιδέψει στον δακτύλιο και να δώσει οδηγίες σε κάθε άλλον πράκτορα για τη συνάντηση σε έναν συγκεκριμένο κόμβο. Έτσι η λύση του προβλήματος εκλογής δίνει πάντα μια λύση για το αντίστοιχο πρόβλημα συνάντησης.

Όμως όπως δείχνουμε παρακάτω, μια λύση για το πρόβλημα της συνάντησης δεν δίνει αναγκαστικά λύση στο πρόβλημα εκλογής αρχηγού. Θεωρήστε την περίπτωση όπου $k = 2$ πράκτορες



Σχήμα 9.1: Δύο πράκτορες με απόσταση ένα μεταξύ τους σε έναν μή-προσανατολισμένο δακτύλιο τριών κόμβων.

είναι τοποθετημένοι σε έναν δακτύλιο με $n = 3$ κόμβους (βλέπε Σχήμα 9.1). Το k είναι γνωστό αλλά οι πράκτορες δεν συμφωνούν ως προς τον προσανατολισμό του δακτυλίου.

Ο Αλγόριθμος 34 θα οδηγήσει τους πράκτορες σε συνάντηση. Αν οι πράκτορες κινούνται

Αλγόριθμος 34

1: Άφησε το σημάδι.
2: Κινήσου προς τα δεξιά μέχρι:
   a. να συναντήσεις τον άλλο πράκτορα, ή
   b. να συναντήσεις κάποιο σημάδι.
3: Αν συνάντησες κάποιο σημάδι, άλλαξε κατεύθυνση και κινήσου μέχρι να συναντήσεις τον άλλο πράκτορα.

αρχικά προς την ίδια κατεύθυνση, θα συναντηθούν μετά από δύο βήματα. Αν κινούνται αρχικά σε αντίθετες κατευθύνσεις θα συναντηθούν μετά από το πολύ δύο βήματα. Σε αυτήν την περίπτωση όμως, η εκλογή αρχηγού δεν είναι δυνατή καθώς μπορεί να δειχθεί ότι οι πράκτορες κάθε στιγμή βρίσκονται στην ίδια κατάσταση. Συνεπώς όταν οι πράκτορες δεν συμφωνούν ως προς τον προσανατολισμό του δακτυλίου, η εκλογή αρχηγού είναι πιο δύσκολη από την επίλυση του προβλήματος της συνάντησης.

Αν όμως οι πράκτορες συμφωνούν ως προς τον προσανατολισμό τότε η λύση του προβλήματος της συνάντησης δίνει πάντα μια λύση στο πρόβλημα της εκλογής αρχηγού. Στον Αλγόριθμο 22 ο τελευταίος πράκτορας που παραμένει ενεργός εκλέγεται αρχηγός και οργανώνει τη συνάντηση. ■

9.2 Πιθανοτικοί αλγόριθμοι για συνάντηση σε δακτύλιο

9.2.1 Εισαγωγή

Σε αυτήν την ενότητα μελετούμε το πρόβλημα της συνάντησης σε έναν συγχρονισμένο δακτύλιο χωρίς προσανατολισμό όταν οι κινήσεις των πρακτόρων είναι τυχαίες. Αρχικά αναλύουμε τη βασική τυχαία κίνηση δύο πρακτόρων και υπολογίζουμε τον αναμενόμενο χρόνο συνάντησης. Στη συνέχεια εξετάζουμε πώς μπορούν να χρησιμοποιηθούν σημάδια για να μειωθεί ο χρόνος συνάντησης. Τέλος αποδεικνύουμε ένα βέλτιστο ισοζύγιο μνήμης-χρόνου για πιθανοτικούς αλγόριθμους όταν οι πράκτορες δεν χρησιμοποιούν σημάδια. Μια περίληψη των αποτελεσμάτων φαίνεται στον

Πίνακας 9.1: Ισοζύγια Μνήμης-Χρόνου για Πιθανοτικούς Αλγόριθμους Συνάντησης

Αλγόριθμος	Χρόνος	Μνήμη	Τυχαία Bits
Random Walk	$O(n^2)$	$O(1)$	$O(n^2)$
Random Walk με Σημάδια	$O(n)$	$O(1)$	$O(1)$
Coin Half-Tour	$O(n)$	$O(\log n)$	$O(1)$
Approximate Counting	$O(n)$	$O(\log \log n)$	$O(n)$

Πίνακα 9.1. Ο αναμενόμενος αριθμός των τυχαίων bits που χρησιμοποιούνται από έναν αλγόριθμο είναι άλλο ένα από τα αγαθά που κάποιος ίσως ενδιαφέρεται να ελαχιστοποιήσει και γι' αυτό περιλαμβάνεται στον πίνακα. Οι τιμές αυτές προκύπτουν εύκολα από την περιγραφή των αλγόριθμων.

Σε αυτήν την ενότητα μοντελοποιούμε κάθε πράκτορα ως πιθανοτικό αυτόματο πεπερασμένων καταστάσεων $A =< S, \delta, s_0 >$ όπου:

- S είναι το σύνολο των καταστάσεων του αυτόματου συμπεριλαμβάνοντας την αρχική κατάσταση s_0 και μια ειδική κατάσταση `halt`, και
- $\delta : S \times C \times P \to S \times M$ όπου, $C = \{H, T\}$ αναπαριστά μια τυχαία ρίψη νομίσματος, $P = \{\texttt{present}, \texttt{notpresent}\}$ αναπαριστά ένα κατηγόρημα που δείχνει αν υπάρχει άλλος πράκτορας στον κόμβο, και $M = \{\texttt{left}, \texttt{right}\}$ αναπαριστά τις κινήσεις που μπορεί να κάνει ο πράκτορας.

Κατά τη διάρκεια ενός συγχρονισμένου βήματος, ανάλογα με την τρέχουσα κατάσταση, την παρουσία άλλου πράκτορα στον κόμβο, και την τυχαία τιμή μιας ανεξάρτητης μεταβλητής η οποία μπορεί να είναι `heads` με πιθανότητα ίση με 0.5, ο πράκτορας χρησιμοποιεί τη συνάρτηση δ για να μεταβεί σε κάποια άλλη κατάσταση και να κινηθεί `left` ή `right`.

Στο παραπάνω μοντέλο είναι εύκολο να δει κανείς ότι αν οι δύο πράκτορες έχουν μεταξύ τους απόσταση περιττού μήκους σε έναν δακτύλιο άρτιου πλήθους κόμβων δεν μπορούν να συναντηθούν καθώς κινούνται σε κάθε βήμα και συνεπώς θα παραμένουν σε απόσταση περιττού μήκους για πάντα. Ένας τρόπος να αντιμετωπιστεί αυτό είναι να προσθέσουμε μια τρίτη επιλογή `stay` στο

σύνολο M. Για απλότητα στην ανάλυση θα θεωρήσουμε ότι οι πράκτορες έχουν μεταξύ τους απόσταση άρτιου μήκους σε ένα δακτύλιο άρτιου πλήθους κόμβων.

9.2.2 Ο Αλγόριθμος Random Walk

Έχει παρατηρηθεί από πολλούς ότι αν δύο πράκτορες κινούνται τυχαία σε ένα δίκτυο τελικά θα συναντηθούν με πιθανότητα ένα. Αυτή είναι η βάση του αλγόριθμου που ακολουθεί ο οποίος οδηγεί δύο πράκτορες σε έναν ανώνυμο, συγχρονισμένο δακτύλιο χωρίς προσανατολισμό. Σημειώνουμε ότι οι πράκτορες δεν χρησιμοποιούν σημάδια.

Αλγόριθμος 35 (Random Walk)

1: **Επανέλαβε μέχρι** να συναντήσεις τον άλλο πράκτορα:
2: **Άν** `heads` κινήσου `right`
Αλλιώς κινήσου `left`

Οι ιδιότητες του αλγορίθμου εξετάζονται στο παρακάτω θεώρημα.

Θεώρημα 9.2 ([Kranakis and Krizanc, 2007]) Δύο πράκτορες με σταθερή μνήμη οι οποίοι αρχικά έχουν μεταξύ τους απόσταση άρτιου μήκους $d \leq n/2$ σε έναν συγχρονισμένο δακτύλιο n κόμβων χωρίς προσανατολισμό μπορούν να συναντηθούν σε αναμενόμενο χρόνο $\frac{d}{2}(n-d)$ εκτελώντας τον Αλγόριθμο 35.

Απόδειξη. Έστω E_d ο αναμενόμενος χρόνος συνάντησης δύο πρακτόρων που ξεκινούν σε απόσταση άρτιου μήκους d ο ένας από τον άλλον σε έναν δακτύλιο άρτιου πλήθους κόμβων n και εκτελούν τον παραπάνω αλγόριθμο. Τότε είναι εύκολο να δούμε ότι ισχύει $E_0 = 0$, and $E_{n/2} = 1 + (1/2)E_{n/2} + (1/2)E_{n/2-2}$. Από την τελευταία εξίσωση προκύπτει:

$$E_{n/2} = 2 + E_{n/2-2}. \tag{9.1}$$

Γενικότερα κατά τη διάρκεια της εκτέλεσης του αλγόριθμου μια από τις ακόλουθες τρεις περιπτώσεις μπορεί να εμφανιστεί. Οι πράκτορες σε ένα βήμα είτε κινούνται προς την ίδια κατεύθυνση με πιθανότητα $1/2$, είτε σε αντίθετη κατεύθυνση μειώνοντας την απόσταση d με πιθανότητα $1/4$ είτε σε αντίθετη κατεύθυνση αυξάνοντας την απόσταση d με πιθανότητα $1/4$. Έτσι παίρνουμε την εξίσωση:

$$E_d = 1 + (1/2)E_d + (1/4)E_{d-2} + (1/4)E_{d+2}, \tag{9.2}$$

για $d = 2, 4, \ldots, n/2 - 2$. (Παρατηρήστε ότι η περίπτωση $d = n/2$ είναι ιδιαίτερη με την έννοια ότι σε αυτήν την περίπτωση βρίσκονται πάντα σε απόσταση $n/2$.) Αντικαθιστώντας με $d+2$ το d στην εξίσωση 9.2 και λύνοντας την εξίσωση που προκύπτει ως προς E_d έχουμε ότι για $d \geq 4$ ισχύει:

$$E_d = 2E_{d-2} - E_{d-4} - 4. \tag{9.3}$$

Η αρχική συνθήκη $E_0 = 0$ και η εξίσωση 9.3 μας δίνουν $E_4 = 2E_2 - 4$. Γενικότερα, μπορούμε να αποδείξουμε την ακόλουθη εξίσωση για $2d \leq n/2$,

$$E_{2d} = dE_2 - 2d(d-1). \tag{9.4}$$

Αποδεικνύουμε επαγωγικά ότι υπάρχουν ακολουθίες a_d, b_d έτσι ώστε

$$E_{2d} = a_d E_2 - 4b_d.$$

Πραγματικά, ισχύει

$$\begin{aligned} E_{2d} &= 2E_{2d-2} - E_{2d-4} - 4 \\ &= 2\left(a_{d-1}E_2 - 4b_{d-1}\right) - \left(a_{d-2}E_2 - 4b_{d-2}\right) - 4 \\ &= (2a_{d-1} - a_{d-2})E_2 - 4(2b_{d-1} - b_{d-2} + 1), \end{aligned}$$

