Κατανεμημένοι Υπολογισμοί και Ασφάλεια Δικτύων

Ευριπίδης Μάρκου[1]

[1]Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων

Άνοιξη 2008

Σκελετός

Παράλληλοι Υπολογισμοί I

Πολλοί ανεξάρτητοι επεξεργαστές

- κάθε επεξεργαστής εκτελεί το δικό του πρόγραμμα
- οι επεξεργαστές επικοινωνούν μεταξύ τους στιγμιαία και συγχρονισμένα χρησιμοποιώντας ένα μεγάλο χώρο κοινής μνήμης

Τυπικά οι επεξεργαστές:

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

Παράλληλοι Υπολογισμοί II

- Απλό μοντέλο για τον σχεδιασμό αλγόριθμων αλλά δύσκολο τεχνικά να επιτευχθεί.
- Ο σκοπός είναι η ελαχιστοποίηση της χρονικής πολυπλοκότητας του προβλήματος.

Πολλαπλασιασμός Πινάκων - Σειριακοί αλγόριθμοι

Σειριακοί αλγόριθμοι

- απλός αλγόριθμος με $O(n^3)$ πολλαπλασιασμούς
- αλγόριθμος *Strassen (1969)* με $O(n^{\log_2 7}) \simeq O(n^{2.807})$
- αλγόριθμος *Coppersmith & Winograd (1990)* με $O(n^{2.376})$

Η πολυπλοκότητα του προβλήματος είναι $\Omega(n^2)$.

Πολλαπλασιασμός Πινάκων - Παράλληλοι αλγόριθμοι Ι

Χρησιμοποιούμε n^3 επεξεργαστές:

- κάθε επεξεργαστής κάνει έναν πολλαπλασιασμό,
- κάθε ένας από n^2 επεξεργαστές κάνει σειριακά $n-1$ προσθέσεις

Η πολυπλοκότητα του αλγόριθμου είναι $\Theta(n)$.

Πολλαπλασιασμός Πινάκων - Παράλληλοι αλγόριθμοι II

Χρησιμοποιούμε n^3 επεξεργαστές:

- κάθε επεξεργαστής κάνει έναν πολλαπλασιασμό,
- $n/2$ επεξεργαστές κάνουν παράλληλα τις προσθέσεις που απαιτούνται για τον υπολογισμό καθενός από τα n^2 στοιχεία του τελικού πίνακα.

Η πολυπλοκότητα του αλγόριθμου είναι $\Theta(\log n)$.

Graph reachability - Σειριακοί αλγόριθμοι

Δεδομένου ενός (κατευθυνόμενου) γραφήματος και δύο κόμβων u, v υπάρχει μονοπάτι που να συνδέει τους u, v;

Σειριακοί αλγόριθμοι

Επίσκεψη των κόμβων του γραφήματος (*DFS, BFS, ...*).
Η πολυπλοκότητα του προβλήματος είναι $\Theta(n)$.

Graph reachability - Παράλληλος αλγόριθμος Ι

Δεδομένου ενός (κατευθυνόμενου) γραφήματος και δύο κόμβων u, v υπάρχει μονοπάτι που να συνδέει τους u, v;

Παράλληλος αλγόριθμος

- Πάρε τον πίνακα γειτνίασης (*adjacency matrix*) A_{ij} του γραφήματος (που περιλαμβάνει τα self loops) και υπολόγισε τους πίνακες $A_{ij}^2, A_{ij}^4, ..., A_{ij}^n$.
- Εάν το στοιχείο $A_{uv}^n = 1$ τότε υπάρχει μονοπάτι από το u στο v αλλιώς όχι.

Ορθότητα του αλγόριθμου:

Αποδεικνύουμε ότι $A_{ij}^k = 1$ αν και μόνο αν υπάρχει μονοπάτι μήκους το πολύ k από τον κόμβο i στον j.