από το οποίο προκύπτουν οι σχέσεις $a_d = 2a_{d-1} - a_{d-2}$ και $b_d = 2b_{d-1} - b_{d-2} + 1$ με αρχικές συνθήκες $a_0 = b_0 = 0, a_1 = 1, b_1 = 0$. Λύνοντας τις εξισώσεις παίρνουμε ότι $a_d = d$ και $b_d = -\frac{1}{2}d + \frac{1}{2}d^2$, το οποίο αποδεικνύει την εξίσωση 9.4. Για να βρούμε έναν τύπο για το E_{2d}, απομένει να υπολογίσουμε το E_2. Η εξίσωση 9.4 μας δίνει τις τιμές

$$\begin{aligned} E_{n/2} &= \frac{n}{4}E_2 - 2\frac{n}{4}\left(\frac{n}{4} - 1\right) \\ E_{n/2-2} &= \left(\frac{n}{4} - 1\right)E_2 - 2\left(\frac{n}{4} - 1\right)\left(\frac{n}{4} - 2\right), \end{aligned}$$

οι οποίες όταν αντικατασταθούν στην εξίσωση 9.1 προκύπτει ότι $E_2 = n - 2$. Τέλος αντικαθιστώντας την τελευταία τιμή στην εξίσωση 9.4 παίρνουμε

$$E_{2d} = d(n - 2d). \tag{9.5}$$

Προφανώς ο παραπάνω αλγόριθμος υλοποιείται από ένα πεπερασμένο αυτόματο και αυτό ολοκληρώνει την απόδειξη. ■

Ενώ οι πράκτορες σε αυτόν τον αλγόριθμο έχουν βέλτιστη μνήμη, αφού στη χειρότερη περίπτωση η απόσταση $d = \Theta(n)$ σημαίνει πως ο αναμενόμενος χρόνος είναι τετραγωνικός. Έτσι λοιπόν μια φυσική ερώτηση είναι αν είναι δυνατόν να πετύχουμε τη συνάντηση σε γραμμικό χρόνο.

9.2.3 Πιθανοτικοί αλγόριθμοι με σημάδια

Μελετούμε τώρα πώς μπορεί να λυθεί το πρόβλημα της συνάντησης από πιθανοτικούς αλγόριθμους που χρησιμοποιούν σημάδια. Θεωρήστε ότι οι πράκτορες έχουν σταθερή μνήμη και δεν γνωρίζουν ούτε το πλήθος n των κόμβων του δακτυλίου ούτε την μεταξύ τους αρχική απόσταση d. Σε αυτήν την περίπτωση μπορούμε να αποδείξουμε το παρακάτω θεώρημα.

Αλγόριθμος 36 (Random Walk με Σημάδια)

1: Άφησε το σημάδι.
2: Θέσε τον μετρητή count = 0.
3: Διάλεξε μια κατεύθυνση και κινήσου μέχρι να συναντήσεις ένα σημάδι.
4: Θέσε τον μετρητή count = count + 1.
5: **Αν** count mod $2 = 0$, άλλαξε κατεύθυνση με πιθανότητα $0 \leq p \leq 1$.
6: **Αλλιώς** διατήρησε την ίδια κατεύθυνση.
7: Κινήσου μέχρι να συναντήσεις το επόμενο σημάδι.
8: **Επανέλαβε** από το βήμα 4 μέχρι να συναντηθείς με τον άλλον πράκτορα.

Θεώρημα 9.3 ([Sawchuk, 2004]) Δύο πράκτορες με σταθερή μνήμη και ένα μή-μετακινήσιμο σημάδι ο καθένας οι οποίοι αρχικά έχουν μια άρτια απόσταση $d \leq n/2$ σε ένα συγχρονισμένο δακτύλιο n κόμβων χωρίς προσανατολισμό συναντιώνται σε αναμενόμενο χρόνο $O(n)$ αν εκτελέσουν τον Αλγόριθμο 36.

Απόδειξη. Ο αναμενόμενος αριθμός των γύρων στον Αλγόριθμο 36 είναι

$$\sum_{i=1}^{\infty}(2p(1-p))(1-2p(1-p))^{i-1}i = \frac{1}{(2p(1-p))}$$

Όλοι οι γύροι εκτός από τον τελευταίο διαρκούν χρόνο n. Από τη στιγμή που οι πράκτορες θα κινηθούν σε αντίθετες κατευθύνσεις ο επιπλέον αναμενόμενος χρόνος για τη συνάντηση είναι $\frac{n}{2}$. Συνεπώς ο συνολικός αναμενόμενος χρόνος συνάντησης είναι

$$n\left(\frac{1}{2p(1-p)} - 1\right) + \frac{n}{2} = \frac{n(1-p(1-p))}{2p(1-p)}$$

ο οποίος είναι της τάξης $O(n)$. Ο αναμενόμενος χρόνος είναι ελάχιστος όταν $p = \frac{1}{2}$. ■

9.2.4 Ισοζύγια χρόνου-μνήμης

Είδαμε στην προηγούμενη ενότητα ότι συνδυάζοντας την τυχαία κίνηση με τη χρήση μή-μετακινήσιμων σημαδιών μπορούμε να πετύχουμε γραμμικό (ως προς το πλήθος των κόμβων του δακτυλίου) αναμενόμενο χρόνο για τη συνάντηση ενώ η μνήμη των πρακτόρων παραμένει σταθερή. Σε αυτήν την ενότητα ενδιαφερόμαστε να απαντήσουμε στην ακόλουθη ερώτηση: Ποιά είναι η ελάχιστη μνήμη που χρειάζονται οι πράκτορες για να συναντηθούν μέσα σε $O(n)$ αναμενόμενο χρόνο, σε έναν συγχρονισμένο δακτύλιο από n κόμβους χωρίς προσανατολισμό, όταν οι πράκτορες δεν χρησιμοποιούν σημάδια (δηλαδή όπως ακριβώς στην ενότητα 9.2.2);

9.2.4.1 Ο Αλγόριθμος Coin Half Tour

Μπορούμε να πετύχουμε γραμμικό αναμενόμενο χρόνο συνάντησης χρησιμοποιώντας τον Αλγόριθμο 37, ο οποίος αναφέρεται σαν 'Coin Half Tour' στο άρθρο [Alpern, 1995].

Αλγόριθμος 37 (Coin Half Tour)

1: **Επανέλαβε** μέχρι να συναντήσεις τον άλλον πράκτορα:
2: **Άν** `heads` κινήσου `right` για $n/2$ βήματα
Αλλιώς κινήσου `left` για $n/2$ βήματα

Θεώρημα 9.4 ([Alpern, 1995]) Δύο πράκτορες με μνήμη $O(\log n)$ ο καθένας, οι οποίοι βρίσκονται αρχικά σε μια άρτια απόσταση $d \leq n/2$ σε έναν συγχρονισμένο δακτύλιο με άρτιο πλήθος κόμβων n και χωρίς προσανατολισμό συναντιούνται μετά από αναμενόμενο χρόνο $O(n)$ εκτελώντας τον Αλγόριθμο 37.

Απόδειξη. Αν θεωρήσουμε κάθε εκτέλεση του βήματος 2 σαν μια φάση ονομάζουμε μια φάση 'επιτυχημένη' αν οι δύο πράκτορες επιλέξουν να κινηθούν σε αντίθετες κατευθύνσεις και αποτυχημένη σε αντίθετη περίπτωση. Τώρα είναι εύκολο να δούμε ότι (α) ο αναμενόμενος αριθμός από αποτυχημένες φάσεις προτού επιτευχθεί μια επιτυχημένη φάση είναι ένα (β) ο αριθμός των βημάτων σε μια αποτυχημένη φάση είναι $n/2$ και (γ) το πλήθος των βημάτων σε μια επιτυχημένη φάση είναι λιγότερο από n. Άρα ο αναμενόμενος αριθμός βημάτων μέχρι οι πράκτορες να συναντηθούν είναι $O(n)$ αφού θα συναντηθούν σίγουρα κατά τη διάρκεια μιας επιτυχημένης φάσης. Παρατηρήστε ότι αυτό είναι αναξάρτητο από τις αρχικές θέσεις των πρακτόρων, θεωρώντας ότι $d > 0$. Επιπλέον ένα αυτόματο που υλοποιεί τον παραπάνω αλγόριθμο χρειάζεται $n/2 + O(1)$ καταστάσεις, δηλαδή $O(\log n)$ μνήμη, το οποίο ολοκληρώνει την απόδειξη. ■

Ο παραπάνω αλγόριθμος είναι βέλτιστος ως προς τον αναμενόμενο χρόνο συνάντησης άλλα απαιτεί μνήμη $O(\log n)$ bits. Είναι δυνατόν να πετύχουμε γραμμικό αναμενόμενο χρόνο εκτέλεσης χρησιμοποιώντας λιγότερη μνήμη;

9.2.4.2 Ο Αλγόριθμος Approximate Counting

Αντικαθιστώντας τον αριθμό των $n/2$ βημάτων που απαιτεί το βήμα 2 του Αλγόριθμου Coin Half Tour με έναν προσεγγιστικά αναμενόμενο αριθμό από $O(n)$ βήματα μπορούμε να μειώσουμε τις απαιτήσεις σε μνήμη για την επίτευξη της συνάντησης. Θεωρήστε τον Αλγόριθμο 38 για δύο πράκτορες με k bits μνήμη ο καθένας.

Αλγόριθμος 38 (Approximate Counting)

1: **Επανέλαβε** μέχρι να συναντήσεις τον άλλον πράκτορα:
2: (a) **Αν** `heads` θέσε $dir =$ `right`
Αλλιώς θέσε $dir =$ `left`
(b) **Επανέλαβε** μέχρι να πετύχεις 2^k `heads` στη σειρά: Κινήσου σε κατεύθυνση dir

Ορίζοντας όπως πρέπει την φάση και κάνοντας μια λεπτομερή ανάλυση μπορούμε να δείξουμε ότι οι φάσεις έχουν αναμενόμενη διάρκεια $O(2^{2^k})$ και σταθερή πιθανότητα επιτυχίας και έτσι αποδεικνύουμε ότι:

Θεώρημα 9.5 ([Kranakis et al., 2008]) Δύο πράκτορες με k bits μνήμη ο καθένας που βρίσκονται αρχικά σε μια άρτια απόσταση $d \leq n/2$ σε έναν συγχρονισμένο δακτύλιο άρτιου πλήθους κόμβων n χωρίς προσανατολισμό μπορούν να συναντηθούν σε αναμενόμενο χρόνο

$$O\left(\frac{n^2}{2^{2^k}} + 2^{2^k}\right)$$

εκτελώντας τον Αλγόριθμο 38. ■

Ειδικότερα, το παραπάνω θεώρημα δείχνει ότι $\log\log n$ bits μνήμη αρκούν για την επίτευξη συνάντησης σε γραμμικό χρόνο. Αποδεικνύεται ότι αυτό το αποτέλεσμα είναι βέλτιστο:

Θεώρημα 9.6 ([Kranakis et al., 2008]) Οποιοσδήποτε αλγόριθμος που οδηγεί σε συνάντηση δύο πράκτορες σε αναμενόμενο χρόνο $\Theta(n)$ σε έναν συγχρονισμένο δακτύλιο από n κόμβους χωρίς προσανατολισμό χρειάζεται $\Omega(\log\log n)$ bits μνήμη. ■

Οι αποδείξεις για τα δύο αυτά θεωρήματα βασίζονται σε μια λεπτομερή ανάλυση του Αλγόριθμου Approximate Counting χρησιμοποιώντας τη θεωρία των martingales, stopping times, και την Wald's Identity. Λεπτομέρειες μπορούν να βρεθούν στο άρθρο [Kranakis et al., 2008].