Graph reachability - Παράλληλος αλγόριθμος ΙΙ

Δεδομένου ενός (κατευθυνόμενου) γραφήματος και δύο κόμβων u, v υπάρχει μονοπάτι που να συνδέει τους u, v;

Παράλληλος αλγόριθμος

- Χρησιμοποιούμε n^3 επεξεργαστές και η πολυπλοκότητα του αλγόριθμου είναι $\Theta(\log^2 n)$.
- Μάλιστα με την ίδια πολυπλοκότητα μπορούμε να απαντήσουμε στις ερωτήσεις:

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

Κατανεμημένο Περιβάλλον

Επίλυση προβλημάτων με αποτελεσματικό τρόπο σε κατανεμημένο περιβάλλον υπολογισμού.

Κατανεμημένο Περιβάλλον

- δίκτυα επικοινωνίας
- κατανεμημένες βάσεις δεδομένων

Τα δεδομένα και οι επεξεργαστές είναι κατανεμημένοι στο δίκτυο. Η θέσεις τους αλλά και η τοπολογία του δικτύου μπορεί να αλλάζει δυναμικά. Δεν υπάρχει κοινή μνήμη.

Κατανεμημένο Περιβάλλον

Ένας πεπερασμένος αριθμός από πράκτορες που μπορούν να κάνουν υπολογισμούς και να επικοινωνούν μεταξύ τους (ανταλλάσοντας μηνύματα, γράφοντας σε κόμβους, αφήνοντας σημάδια κλπ.) με σκοπό την επίτευξη ενός κοινού στόχου:

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

Πράκτορες και Πρωτόκολλα

Πράκτορες

Κάθε πράκτορας έχει τη δυνατότητα να εκτελεί υπολογισμούς και η συνδυασμένη προσπάθεια όλων λύνει το πρόβλημα ή εκτελεί τη λειτουργία στο δίκτυο.

Πρωτόκολλα

Για τη λύση του προβλήματος πρέπει να σχεδιάσουμε έναν κατανεμημένο αλγόριθμο ή κατανεμημένο πρωτόκολλο για τους πράκτορες: ένα σύνολο κανόνων που καθορίζουν τί πρέπει να κάνει κάθε πράκτορας. Οι πράκτορες εκτελούν αυτόνομα τους κανόνες.

Κατανεμημένοι Αλγόριθμοι

- Ορθότητα: Το πρωτόκολλο λύνει το πρόβλημα.
- Αποτελεσματικότητα: Το πρωτόκολλο έχει 'μικρό' κόστος.

Προβλήματα - ***Leader Election***

Το πρόβλημα της εκλογής αρχηγού (***Leader Election***)

Αρχικά όλοι οι πράκτορες βρίσκονται στην ίδια κατάσταση (*available*). Στο τέλος όλοι οι πράκτορες βρίσκονται στην ίδια κατάσταση (*follower*) εκτός από έναν που βρίσκεται στην κατάσταση *leader*.

Υποθέσεις Μοντέλου

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

Προβλήματα - ***Leader Election***

Αλγόριθμος για το πρόβλημα της εκλογής αρχηγού

1. πήγαινε στην κατάσταση *leader*,
2. στείλε σε όλους την ταυτότητά σου (*broadcast*),
3. αν λάβεις κάποια ταυτότητα μικρότερη από τη δική σου τότε άλλαξε την κατάστασή σου σε *follower*.

Προβλήματα - ***Rendezvous***

Το πρόβλημα της συνάντησης (***Rendezvous***)

Πώς πρέπει να κινηθούν δύο mobile agents έτσι ώστε να συναντηθούν (rendezvous) σε κάποιο κόμβο του δικτύου;

Ενδιαφέρουσες ερωτήσεις

Ποιο είναι το πιο `αδύναμο' σενάριο για το οποίο η συνάντηση είναι δυνατή;

- τι ικανότητες πρέπει να έχουν οι πράκτορες (π.χ. να μπορούν να αφήνουν μηνύματα στους κόμβους, να αφήνουν σημάδια (*tokens*) στους κόμβους)
- μνήμη (π.χ. για να μετρούν), γνώση (π.χ. τον αριθμό των κόμβων του δικτύου)