9.3 Συνάντηση με σημάδια που σβήνουν

Σε αυτήν την ενότητα μελετούμε και αναλύουμε το πρόβλημα της συνάντησης μεταξύ πρακτόρων που χρησιμοποιούν σημάδια τα οποία μπορεί να σβήσουν και να μην είναι πλέον ορατά από τους πράκτορες. Για παράδειγμα, ένα σημάδι μπορεί να αποτύχει αν τοποθετηθεί σε έναν εχθρικό κόμβο ή αν ένας εχθρικός πράκτορας που περνά από τον κόμβο το σβήσει.

Έστω n το μέγεθος ενός συγχρονισμένου μή-προσανατολισμένου δακτυλίου όπου βρίσκονται διασκορπισμένοι k πράκτορες. Οι πράκτορες χρησιμοποιούν από ένα μή-μετακινήσιμο σημάδι, ενώ το πολύ $k-1$ σημάδια μπορούν να σβήσουν. Επίσης θεωρούμε ότι ισχύει $\gcd(k', n) = 1$ για κάθε $k' \leq k$. Θεωρούμε ως ακολουθία των αποστάσεων την ακολουθία των αποστάσεων μεταξύ των σημαδιών που δεν σβήνουν. Αφού τα σημάδια είναι μή-μετακινήσιμα, οι αρχικές τους αποστάσεις διατηρούνται, εκτός αν κάποιο σημάδι σβήσει. Όταν ένα σημάδι σβήσει, δεν είναι πια ορατό σε κανέναν πράκτορα που επισκέπτεται τον κόμβο. Θα μελετήσουμε δύο ειδών αποτυχίες των σημαδιών: όταν τα σημάδια σβήνουν αμέσως αφού τοποθετηθούν σε κάποιο κόμβο, και όταν τα σημάδια μπορούν να σβήσουν οποιαδήποτε στιγμή. Έστω f ο αριθμός των σημαδιών που μπορεί να αποτύχουν. Θα δούμε ότι όπως και σε προηγούμενο κεφάλαιο, η πολυπλοκότητα του προβλήματος εξαρτάται από την γνώση των πρακτόρων για τις τιμές των k και n.

Σχετικά με την επιλυσιμότητα του προβλήματος της συνάντησης, παρατηρείστε ότι αν όλα τα σημάδια σβήσουν ή αν το πλήθος των σημαδιών k είναι τέτοιο ώστε $\gcd(k', n) \neq 1$ για κάποιο $k' \leq k$, τότε η συνάντηση είναι αδύνατη όπως αναφέρεται στο άρθρο [Flocchini et al., 2004b].

9.3.1 Όταν τα σημάδια σβήνουν αμέσως μετά την τοποθέτησή τους

Έστω ότι τα σημάδια σβήνουν αμέσως μετά την τοποθέτησή τους. Ο πράκτορας που τοποθέτησε ένα τέτοιο σημάδι δεν γνωρίζει ότι έσβησε. Αν ούτε το μέγεθος n του δακτυλίου, ούτε το πλήθος k των πρακτόρων είναι γνωστά στους πράκτορες, τότε το πρόβλημα δεν λύνεται (βλέπε άρθρο [Flocchini et al., 2004b]). Συνεπώς θεωρούμε τις εξής δύο περιπτώσεις: όταν μόνο το n είναι γνωστό και όταν μόνο το k είναι γνωστό.

n γνωστό, k άγνωστο

Ο αλγόριθμος είναι αρκετά απλός. Κάθε πράκτορας αφήνει το σημάδι του και κινείται στον δακτύλιο υπολογίζοντας τις αποστάσεις μεταξύ των διαδοχικών σημαδιών που συναντά μέχρι να επιστρέψει στον κόμβο εκκίνησης. Έστω $s_1, \ldots, s_h$ η ακολουθία που υπολογίστηκε, όπου s_1 και s_h οι αποστάσεις μεταξύ του κόμβου εκκίνησης και του πρώτου και τελευταίου σημαδιού αντίστοιχα. Αν το σημάδι ενός πράκτορα δεν αποτυγχάνει, $h = k - f$ και αυτή η ακολουθία είναι ακριβώς η (κυκλική) ακολουθία $\mathcal{S} = d_1, \ldots, d_h$ των αποστάσεων των σημαδιών που δεν αποτυγχάνουν. Αν ο πράκτορας βρει ότι το σημάδι του έχει σβηστεί, τότε $h = k - f + 1$ και ο πράκτορας πρέπει να μετασχηματίσει την ακολουθία για να πάρει την πραγματική (κυκλική) ακολουθία των αποστάσεων: $\mathcal{S} = d_1, \ldots, d_{h-1} = s_1 + s_h, s_2, \ldots, s_{h-1}$.

Αφού τα σημάδια σβήνουν πριν οι πράκτορες κινηθούν, η κυκλική ακολουθία που υπολογίζεται είναι η ίδια για όλους τους πράκτορες, με τη διαφορά της κυκλικής μετάθεσης (αφού ο δακτύλιος δεν είναι προσανατολισμένος).

Στη συνέχεια οι πράκτορες θα συμφωνήσουν σε ένα σημείο συνάντησης μετά από επεξεργασία της ακολουθίας (η οποία δεν μπορεί να είναι περιοδική αφού $\gcd(k', n) \neq 1$ για κάθε $k' \leq k$). Πιο συγκεκριμένα έστω $\mathcal{S}^{\mathcal{R}}$ η αντίστροφη της ακολουθίας $\mathcal{S}$. Κάθε πράκτορας υπολογίζει τις λεξικογραφικά μέγιστες συμβολοακολουθίες στις $\mathcal{S}$ και $\mathcal{S}^{\mathcal{R}}$. Αν και οι δύο αυτές συμβολοακολουθίες ξεκινούν στον ίδιο κόμβο, τότε αυτός ο κόμβος γίνεται το σημείο συνάντησης. Αλλιώς το σημείο συνάντησης θα είναι ο μεσαίος κόμβος στο μονοπάτι περιττού μήκους μεταξύ των δύο πρώτων κόμβων των συμβολοακολουθιών (αφού το n είναι περιττό, η ύπαρξη ενός τέτοιου μονοπατιού είναι εξασφαλισμένη).

Η διαδικασία που περιγράφει πώς επιλέγεται το σημείο συνάντησης, και θα χρησιμοποιηθεί από τους αλγόριθμούς μας, φαίνεται στον Αλγόριθμο 39, όπου η $lexi(someSequence)$ επιστρέφει τη λεξικογραφικά μέγιστη κυκλική μετάθεση της ακολουθίας $someSequence$. Ολόκληρος ο αλγόριθμος περιγράφεται στον Αλγόριθμο 40. (Σημειώστε ότι $m = k - f$ είναι ο αριθμός των σημαδιών που απομένουν.)

Η απόδειξη του θεωρήματος που ακολουθεί, προκύπτει εύκολα.

Θεώρημα 9.7 ([Flocchini et al., 2004a]) Αν οι πράκτορες έχουν $O(k \log n)$ μνήμη, το n είναι γνωστό, ισχύει $\gcd(k', n) = 1$ για κάθε $k' \leq k$, $f \leq (k-1)$ σημάδια μπορεί να σβήσουν, και τα σημάδια σβήνουν αμέσως μετά την τοποθέτησή τους, τότε το πρόβλημα της συνάντησης μπορεί να λυθεί σε λιγότερα από $2n$ βήματα. ■

Αλγόριθμος 39 MeetingPoint(string)

1: Υπολόγισε τις ακολουθίες $forward = lexi(string)$ και $reverse = lexi(string^R)$.
2: Έστω x και y οι πρώτοι κόμβοι των ακολουθιών *forward* και *reverse* αντίστοιχα.
3: Αν $x = y$, τότε το σημείο συνάντησης είναι το x.
4: Αν $x \neq y$, τότε το σημείο συνάντησης είναι ο μεσαίος κόμβος του περιττού μονοπατιού μεταξύ x και y.

Αλγόριθμος 40 (MEET1)

1: Άφησε το σημάδι στον κόμβο εκκίνησης.
2: Κινήσου δεξιόστροφα.
3: Υπολόγισε την ακολουθία των αποστάσεων $s_1, \ldots, s_h$ μεταξύ των σημαδιών.
Αν υπάρχει σημάδι στον κόμβο που βρίσκεσαι, τότε $\mathcal{S} = d_1, \ldots, d_m = s_1, \ldots, s_h$.
Αν δεν υπάρχει σημάδι στον κόμβο που βρίσκεσαι, τότε $\mathcal{S} = d_1, \ldots, d_m = s_1 + s_h, s_2, \ldots, s_{h-1}$.
4: Κινήσου στο MeetingPoint(S).

Παρατηρήστε ότι η λειτουργία του αλγορίθμου δεν απαιτεί συγχρονισμένο δίκτυο. Στην πραγματικότητα κάθε πράκτορας μπορεί να κάνει τους υπολογισμούς του και να κινηθεί στο σημείο συνάντησης, ακόμη και αν ο δακτύλιος είναι ασύγχρονος.

k γνωστό, n άγνωστο

Ιδιαίτερο ενδιαφέρον παρουσιάζει η περίπτωση που το n είναι άγνωστο. Όπως και στην προηγούμενη περίπτωση, οι πράκτορες κινούνται στο δακτύλιο και υπολογίζουν την ακολουθία των αποστάσεων η οποία πρέπει να περιέχει κάποια ασυμμετρία έτσι ώστε να δίνει τη δυνατότητα στους πράκτορες να συμφωνήσουν σε κάποιο σημείο συνάντησης.

Όμως, αντίθετα με την προηγούμενη περίπτωση, αφού το n δεν είναι γνωστό, οι κυκλικές ακολουθίες που υπολογίζονται από τους πράκτορες με μόνη τη γνώση του k, δεν θα είναι οπωσδήποτε (με τη διαφορά κάποιων κυκλικών μεταθέσεων) ίδιες. Στην πραγματικότητα, κάθε πράκτορας, για να κατασκευάσει την ακολουθία, θα διασχίσει τον δακτύλιο τον ίδιο (άγνωστο) αριθμό φορών, αλλά ανάλογα με τον κόμβο εκκίνησης και το πλήθος των σημαδιών που έχουν σβήσει, θα διασχίσει ένα μέρος του δακτυλίου γενικώς διαφορετικό από εκείνο που διασχίζουν άλλοι πράκτορες.

Αν και οι ακολουθίες δεν είναι ίδιες, μπορεί να αποδειχθεί ότι οι πράκτορες μπορούν και σε αυτήν την περίπτωση να συμφωνήσουν σε ένα σημείο συνάντησης, υπολογίζοντας αποστάσεις μήκους $3k$ μεταξύ των σημαδιών. Η διαδικασία περιγράφεται στον Αλγόριθμο 41.

Έστω $\mathcal{S}^{\mathcal{R}}$ η αντίστροφη της ακολουθίας $\mathcal{S}$. Αφού τα σημάδια σβήνουν αμέσως μετά την τοποθέτησή τους, η ακολουθία $\mathcal{S}^{\mathcal{R}}$ μπορεί να χωριστεί ως εξής:

$$\mathcal{S}^{\mathcal{R}} = Q^q + d_\gamma, \ldots, d_1, \tag{9.6}$$

όπου Q^q είναι η συνένωση των q αντιγράφων μιας μοναδικής μή-περιοδικής υποακολουθίας Q, $+$ είναι ο τελεστής συνένωσης, και $d_\gamma, \ldots, d_1$ είναι μια υποακολουθία τέτοια ώστε $\gamma < |Q|$. Μετά τον υπολογισμό της υποακολουθίας Q, οι πράκτορες μπορούν να συμφωνήσουν σε έναν κόμβο συνάντησης.

Αλγόριθμος 41 (MEET2)

1: Άφησε το σημάδι στον κόμβο εκκίνησης.
2: Κινήσου δεξιόστροφα.
3: Υπολόγισε την ακολουθία αποστάσεων μήκους $3k$ μεταξύ των σημαδιών, $S = d_1, d_2, \ldots, d_{3k}$.
4: Έστω S^R η αντίστροφη ακολουθία της S.
5: Βρες τη μικρότερη μή-περιοδική υποακολουθία Q που ξεκινά με τον πρώτο κόμβο της S^R και επαναλαμβάνεται έτσι ώστε $S^R = Q^q + d_\gamma, \ldots, d_1$, όπου $\gamma < |Q|$.
6: Κινήσου στο MeetingPoint(Q)

Θεώρημα 9.8 ([Flocchini et al., 2004a]) Αν οι πράκτορες έχουν $O(k \log n)$ μνήμη, το k είναι γνωστό, ισχύει $\gcd(k', n) = 1$ για κάθε $k' \leq k$, $f \leq (k-1)$ σημάδια μπορούν να σβήσουν, και τα σημάδια σβήνουν αμέσως μετά την τοποθέτησή τους, τότε το πρόβλημα της συνάντησης μπορεί να λυθεί σε χρόνο $O(kn)$.

Απόδειξη. Έστω $m = k - f$ το πλήθος των σημαδιών που δεν αποτυγχάνουν. Έστω $A = \delta_1, \ldots, \delta_m$ η ακολουθία των m αποστάσεων μεταξύ των σημαδιών που υπάρχουν μετά την αποτυχία των f σημαδιών. Ισχύει $\sum_{i=1}^{m} \delta_i = n$. Έστω $S(a)$ η ακολουθία των $3k$ αποστάσεων μεταξύ των σημαδιών που υπολογίζονται από έναν πράκτορα a στο βήμα 3 του Αλγόριθμου 41. Έστω $S^R(a)$ η αντίστροφη ακολουθία της $S(a)$ και A^R η αντίστροφη ακολουθία της A. Για οποιοδήποτε πράκτορα, οι $3k$ αποστάσεις μεταξύ των σημαδιών έχουν τη μορφή

$$S^R(a) = (A^R)^\rho + d_\gamma, \ldots, d_1 = (\delta_m, \ldots, \delta_1)^\rho + d_\gamma, \ldots, d_1. \tag{9.7}$$

όπου $(A^R)^\rho$ είναι η παράθεση από ρ αντίγραφα της μή-περιοδικής υποακολουθίας A^R, $+$ είναι ο τελεστής παράθεσης, και $d_\gamma, \ldots, d_1$ είναι μια υποακολουθία τέτοια ώστε $\gamma < m$. Συνεπώς, υπάρχει τουλάχιστον μία μή-περιοδική υποακολουθία, η A^R, που ικανοποιεί την εξίσωση 9.6. Παρατηρήστε ότι η A^R δεν είναι περιοδική αφού περιλαμβάνει k ή λιγότερες αποστάσεις και από την υπόθεση ισχύει ότι $\gcd(k', n) = 1$ για κάθε $k' \leq k$.

Αν η A^R είναι η μικρότερη σε μήκος υποακολουθία που ικανοποιεί την εξίσωση 9.6, οι πράκτορες ανακαλύπτουν την A^R στο βήμα 5 του Αλγόριθμου 41. Αλλιώς, οι πράκτορες ανακαλύπτουν μια μικρότερη σε μήκος μή-περιοδική υποακολουθία Q, που ικανοποιεί την εξίσωση 9.6.

Η υποακολουθία που έχει ανακαλυφθεί στο βήμα 5 του Αλγόριθμου 41 είναι μοναδική. Ας δούμε γιατί: Αν η μικρότερη υποακολουθία έχει z στοιχεία, αυτά είναι τα πρώτα z στοιχεία της ακολουθίας S^R. Οποιαδήποτε άλλη υποακολουθία του ίδιου μήκους που ικανοποιεί την εξίσωση 9.6

αποτελείται επίσης από τα πρώτα z στοιχεία της S^R και συνεπώς η υποακολουθία που έχει ανακαλυφθεί στο βήμα 5 είναι μοναδική. Αυτό σημαίνει πως όλοι οι πράκτορες επιλέγουν τον ίδιο κόμβο συνάντησης στα υπόλοιπα βήματα του Αλγόριθμου 41 και συναντιούνται.

Ο υπολογισμός της ακολουθίας S, των $3k$ αποστάσεων μεταξύ των σημαδιών απαιτεί $O(k \log n)$ μνήμη και $O(kn)$ χρόνο. Η αναγνώριση της κατάλληλης υποακολουθίας στο βήμα 5, του κόμβου συνάντησης, και η μετακίνηση σε αυτόν τον κόμβο χρειάζονται $O(kn)$ χρόνο. Έτσι ο συνολικός χρόνος που απαιτείται είναι $O(kn)$. ■

Όταν τα σημάδια σβήνουν αμέσως μετά την τοποθέτησή τους, ο Αλγόριθμος 41 λύνει το πρόβλημα ακόμη και στην περίπτωση που ο δακτύλιος είναι ασύγχρονος. Μετά τον υπολογισμό της S^R, ο πράκτορας μπορεί να αναγνωρίσει τη μικρότερη μή-περιοδική ακολουθία που ικανοποιεί την εξίσωση 9.6. Αφού αυτή η ακολουθία είναι μοναδική, ο πράκτορας μπορεί να υπολογίσει τον μοναδικό κόμβο συνάντησης, να μετακινηθεί εκεί και να περιμένει τους υπόλοιπους k πράκτορες. Ο αλγόριθμος βασίζεται στην ικανότητα των πρακτόρων να μετρούν και να αναγνωρίσουν τον κόμβο συνάντησης.

9.3.2 Όταν τα σημάδια μπορεί να σβήσουν οποιαδήποτε στιγμή

Μελετούμε τώρα την περίπτωση στην οποία τα σημάδια μπορούν να σβήσουν οποιαδήποτε στιγμή. Σε αυτήν την περίπτωση, για να δουλέψουν οι αλγόριθμοι που παρουσιάζουμε, ο δακτύλιος πρέπει να είναι συγχρονισμένος.

Η ιδέα παραμένει η χρησιμοποίηση των αποστάσεων μεταξύ των σημαδιών για την αναγνώριση ενός κόμβου συνάντησης. Σε αυτήν την περίπτωση όμως, το πρόβλημα είναι πιο πολύπλοκο εξαιτίας του ότι αυτές οι αποστάσεις μπορεί να διαφέρουν απρόβλεπτα κατά τη διάρκεια του αλγόριθμου. Για να παρακάμψουμε αυτήν τη δυσκολία, ο αλγόριθμός μας λειτουργεί σε γύρους: σε κάθε γύρο οι πράκτορες υπολογίζουν κάποιες αποστάσεις μεταξύ των σημαδιών και προσπαθούν να συναντηθούν σε ένα κόμβο. Μπορεί όμως η συνάντηση να μην πραγματοποιηθεί κατά τη διάρκεια ενός γύρου αφού οι πράκτορες μπορεί να έχουν υπολογίσει διαφορετικές αποστάσεις μεταξύ των σημαδιών. Σε μια τέτοια περίπτωση, μόνο ομάδες πρακτόρων (εκείνες που έχουν υπολογίσει τις ίδιες αποστάσεις) μπορεί να συναντηθούν στον ίδιο κόμβο.

Για να λειτουργήσουν οι αλγόριθμοι, οι γύροι πρέπει να είναι συγχρονισμένοι έτσι ώστε οι πράκτορες να ξεκινούν κάθε γύρο είτε ταυτόχρονα είτε με μια περιορισμένη διαφορά χρόνου. Με άλλα λόγια, κάθε πράκτορας που φτάνει σε κάποιο κόμβο που έχει υπολογίσει σαν κόμβο συνάντησης, πρέπει να ξέρει πόσο να περιμένει πριν ξεκινήσει τον επόμενο γύρο (καταλαβαίνοντας πως εκείνος ο κόμβος δεν είναι το σημείο συνάντησης όλων των πρακτόρων).

n γνωστό k άγνωστο

Αρχικά παρατηρούμε ότι αφού το k είναι άγνωστο, μια ομάδα πρακτόρων που επισκέπτεται ταυτόχρονα τον ίδιο κόμβο, δεν μπορεί να αποφασίσει αν η συνάντηση όλων των πρακτόρων έχει επιτευχθεί μετρώντας απλά το πλήθος τους. Συνεπώς χρειάζεται μια διαφορετική στρατηγική.

Ένας γύρος αποτελείται από τρείς διαφορετικές φάσεις. Στην πρώτη φάση, ο πράκτορας εξερευνά τον δακτύλιο για να υπολογίσει τις αποστάσεις μεταξύ των σημαδιών. Κατόπιν βρίσκει τη λεξικογραφικά μέγιστη ακολουθία αποστάσεων. Στη δεύτερη φάση ο πράκτορας μετακινείται στο υπολογισμένο σημείο συνάντησης (έστω ότι χρειάζεται χρόνο t για να πάει εκεί), και περιμένει για χρόνο $n - t$ έτσι ώστε να συγχρονιστεί με τους άλλους πράκτορες για τις ανάγκες της επόμενης φάσης. Παρατηρήστε ότι ο πράκτορας δεν μπορεί να αποφασίσει εκείνη τη στιγμή εάν η συνάντηση όλων των πρακτόρων έχει επιτευχθεί απλά μετρώντας τους πράκτορες που βρίσκονται σε αυτόν τον κόμβο. Όπως θα αποδείξουμε, εάν αυτή είναι η τρίτη φορά που υπολογίζεται η ίδια ακολουθία, τότε ο πράκτορας μπορεί να είναι σίγουρος ότι αυτό είναι το πραγματικό σημείο συνάντησης. Σε αυτήν την περίπτωση, ο πράκτορας ξέρει ότι κάθε άλλος πράκτορας έχει αναγνωρίσει την ίδια ακολουθία κατά τη διάρκεια του προηγούμενου και τους τρέχοντος γύρου, και τώρα βρίσκεται στο σημείο συνάντησης που έχει υπολογίσει. Στην τρίτη φάση όλοι οι πράκτορες ειδοποιούνται ότι το σημείο συνάντησης είναι το σωστό. Οι πράκτορες που γνωρίζουν κάνουν ένα γύρο του δακτυλίου. Οι πράκτορες που δεν γνωρίζουν, περιμένουν για χρόνο n: Αν δεν συμβεί τίποτα, συνεχίζουν στον επόμενο γύρο. Αν τους επισκεφθεί ένας πράκτορας που γνωρίζει, τότε τερματίζουν τον αλγόριθμο. Η διαδικασία περιγράφεται στον Αλγόριθμο 42.