Προβλήματα - ***Rendezvous***

Διάφορα μοντέλα

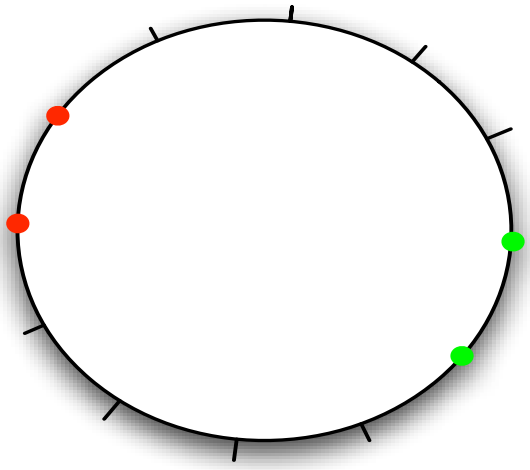
[Yu, Yung, 1996]: Μη-επιλύσιμο σε γενικά γραφήματα αν οι πράκτορες χρησιμοποιούν τον ίδιο ντετερμινιστικό αλγόριθμο.

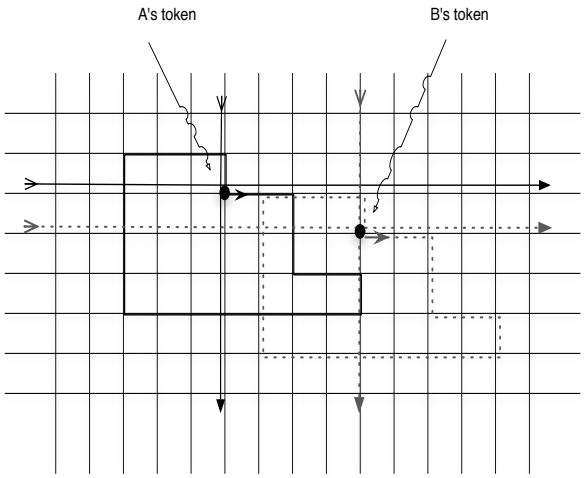
- [Baston, Gal, 2001], [Dessmark, Fraigniaud, Pelc, 2003]: Πιθανοτικοί αλγόριθμοι ή διαφορετικοί ντετερμινιστικοί αλγόριθμοι και μνήμη.
- [Barriere et al, 2003], [Dobrev et al, 2004]: Ανώνυμοι πράκτορες που αφήνουν μηνύματα στους κόμβους.
- [Kranakis et al, 2003], [Sawchuk, 2004], [Gasieniec et al, 2006]: Ανώνυμοι πράκτορες που αφήνουν *tokens* στους κόμβους ενός δακτύλιου.

Προβλήματα - ***Rendezvous***

Μπορούμε να σχεδιάσουμε

έναν ντετερμινιστικό αλγόριθμο που να οδηγεί τους δύο πράκτορες σε συνάντηση μέσα σε πεπερασμένο χρόνο; (ανεξάρτητα από τις αρχικές τους θέσεις ή το μέγεθος του δικτύου).