Αλγόριθμος 42 (MEET3)

1: Άφησε το σημάδι στον κόμβο εκκίνησης.
2: $S_1 = S_2 = S_3 = \emptyset$
3: Θέσε $r = 0$, επέλεξε μια κατεύθυνση και κινήσου.
4: Κινήσου για χρόνο n υπολογίζοντας τις αποστάσεις μεταξύ των σημαδιών $S = (s_1, \ldots, s_h)$.
Αν στον τελευταίο κόμβο υπάρχει σημάδι: $\mathcal{S} = d_1, \ldots, d_m = s_1, \ldots, s_h$.
Αν δεν υπάρχει σημάδι στον τελευταίο κόμβο: $\mathcal{S} = d_1, \ldots, d_m = s_1 + s_h, \ldots, s_{h-1}$.
5: $t = 0$.
6: Πήγαινε στο σημείο συνάντησης $MeetingPoint(S)$ και αύξησε τη μεταβλητή t για κάθε κόμβο που συναντάς.
7: Περίμενε για χρόνο $n - t$.
8: $S_1 = S_2$; $S_2 = S_3$; $S_3 = S$.
9: Αν $S_1 = S_2 = S_3$
10: ειδοποίησε τους άλλους πράκτορες, ταξιδεύοντας στο δακτύλιο και και μετά τερμάτισε
11: Αλλιώς περίμενε για χρόνο n
12: Αν, ενώ περιμένεις, σε συναντήσει κάποιος πράκτορας που ειδοποιεί, τερμάτισε.
/* Η συνάντηση έχει πραγματοποιηθεί. */
13: Αλλιώς, μετά την αναμονή, θέσε $r = r + 1$ και επανέλαβε από το βήμα 3.

Μπορεί να αποδειχθεί το επόμενο λήμμα:

Λήμμα 9.9 **[Flocchini et al., 2004a])** Όταν ένας πράκτορας βλέπει την ίδια συμβολοακολουθία για τρίτη φορά, όλοι οι άλλοι πράκτορες την έχουν δει τουλάχιστον δύο φορές (κατά τη διάρκεια του προηγούμενου και του τρέχοντος γύρου). ■

Θεώρημα 9.10 **([Flocchini et al., 2004a])** Αν οι πράκτορες έχουν μνήμη $O(k \log n)$, γνωρίζουν το n, και ισχύει $\gcd(k', n) = 1$ για όλα τα $k' \leq k$, και $f \leq (k-1)$ σημάδια σβήνουν είτε αμέσως μετά την τοποθέτησή τους ή αργότερα, τότε το πρόβλημα της συνάντησης μπορεί να λυθεί σε χρόνο $O(kn)$.

Απόδειξη. Από το Λήμμα 9.9 προκύπτει πως όταν ένας πράκτορας βλέπει την ίδια συμβολοακολουθία για τρίτη φορά, τότε το σημείο συνάντησης που υπολογίζει είναι το σημείο συνάντησης όλων των πρακτόρων. Όλοι οι πράκτορες συναντιούνται εκεί και μετά από χρόνο n τερματίζουν. Σε κάθε γύρο ο κάθε πράκτορας δαπανά χρόνο n για να υπολογίσει τις αποστάσεις μεταξύ των σημαδιών, έναν επιπλέον χρόνο n για να κινηθεί στο σημείο συνάντησης, και ακόμη χρόνο n για τον έλεγχο τερματισμού. Συνεπώς όλοι οι πράκτορες είναι συγχρονισμένοι και ξεκινούν κάθε γύρο ταυτόχρονα. Η συνάντηση επιτυγχάνεται μετά από το πολύ $k+2$ γύρους και ο συνολικός χρόνος που απαιτείται είναι $O(kn)$. ■

k γνωστό n άγνωστο

Όταν το n δεν είναι γνωστό η δυσκολία του προβλήματος είναι ο συγχρονισμός των πρακτόρων.

Στον αλγόριθμο που ακολουθεί, εάν περισσότεροι από ένας αλλά λιγότεροι από k πράκτορες, συναντηθούν σε κάποιον κόμβο, συμπεριφέρονται σαν ένας για το υπόλοιπο του αλγόριθμου. Αν σε έναν κόμβο βρίσκονται πολλοί πράκτορες, αυτό μπορεί να γίνει αντιληπτό από κάποιον πράκτορα που επισκέπτεται τον κόμβο (ανίχνευση πολλαπλότητας). Κάθε πράκτορας που βρίσκεται σε κάποιον κόμβο u μπορεί να μετρήσει το πλήθος των πρακτόρων που βρίσκονται στον u. Πριν από την εκκίνηση του επόμενου γύρου, οι πράκτορες που συναντιούνται σε κάποιον κόμβο που δεν είναι το πραγματικό σημείο συνάντησης όλων, συμπεριφέρονται πλέον σαν ένας πράκτορας για το υπόλοιπο του αλγορίθμου.

Αφού οι πράκτορες δεν γνωρίζουν το n, δεν μπορούν να ξέρουν πότε ακριβώς έχουν ολοκληρώσει μια επίσκεψη όλων των κόμβων του δακτυλίου. Έτσι μετακινούνται μαντεύοντας το μέγεθος του δακτυλίου σε κάθε γύρο. Αρχικά υπολογίζουν k αποστάσεις μεταξύ των σημαδιών και μαντεύουν σαν n το άθροισμα αυτών των αποστάσεων (θα χρησιμοποιήσουν αυτήν την τιμή για να συγχρονιστούν. Σε κάθε επόμενο γύρο υπολογίζουν μία λιγότερη απόσταση και αλλάζουν την εκτίμησή τους για το n. Ο αλγόριθμος είναι σχεδιασμένος έτσι ώστε:

1. οι πράκτορες έχουν το πολύ ένα γύρο διαφορά,
2. μετά από το πολύ $k-1$ γύρους, επιτυγχάνεται συνάντηση.

Η μέθοδος περιγράφεται πιο τυπικά στον Αλγόριθμο 43.

Αλγόριθμος 43 (MEET4)

1: Άφησε το σημάδι.
2: Θέσε $r = 0$, όπου το r είναι το πλήθος των γύρων που έχουν ήδη γίνει.
3: Διάλεξε μια κατεύθυνση και μετακινήσου.
4: Αν συναντήσεις κάποιον άλλον πράκτορα θα αποτελείτε μια ομάδα.
5: Υπολόγισε τις πρώτες $k - r$ αποστάσεις μεταξύ των σημαδιών, i.e., $S = (d_1, \ldots, d_{k-r})$.
6: Μάντεψε το n ως $\hat{n} = \sum_{i=1}^{k-r} d_i$.
7: Αν η S είναι περιοδική, περίμενε για $2\hat{n}$ χρόνο, θέσε $r = r + 1$, και επανέλαβε από το βήμα 3.
8: Υπολόγισε την λεξικογραφικά μέγιστη κυκλική μετάθεση S_{LMR} του S.
9: Θέσε $t = 0$.
10: Πήγαινε στον πρώτο κόμβο της S_{LMR} αυξάνοντας κατά ένα το t για κάθε κόμβο που συναντάς.
11: Περίμενε για χρόνο $2\hat{n}$ - t.
12: Αν υπάρχουν k πράκτορες στον κόμβο που βρίσκεσαι, τότε σταμάτα.
/* Η συνάντηση έχει επιτευχθεί */
13: Αλλιώς, αν υπάρχουν $1 < v < k$ πράκτορες στον κόμβο που βρίσκεσαι, τότε σχημάτισε μια ομάδα με αυτούς.
14: Θέσε $r = r + 1$ και επανέλαβε από το βήμα 3.

Τα απόμενα τρία λήμματα χρησιμοποιούνται στην απόδειξη του Θεωρήματος 9.14. Οι αποδείξεις τους βρίσκονται στο άρθρο [Flocchini et al., 2004a]. Στο Λήμμα 9.11 αποδεικνύεται ότι δύο οποιοιδήποτε πράκτορες έχουν διαφορά μικρότερη από ένα γύρο και συνεπώς ένας πράκτορας χρειάζεται να περιμένει μόνο $\frac{3\hat{n}}{2}$ χρόνο για πράκτορες που έχουν υπολογίσει της ίδια με αυτόν ακολουθία.

Λήμμα 9.11 Έστω a ο πρώτος πράκτορας που τελειώνει μαντεύοντας το n στο γύρο r, όπου $0 \leq r \leq k - 1$, και έστω τ ο χρόνος κατά τον οποίον ο πράκτορας a κάνει αυτόν τον υπολογισμό. Όλοι οι άλλοι πράκτορες μέχρι τη χρονική στιγμή τ είτε θα τελειώσουν το γύρο $r - 1$, ή θα ενταχθούν στην ίδια ομάδα με τον a. ■

Λήμμα 9.12 Οι πράκτορες που υπολογίζουν την ίδια ακολουθία αποστάσεων S, με το πολύ μια κυκλική μετάθεση, βρίσκονται στον ίδιο γύρο. Αν η S δεν είναι περιοδική, τότε αυτοί οι πράκτορες θα συναντηθούν στο τέλος του γύρου. ■

Λήμμα 9.13 Αν το πολύ f σημάδια έχουν αποτύχει, τότε οι πράκτορες δεν θα εκτελέσουν το γύρο $r = f + 1$ στον αλγόριθμο. ■

Θεώρημα 9.14 ([Flocchini et al., 2004a]) Αν οι πράκτορες έχουν μνήμη $O(k \log n)$, γνωρίζουν το k, ισχύει $\gcd(k', n) = 1$ για κάθε $k' \leq k$, και $f \leq (k - 1)$ σημάδια μπορούν να αποτύχουν οποιαδήποτε στιγμή, τότε το πρόβλημα της συνάντησης λύνεται μετά από $O(k^2 n)$ χρόνο.