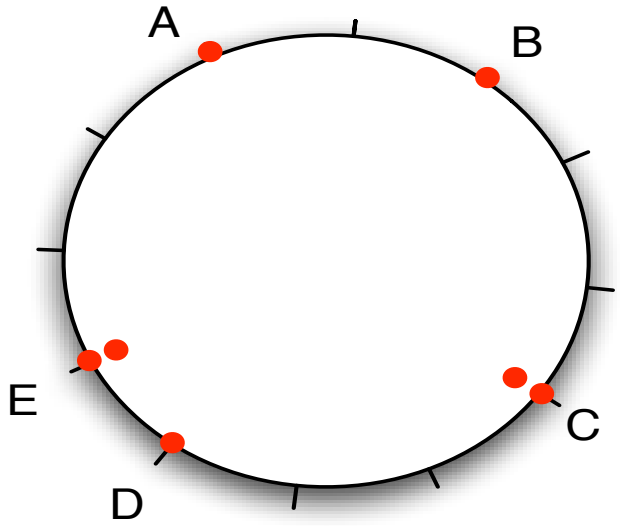




Προβλήματα - ***Gathering***

Μπορούμε να σχεδιάσουμε

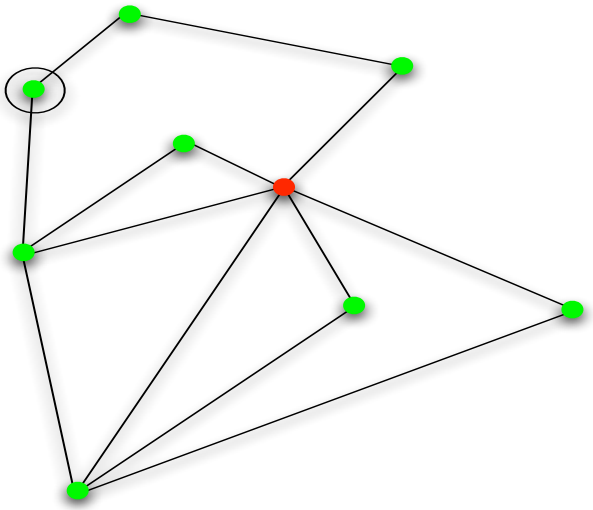
έναν ντετερμινιστικό αλγόριθμο που να οδηγεί όλους τους πράκτορες σε συνάντηση μέσα σε πεπερασμένο χρόνο; (ανεξάρτητα από τις αρχικές τους θέσεις ή το μέγεθος του δικτύου).



Προβλήματα - ***Black Hole***

Το πρόβλημα της μαύρης τρύπας (***Black Hole***)

Η μαύρη τρύπα είναι ένας εχθρικός κόμβος που καταστρέφει οποιονδήποτε πράκτορα την επισκέπτεται χωρίς να αφήνει ίχνη. Μια ομάδα πρακτόρων ψάχνει το δίκτυο με αποστολή την ανακάλυψη και αναφορά των εχθρικών κόμβων.



Προβλήματα - ***Black Hole***

Ερωτήσεις

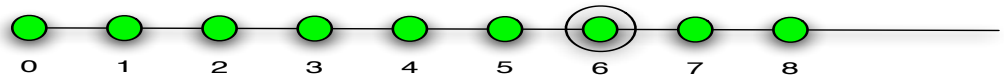
- Πόσοι πράκτορες χρειάζονται για να ανακαλύψουν τη μαύρη τρύπα;
- Τί γνώση χρειάζεται να έχουν οι πράκτορες;
- Σε πόσο χρόνο μπορούν να ανακαλύψουν τη μαύρη τρύπα;

Προβλήματα - ***Black Hole***

Η περίπτωση της γραμμής

Ένας βέλτιστος αλγόριθμος που ανακαλύπτει τη μαύρη τρύπα σε χρόνο $4n - 8$.

- Split(s-1, s+1);
- Split(s-2, s+2);
- Walk(s-1);
- Walk-and-Probe(1);
- Split(0, s+3);
- Walk(s+2);
- Walk-and-Probe(n);