Απόδειξη. Έστω $f \leq k-1$ ο αριθμός των σημαδιών που αποτυγχάνουν. Σύμφωνα με το Λήμμα 9.13 κανείς από τους πράκτορες δεν θα εκτελέσει περισσότερους από $f+1$ γύρους στον αλγόριθμο. Ας υποθέσουμε ότι η συνάντηση δεν έχει επιτευχθεί μέχρι το τέλος του γύρου $r = f$. Έστω a^* ο πρώτος πράκτορας που ξεκινά το γύρο $r = f+1$ και έστω t_0 η στιγμή που ο a^* ξεκινά το γύρο $r = f+1$. Αφού f σημάδια έχουν αποτύχει, η εκτίμηση του a^* για το n στο γύρο $r = f+1$ είναι σωστή, δηλαδή, $\hat{n} = n$. Οι υπόλοιποι πράκτορες υπολογίζουν την ίδια μή-περιοδική ακολουθία με εκείνη που έχει υπολογίσει ο a^*, με τη διαφορά μιας κυκλικής μετάθεσης και συνεπώς από το Λήμμα 9.12 συμπεραίνουμε ότι η συνάντηση επιτυγχάνεται στο τέλος του γύρου f.

Το πλήθος των σημαδιών f που αποτυγχάνουν είναι το πολύ $k-1$, και έτσι, όπως αναφέρθηκε παραπάνω, εκτελούνται το πολύ k γύροι. Κάθε γύρος διαρκεί το πολύ $k(n-1)$ χρόνο, δηλαδή το γινόμενο του πλήθους των αποστάσεων που έχουν καταμετρηθεί και της μέγιστης απόστασης. Άρα ο χρόνος που απαιτείται είναι $O(k^2 n)$. Επειδή υπολογίζονται το πολύ k αποστάσεις και η μέγιστη απόσταση είναι $n-1$, η μνήμη που απαιτείται είναι $O(k \log n)$. ■

9.3.3 Το κόστος της αποτυχίας των σημαδιών

Όταν τα σημάδια αποτυγχάνουν, οι απαιτήσεις σε χρόνο και μνήμη για την επίλυση του προβλήματος αυξάνουν. Στον Πίνακα 9.2, αντιπαραθέτουμε τις απαιτήσεις σε μνήμη και χρόνο για τη συνάντηση με ή χωρίς αποτυχία των σημαδιών.

Πίνακας 9.2: Το κόστος της αποτυχίας των σημαδιών

Αποτυχία Σημαδιών	Γνώση	Χρόνος	Μνήμη	Αλγόριθμος
Ποτέ	n ή k	$O(n)$	$O(\log n)$	Αλγόριθμος 23
		$O(n \log k)$	$O(\log k)$	Αλγόριθμος 25
Πρώτη τοποθέτηση	n	$O(n)$	$O(k \log n)$	Αλγόριθμος 40
	k	$O(k\, n)$	$O(k \log n)$	Αλγόριθμος 41
Οποτεδήποτε	n	$O(k\, n)$	$O(k \log n)$	Αλγόριθμος 42
	k	$O(k^2\, n)$	$O(k \log n)$	Αλγόριθμος 43

9.4 Συνάντηση σε δέντρα

Σε αυτήν την ενότητα εξετάζουμε τη συνάντηση δύο πρακτόρων (χωρίς σημάδια) σε ένα συγχρονισμένο δέντρο. Σε αυτήν την περίπτωση το πρόβλημα λύνεται αν και μόνο αν οι αρχικές θέσεις των πρακτόρων είναι μή-συμμετρικές. Στο άρθρο [Fraigniaud and Pelc, 2008] αποδεικνύεται ότι η ελάχιστη μνήμη που χρειάζεται για τη λύση του προβλήματος σε οποιοδήποτε δέντρο με το πολύ n κόμβους είναι $\Theta(\log n)$ bits. Ενώ με $O(\log n)$ bits μνήμη είναι δυνατή η συνάντηση το κάτω φράγμα προκύπτει από το γεγονός ότι ο πράκτορας πρέπει να έχει αρκετή μνήμη για να

ξεχωρίζει μέχρι και $n-1$ ακμές που προσπίπτουν σε κάποιον κόμβο. Σε αυτό το άρθρο αναφέρεται επίσης πως δύο πράκτορες με σταθερή μνήμη δεν μπορούν να συναντηθούν σε οποιοδήποτε δέντρο φραγμένου βαθμού.

Οι κόμβοι του δέντρου δεν έχουν ετικέττες, ενώ οι ακμές που προσπίπτουν στον ίδιο κόμβο έχουν (τοπικά) διαφορετικές ετικέττες. Ένα ζευγάρι κόμβων του δέντρου καλείται συμμετρικό αν υπάρχει ένας αυτομορφισμός (τυπικά, ένας αυτομορφισμός $f : V \to V$ είναι μια συνάρτηση '1-1' και 'επί') που απεικονίζει κάποιον κόμβο σε κάποιον άλλο διατηρώντας τις ετικέττες των ακμών.

Είναι εύκολο να δούμε ότι αν οι αρχικές θέσεις των πρακτόρων δεν είναι συμμετρικές τότε οι πράκτορες μπορούν να συναντηθούν ως εξής: Έστω ένα γενικευμένο δέντρο. Είναι γνωστό ότι τα δέντρα έχουν είτε έναν κεντρικό κόμβο είτε μια κεντρική ακμή.

- Αν το δέντρο έχει έναν κεντρικό κόμβο τότε οι δύο πράκτορες με αρκετή μνήμη μπορούν να συναντηθούν σε αυτόν τον κόμβο, ανεξάρτητα από τις αρχικές τους θέσεις.
- Αν το δέντρο έχει μια κεντρική ακμή, τότε για μή-συμμετρικές αρχικές θέσεις, το ένα από τα άκρα της κεντρικής ακμής μπορεί να διαχωριστεί από το άλλο και οι πράκτορες μπορούν να συναντηθούν σε κάποιο από αυτά.
- Τέλος, για συμμετρικές αρχικές θέσεις σε ένα δέντρο με κεντρική ακμή, αν οι πράκτορες ξεκινούν ταυτόχρονα δεν είναι δυνατόν να συναντηθούν, αφού όλες τους οι πράξεις θα είναι συμμετρικές ως προς την κεντρική ακμή.

Το βασικό αποτέλεσμα φαίνεται στο παρακάτω θεώρημα.

Θεώρημα 9.15 ([Fraigniaud and Pelc, 2008])

1. Δύο πανομοιότυποι πράκτορες με $O(\log n)$ bits μνήμη ο καθένας μπορούν να λύσουν το πρόβλημα της συνάντησης με ανίχνευση σε οποιοδήποτε δέντρο μεn κόμβους.
2. Δύο πανομοιότυποι πράκτορες με σταθερή μνήμη δεν μπορούν να λύσουν το πρόβλημα της συνάντησης χωρίς ανίχνευση σε όλα τα δέντρα-αλυσίδες. ■

Την απόδειξη του παραπάνω θεωρήματος μπορεί να βρει ο ενδιαφερόμενος αναγνώστης στο άρθρο [Fraigniaud and Pelc, 2008].

9.5 Συνάντηση σε γενικευμένα γραφήματα

Το πρόβλημα της συνάντησης σε γενικευμένα γραφήματα (δηλαδή μή-κατευθυνόμενα συνδεδεμένα δίκτυα με τοπικά διαφορετικές ετικέττες στις ακμές που προσπίπτουν στον ίδιο κόμβο, αλλά χωρίς ετικέττες στους κόμβους) μελετούν στο άρθρο [Dessmark et al., 2003]. Στο μοντέλο που ερευνούν στο παραπάνω άρθρο, οι πράκτορες μπορεί να ξεκινούν σε διαφορετικές στιγμές (που αποφασίζονται από κάποιον adversary) και έχουν διαφορετικές ταυτότητες που είναι θετικοί ακέραιοι.

Πριν αναφέρουμε τα κυριότερα αποτελέσματα εισάγουμε κάποιες βασικές έννοιες που χρησιμοποιούνται. Η διαφορά μεταξύ των χρόνων εκκίνησης των πρακτόρων συμβολίζεται με τ. Οι πράκτορες έχουν ταυτότητες τους αριθμούς L_1, L_2 και έστω ℓ ο μικρότερος από αυτούς. Το γράφημα έχει n κόμβους και Δ είναι ο μέγιστος βαθμός του. Επιπλέον D είναι η αρχική απόσταση μεταξύ των πρακτόρων. Ο πράκτορας που ξεκινά πρώτος καλείται *πρώτος* πράκτορας και ο άλλος πράκτορας καλείται *δεύτερος*. Στην περίπτωση ταυτόχρονης εκκίνησης, ο πρώτος πράκτορας καλείται πράκτορας 1. (Ένας πράκτορας δεν γνωρίζει αν είναι πρώτος ή δεύτερος.)

Το άρθρο [Dessmark et al., 2003] αναλύει το πρόβλημα της συνάντησης στο παραπάνω μοντέλο σε δέντρα, δακτύλιους και γενικευμένα γραφήματα. Το κύριο θεώρημα έχει ως εξής.

Θεώρημα 9.16 ([Dessmark et al., 2003]) Δύο πράκτορες με διαφορετικές ταυτότητες:

1. Μπορούν να συναντηθούν σε οποιοδήποτε δέντρο n κόμβων μετά από $O(n + \log \ell)$ βήματα. Επιπλέον, υπάρχουν δέντρα n κόμβων στα οποία οποιοσδήποτε αλγόριθμος συνάντησης απαιτεί $\Omega(n + \log \ell)$ χρόνο, ακόμη και με ταυτόχρονη εκκίνηση των πρακτόρων.

2. Στο μοντέλο της ταυτόχρονης εκκίνησης, ο ελάχιστος χρόνος που απαιτείται για συνάντηση σε ένα δακτύλιο είναι $\Theta(D \log \ell)$. Επιπλέον για εκκινήσεις όχι υποχρεωτικά ταυτόχρονες ο ελάχιστος χρόνος για συνάντηση σε έναν δακτύλιο n κόμβων είναι $\Omega(n + D \log \ell)$.

3. Σε γενικευμένα γραφήματα με ταυτόχρονη εκκίνηση, η συνάντηση μπορεί να επιτευχθεί μέσα σε $O(D\Delta^D \log \ell)$ βήματα. ■

Την απόδειξη του παραπάνω θεωρήματος μπορεί να βρει ο ενδιαφερόμενος αναγνώστης στο άρθρο [Dessmark et al., 2003].

9.6 Σχόλια και βιβλιογραφικές παρατηρήσεις

Η μελέτη πιθανοτικών αλγορίθμων για το πρόβλημα της συνάντησης έχει τις ρίζες της στον George Pólya ο οποίος χαιρόταν τους περιπάτους στο δάσος ενώ σκεφτόταν πάνω σε μαθηματικά προβλήματα. Σύμφωνα με το άρθρο [Alexanderson, 2000] (βλέπε επίσης το άρθρο [Hayes, 2010]) 'Κάποια μέρα στον περίπατό του συνάντησε κάποιον από τους φοιτητές του να κάνει βόλτα με τη φίλη του. Λίγο αργότερα συναντήθηκαν ξανά και ξανά. Όντας σε αμηχανία, του προκλήθηκε η απορία πόσο πιθανό ήταν να συναντήσεις κάποιον άλλον τυχαία ακολουθώντας τα μονοπάτια στο δάσος'. Αυτές οι σκέψεις ήταν η αφετηρία μιας μελέτης που οδήγησε στο σημαντικό του άρθρο [Pólya, 1921].

Από τότε πολλοί ερευνητές παρατήρησαν ότι το πρόβλημα της συνάντησης μπορεί να λυθεί από πανομοιότυπους πράκτορες σε ένα ανώνυμο δίκτυο δίνοντας οδηγία στους πράκτορες να εκτελέσουν μια τυχαία διαδρομή. Ο αναμενόμενος χρόνος συνάντησης αποδεικνύεται ότι είναι μια πολυωνυμική συνάρτηση του μεγέθους του δικτύου και σχετίζεται με τον χρόνο κάλυψης του δικτύου. Ο ενδιαφερόμενος αναγνώστης μπορεί να βρει στο άρθρο [Motwani and Raghavan, 1995]

ορισμούς που σχετίζονται με τυχαίες διαδρομές, ενώ στο άρθρο [Coppersmith et al., 1993] μπορεί να βρει μια ανάλυση για τον χρόνο συνάντησης που πετυχαίνουν οι τυχαίες διαδρομές.

Για μια πολύ ενδιαφέρουσα συζήτηση σε συνεχή περιβάλλοντα με πιθανοτικούς πράκτορες ο αναγνώστης παραπέμπεται στο βιβλίο [Alpern and Gal, 2003].

Μια διαφορετική και ενδιαφέρουσα εκδοχή του προβλήματος συνάντησης για μηνύματα ενός bit σε γενικευμένα γραφήματα εξετάζεται στο άρθρο [Métivier et al., 2000]. Κάθε κόμβος u του δικτύου επαναλαμβάνει για πάντα τις εξής πράξεις: 1) επιλέγει τυχαία κάποιον από τους γειτονικούς του κόμβους, έστω $s(u)$, 2) στέλνει το ψηφίο 1 στον $s(u)$ και το ψηφίο 0 στους υπόλοιπους γειτονικούς του κόμβους, 3) λαμβάνει μηνύματα από όλους τους γείτονές του. Η συνάντηση συμβαίνει μεταξύ των u και $s(u)$, αν ο u λαμβάνει 1 από τον $s(u)$. Οι συγγραφείς του άρθρου αποδεικνύουν πάνω και κάτω φράγματα στον αναμενόμενο μέγιστο χρόνο συνάντησης για στιγμιότυπα σε γενικευμένους γράφους αλλά και σε ειδικές τοπολογίες γραφημάτων όπως δακτύλιους, αλυσίδες και πλήρη γραφήματα. Επιπλέον λεπτομέρειες μπορούν να βρεθούν στο άρθρο [Métivier et al., 2003].

Στο άρθρο [Das et al., 2008] οι συγγραφείς μελετούν το πρόβλημα της συνάντησης k πρακτόρων που μεταφέρουν σημάδια που μπορεί να αποτύχουν, για ασύγχρονους δακτύλιους n κόμβων για οποιεσδήποτε τιμές των n και k για τις οποίες το πρόβλημα λύνεται. Παρουσιάζουν επίσης έναν πιο πολύπλοκο αλγόριθμο για την περίπτωση μιας γενικής και άγνωστης τοπολογίας που απαιτεί $O(k\Delta^{2n})$ κινήσεις, όπου Δ είναι ο μέγιστος βαθμός του γραφήματος. Ένα σχετικό άρθρο για το πρόβλημα της συνάντησης σε δακτύλιο χρησιμοποιώντας σημάδια που μπορεί να αποτύχουν είναι το [Das, 2008].

Ένα σχετικό άρθρο για το πρόβλημα της συνάντησης σε ασύγχρονα γενικευμένα γραφήματα είναι το [Czyzowicz et al., 2010]. Εκεί οι συγγραφείς αποδεικνύουν ότι δύο πράκτορες με διαφορετικές ταυτότητες μπορούν να συναντηθούν σε οποιοδήποτε, πιθανόν άπειρο, άγνωστης τοπολογίας και ασύγχρονο γράφημα εκτελώντας ντετερμινιστικούς αλγόριθμους. Μελετούν επίσης το πρόβλημα της συνάντησης σε γεωμετρικά μοντέλα. Σχετικά με το πρόβλημα της συνάντησης ρομπότ σε γεωμετρικά περιβάλλοντα μια καλή αρχή είναι τα [Roy and Dudek, 2001] και [Ando et al., 1999].

Βιβλιογραφία

[Alexanderson, 2000] Alexanderson, G. L. (2000). *The random walks of George Pólya*. The Mathematical Association of America.

[Alpern, 1995] Alpern, S. (1995). The rendezvous search problem. *SIAM Journal of Control and Optimization*, 33:673–683.

[Alpern and Gal, 2003] Alpern, S. and Gal, S. (2003). *The Theory of Search Games and Rendezvous*. Kluwer Academic Publishers.

[Ando et al., 1999] Ando, H., Oasa, Y., Suzuki, I., and Yamashita, M. (1999). A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *I.E.E.E. Transactions on Robotics and Animation*, 15(5):818–828.

[Coppersmith et al., 1993] Coppersmith, D., Tetali, P., and Winkler, P. (1993). Collisions among random walks on a graph. *SIAM J. of Discrete Mathematics*, 6:363–374.

[Czyzowicz et al., 2010] Czyzowicz, J., Labourel, A., and Pelc, A. (2010). How to meet asynchronously (almost) everywhere. In *Proceedings of 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 22–30.

[Das, 2008] Das, S. (2008). Mobile agent rendezvous in a ring using faulty tokens. In *Proceedings of 9th International Conference in Distributed Computing and Networking*, LNCS 4904, pages 292–297.

[Das et al., 2008] Das, S., Mihalak, M., Sramek, R., Vicari, E., and Widmayer, P. (2008). Rendezvous of mobile agents when tokens fail anytime. In *Proceedings of 12th International Conference on Principles of Distributed Systems*, LNCS 5401, pages 463–480.

[Dessmark et al., 2003] Dessmark, A., Fraigniaud, P., and Pelc, A. (2003). Deterministic rendezvous in graphs. In *Proc. of 11th Annual European Symp. on Algorithms*, pages 184–195.

[Flocchini et al., 2004a] Flocchini, P., Kranakis, E., Krizanc, D., Luccio, F., Santoro, N., and Sawchuk, C. (2004a). Mobile agent rendezvous when tokens fail. In *Proceedings of 11th International Colloquium on Structural Information and Communication Complexity*, LNCS 3104, pages 161–172.

[Flocchini et al., 2004b] Flocchini, P., Kranakis, E., Krizanc, D., Santoro, N., and Sawchuk, C. (2004b). Multiple mobile agent rendezvous in the ring. In *Proceedings of 6th Latin American Theoretical Informatics Symposium*, pages 599–608.

[Fraigniaud and Pelc, 2008] Fraigniaud, P. and Pelc, A. (2008). Deterministic rendezvous in trees with little memory. In *Proc. of 22nd Int. Symp. on Distributed Computing*, LNCS 5318, pages 242–256.

[Hayes, 2010] Hayes, B. (2010). The teetotaler's walk. *Bit-Player (web resource)*.

[Kranakis and Krizanc, 2007] Kranakis, E. and Krizanc, D. (2007). An algorithmic theory of mobile agents. In *Proc. of 2nd Symp. on Trustworthy Global Computing*, LNCS 4661, pages 86–97.

[Kranakis et al., 2008] Kranakis, E., Krizanc, D., and Morin, P. (2008). Randomized Rendez-Vous with Limited Memory. In *Proc. of 8th Latin American Theoretical Informatics Symp.*, LNCS 4957, pages 605–614.

[Métivier et al., 2000] Métivier, Y., Saheb, N., and Zemmari, A. (2000). Randomized rendezvous. *Mathematics and Computer Science: Algorithms, trees, combinatorics and probabilities*, pages 183–194.

[Métivier et al., 2003] Métivier, Y., Saheb, N., and Zemmari, A. (2003). Analysis of a randomized rendezvous algorithm. *Information and Computation*, 184(1):109–128.

[Motwani and Raghavan, 1995] Motwani, R. and Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press, New York.

[Pólya, 1921] Pólya, G. (1921). Über eine aufgabe betreffend die irrfahrt im strassennetz. *Mathematische Annalen*, 84:149–160.

[Roy and Dudek, 2001] Roy, N. and Dudek, G. (2001). Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11:117–136.

[Sawchuk, 2004] Sawchuk, C. (2004). *Mobile Agent Rendezvous in the Ring*. PhD thesis, School of Computer Science, Carleton University, Ottawa, Canada.

ΚΕΦΑΛΑΙΟ 10

Τρέχουσες και Μελλοντικές Ερευνητικές Κατευθύνσεις

Σε αυτό το κεφάλαιο περιγράφουμε νέες κατευθύνσεις και συζητούμε τις προοπτικές της αλγοριθμικής θεωρίας στους κατανεμημένους υπολογισμούς.

Ο Claude Shannon [Shannon, 1952] αναφέρεται ως ο πρώτος που σχεδίασε ένα πεπερασμένο αυτόματο ικανό να εξερευνήσει ένα οποιονδήποτε λαβύρινθο μεγέθους 5×5. Από τότε πολλοί επιστήμονες μελέτησαν προβλήματα εξερεύνησης και αναζήτησης σε γραφήματα αλλά και γεωμετρικά περιβάλλοντα με τη βοήθεια κατάλληλων κινητών πρακτόρων. Παρόλ' αυτά η περιοχή αυτή παραμένει μια ανεξάντλητη πηγή προβλημάτων που αναζητούν λύση.

Σήμερα, η έρευνα έχει εστιάσει στη σχεδίαση και ανάλυση αλγορίθμων για την εξερεύνηση γραφημάτων, την αναζήτηση πληροφοριών, τα προβλήματα της συνάντησης και της ανακάλυψης εχθρικών κόμβων, κλπ. Μερικά από αυτά μελετήσαμε στις προηγούμενες σελίδες του βιβλίου αυτού. Τα προβλήματα αυτά μαζί με άλλα σχετικά προβλήματα, όπως το πρόβλημα της επίτευξης ενός συγκεκριμένου σχηματισμού από πράκτορες σε ένα περιβάλλον, το πρόβλημα του καθαρισμού ενός μολυσμένου δικτύου και άλλα παρουσιάζονται πολύ συχνά τόσο σε δίκτυα επικοινωνίας όσο και σε κοινωνικά δίκτυα. Οι μέθοδοι λύσεις τέτοιων προβλημάτων βρίσκουν εφαρμογές σε πολλούς τομείς όπως η συντήρηση δικτύων επικοινωνίας, το ηλεκτρονικό εμπόριο, η εξερεύνηση ενός περιβάλλοντος με τη βοήθεια ρομπότ αλλά και η διαχείριση και λήψη αποφάσεων σε κατανεμημένα περιβάλλοντα (π.χ., δημοπρασίες στο Internet).

Το γενικό ερώτημα που συνεχίζει να απασχολεί τους ερευνητές είναι το εξής:

> *Πώς μπορούμε να εξερευνήσουμε ένα δίκτυο και να αναζητήσουμε πληροφορίες όσο το δυνατόν πιο γρήγορα και με το μικρότερο δυνατό κόστος, με τη βοήθεια κινητών πρακτόρων;*

Μολονότι το παραπάνω ερώτημα είναι πολύ γενικό, αναφέρουμε παρακάτω μερικές από τις πιο ενδιαφέρουσες τρέχουσες και μελλοντικές κατευθύνσεις της έρευνας.

Ισοζύγια χρόνου/μνήμης/αρχικής γνώσης με τεχνικές της Θεωρίας Παιγνίων: Η μελέτη του ισοζύγιου μεταξύ χρόνου και μνήμης σε προβλήματα των κατανεμημένων υπολογισμών δεν έχει μελετηθεί αρκετά. Η κατασκευή μεθόδων που εκτελούνται σε βέλτιστο χρόνο έχοντας σαν παράμετρο τη μνήμη που χρησιμοποιείται ή την αρχική γνώση για το περιβάλλον, θα είναι πολύ χρήσιμες στη σχεδίαση μηχανισμών που χρησιμοποιούνται στην οικονομία αλλά και άλλες περιοχές στις οποίες η εκπροσώπηση των χρηστών με πράκτορες που δρουν με μόνο κριτήριο το συμφέρον τους (και συνήθως δεν γνωρίζουν τις στρατηγικές των άλλων πρακτόρων) βρίσκει πολλές εφαρμογές. Γενικώς η μελέτη προβλημάτων κατανεμημένων υπολογισμών με τεχνικές της Αλγοριθμικής Θεωρίας Παιγνίων είναι ένα πολύ ενδιαφέρον και ελάχιστα μελετημένο πεδίο. Για παράδειγμα στο άρθρο [Karakostas and Markou, 2014] μελετάται με τεχνικές της Θεωρίας Παιγνίων, το πρόβλημα της διατήρησης της συνεκτικότητας ενός δικτύου χωρίς κεντρική αρχή όταν οι (ακίνητοι) πράκτορες ενδιαφέρονται μόνο για τη δική τους επικοινωνία.

Συστήματα με πολλαπλούς πράκτορες: Τα συστήματα με πολλαπλούς πράκτορες είναι πολύ χρήσιμα στην αναζήτηση και στην εξερεύνηση, ιδιαίτερα όταν ο συγχρονισμός τους δεν είναι δεδομένος. Η βαθύτερη μελέτη αυτής της περιοχής θα μπορούσε να προσφέρει νέες πιο αποδοτικές τεχνικές για την αναζήτηση στο διαδίκτυο. Ακόμη και πολύ απλά προβλήματα δεν έχουν μελετηθεί πλήρως σε ασύγχρονα μοντέλα δικτύων με πολλαπλούς πράκτορες.

Σενάρια με πράκτορες που έχουν ελάχιστη μνήμη και περιορισμένες δυνατότητες: Οι πράκτορες με περιορισμένη μνήμη (δηλαδή ανεξάρτητη από το μέγεθος του δικτύου) και δυνατότητα να αντιλαμβάνονται μόνο μια περιορισμένη τοπική περιοχή του περιβάλλοντός τους για να πάρουν αποφάσεις, αποτελούν στοιχεία μοντέλων που έχουν πολλές εφαρμογές στην εξερεύνηση δικτύων, ιδιαίτερα όταν οι κόμβοι τους έχουν περιορισμούς σε μνήμη και δυνατότητες. Για παράδειγμα ένα σημαντικό πρόβλημα που παρά την εκτεταμένη μελέτη του παραμένει ανοικτό είναι πόσο γρήγορα μπορεί ένας πράκτορας να εξερευνήσει ένα οποιοδήποτε δίκτυο όταν έχει πρόσβαση μόνο σε τοπική πληροφορία [Kalyanasundaram and Pruhs, 1994, Megow et al., 2011, Miyazaki et al., 2009, Dobrev et al., 2012].

Γεωμετρικά περιβάλλοντα: Η πλοήγηση και δρομολόγηση σε γεωμετρικά περιβάλλοντα δύο και τριών διαστάσεων με τη βοήθεια μόνο τοπικής πληροφορίας είναι μια περιοχή με πολλά ανοιχτά προβλήματα και εφαρμογές, όπως για παράδειγμα στην εξερεύνηση ενός επικίνδυνου περιβάλλοντος (π.χ., ναρκοπέδιο) στα οποίο είναι προτιμητέα η χρήση φθηνών και αναλώσιμων ρομπότ για την εξερεύνηση.

Ασφάλεια κατανεμημένων συστημάτων με κινητούς πράκτορες: Όπως τονίστηκε σε διάφορα κεφάλαια αυτού του βιβλίου, η ασφάλεια των κατανεμημένων συστημάτων στα οποία δρουν κινητοί πράκτορες έχει θεμελιώδη σημασία ενώ παράλληλα είναι μία δύσκολη και όχι πολύ μελετημένη πε-

ριοχή, ιδίως όσον αφορά την ανακάλυψη εχθρικών κόμβων μεγάλης δύναμης ή εχθρικών κινητών πρακτόρων ή την λύση του προβλήματος παρά την ύπαρξη τέτοιων εχθρικών οντοτήτων. Σε αυτή την κατεύθυνση, ένα από τα προβλήματα που έχει σχετικά πρόσφατα μελετηθεί είναι η περιοδική επίσκεψη των κόμβων ενός δικτύου από κινητούς πράκτορες και η μεταφορά πληροφοριών από τους κόμβους αυτούς σε έναν δεδομένο κόμβο παρά την ύπαρξη εχθρικών κόμβων μεγάλης δύναμης [Kràlovic and Miklìk, 2010, Bampas et al., 2015]. Υπάρχουν όμως ελάχιστα αποτελέσματα για σενάρια στα οποία συμμετέχουν εχθρικοί πράκτορες που είναι ικανοί να αλλοιώσουν τα περιεχόμενα κόμβων ή άλλων πρακτόρων. Ένα άλλο σενάριο που έχει μελετηθεί πρόσφατα, είναι η επίτευξη της συνάντησης όλων των πρακτόρων που δρουν σε ένα δίκτυο παρά την ύπαρξη ενός εχθρικού κινητού πράκτορα που έχει τη δυνατότητα να εμποδίζει οποιονδήποτε άλλον πράκτορα να μετακινηθεί στη θέση που καταλαμβάνει κάθε φορά ο εχθρικός πράκτορας [Das et al., 2015].

Καθαρισμός μολυσμένων δικτύων με τη βοήθεια κινητών πρακτόρων: Ένα σημαντικό πρόβλημα με πολλές εφαρμογές είναι η σχεδίαση και ανάλυση μεθόδων για τον καθαρισμό ενός μολυσμένου δικτύου με τη βοήθεια κινητών πρακτόρων. Το πρόβλημα έχει μελετηθεί αρκετά για μια σειρά σεναρίων που έχουν να κάνουν με τους τρόπους που επιτρέπεται να κινούνται και να καθαρίζουν οι πράκτορες στο δίκτυο, κυρίως όμως για πράκτορες με διαφορετικές ετικέτες που παίρνουν εντολές από μια κεντρική αρχή και συνεπώς δεν δρουν αυτόνομα [Breisch, 1967, Parson, 1978, Kirousis and Papadimitriou, 1986, Bienstock and Seymour, 1991, Blin et al., 2012, Blin et al., 2013]. Όταν οι πράκτορες είναι αναγκασμένοι να δρουν αυτόνομα, το πρόβλημα έχει μελετηθεί ελάχιστα. Τέτοια σενάρια έχουν εφαρμογές σε καταστάσεις έκτακτης ανάγκης, όπως είναι η κατάσβεση μιας πυρκαγιάς που εξαπλώνεται, και ιδιαίτερα σε περιπτώσεις όπου τα δίκτυα επικοινωνίας μεταξύ των χρηστών που προσπαθούν να βοηθήσουν στην κατάσβεση είτε έχουν καταρρεύσει είτε δεν μπορούν να λειτουργήσουν λόγω των συνθηκών (π.χ., παγίδευση σε ορυχεία).

Βιβλιογραφία

[Bampas et al., 2015] Bampas, E., Leonardos, N., Markou, E., Pagourtzis, A., and Petrolia, M. (2015). Improved periodic data retrieval in asynchronous rings with a faulty host. *Theoretical Computer Science*, 608:231–254.

[Bienstock and Seymour, 1991] Bienstock, D. and Seymour, P. (1991). Monotonicity in graph searching. *Journal of Algorithms*, 12:239–245.

[Blin et al., 2012] Blin, L., Burman, J., and Nisse, N. (2012). Brief announcement: Distributed exclusive and perpetual tree searching. In *26th Int. Symp. on Distributed Computing (DISC)*, volume 7611 of *LNCS*, pages 403–404. Springer.

[Blin et al., 2013] Blin, L., Burman, J., and Nisse, N. (2013). Exclusive graph searching. In *21st European Symposium on Algorithms (ESA)*, volume 8125 of *LNCS*, pages 181–192. Springer.

[Breisch, 1967] Breisch, R. (1967). An intuitive approach to speleotopology. *Southwestern Cavers VI(5)*, pages 72–78.

[Das et al., 2015] Das, S., Luccio, F., and Markou, E. (2015). Mobile agents rendezvous in spite of a malicious agent. In *Proc. 11th International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)*, LNCS 9536, pages 211–224.

[Dobrev et al., 2012] Dobrev, S., Kràlovic, R., and Markou, E. (2012). Online graph exploration with advice. In *Proc. 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, LNCS 7355, pages 267–278.

[Kalyanasundaram and Pruhs, 1994] Kalyanasundaram, B. and Pruhs, K. R. (1994). Constructing competitive tours from local information. *Theoretical Computer Science*, 130(1):125 – 138.

[Karakostas and Markou, 2014] Karakostas, G. and Markou, E. (2014). Emergency connectivity in ad-hoc networks with selfish nodes. *Algorithmica*, 68(2):358–389.

[Kirousis and Papadimitriou, 1986] Kirousis, L. and Papadimitriou, C. (1986). Searching and pebbling. *Theoretical Computer Science*, 47(2):205–218.

[Kràlovic and Miklìk, 2010] Kràlovic, R. and Miklìk, S. (2010). Periodic data retrieval problem in rings containing a malicious host. In *Proc. of 17th Int. Colloquium on Structural Information and Communication Complexity*, pages 156–167.

[Megow et al., 2011] Megow, N., Mehlhorn, K., and Schweitzer, P. (2011). Online graph exploration: New results on old and new algorithms. In *ICALP (2)*, pages 478–489.

[Miyazaki et al., 2009] Miyazaki, S., Morimoto, N., and Okabe, Y. (2009). The online graph exploration problem on restricted graphs. *IEICE Transactions on Information and Systems*, 92(9):1620–1627.

[Parson, 1978] Parson, T. (1978). The search number of a connected graph. In *Proceedings of 9th Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 549–554. Utilitas Mathematica.

[Shannon, 1952] Shannon, C. (1952). Presentation of a Maze Solving Machine in Cybernetics: Circular, Causal and Feedback Mechanisms in Biological and Social Systems. In *Transactions Eighth Conf., von Foerster, H., Mead, M. and Teuber, HL (eds). New York, Josiah Macy Jr. Foundation*, pages 169–181.

Ευρετήριο όρων