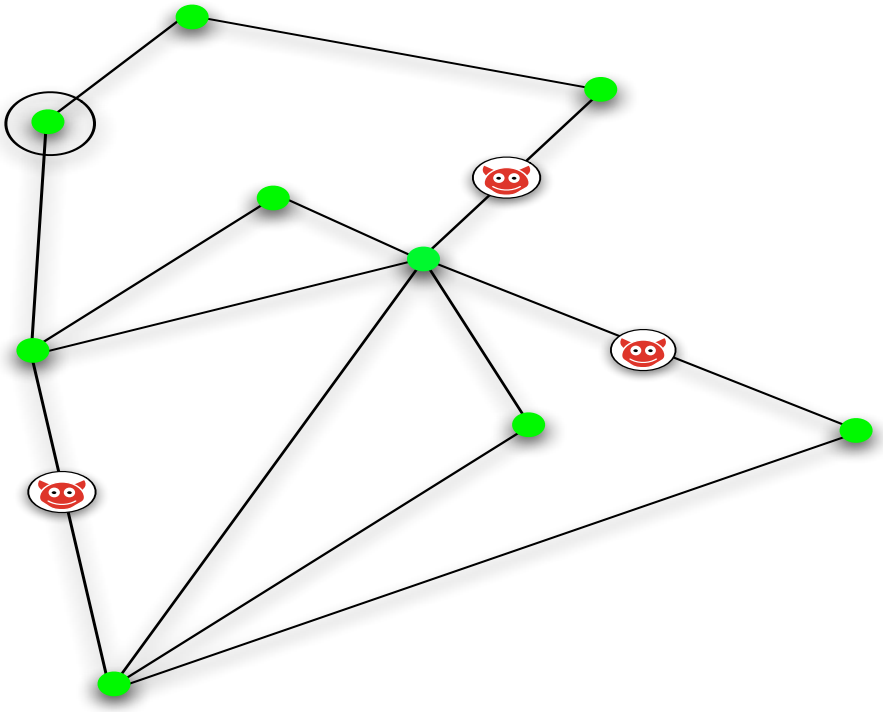


Προβλήματα - ***Fault Tolerance***

Προβλήματα Ανοχής Σφάλματος (***Fault Tolerance***)

Δίνεται ένα γράφημα με ετικέτες (*labels*) στους κόμβους, του οποίου κάποιες ακμές είναι κομμένες. 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. Να βρεθεί ένας αλγόριθμος που να εγγυάται την επίσκεψη όλων των κόμβων του δικτύου στον ελάχιστο χρόνο.

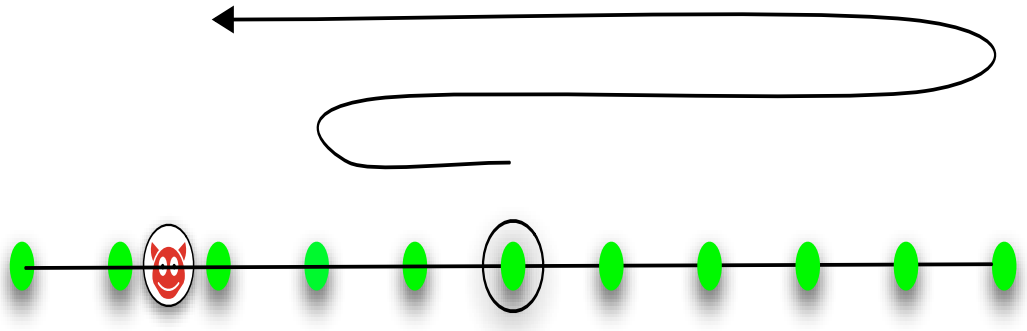
Προβλήματα - ***Fault Tolerance***



Προβλήματα - ***Fault Tolerance***

Η περίπτωση της γραμμής

Ένας βέλτιστος αλγόριθμος με *overhead* $\frac{2a+7}{a+6}$.



Κινητοί Πράκτορες (***Mobile Agents***)

Πλεονεκτήματα I

- μετάβαση 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,

Κινητοί Πράκτορες (***Mobile Agents***)

Πλεονεκτήματα II

- καλύτερη διαχείριση στις περιπτώσεις εμφάνισης σφαλμάτων στο δίκτυο,
- οι υπολογισμοί μπορούν να εκτελεστούν ασύγχρονα και αυτόνομα ακόμη και χωρίς την παρουσία του χρήστη,
- αποτελεσματικότητα,
- νέες εφαρμογές σε *ad-hoc* δίκτυα.

Μειονεκτήματα

- ασφάλεια.

Κινητοί Πράκτορες

Εφαρμογές

- ηλεκτρονικό εμπόριο,
- συντήρηση του δικτύου,
- *Personal Digital Assistants (PDAs)*

Κινητοί Πράκτορες

Ασφάλεια

- επίθεση ενός πράκτορα σε κόμβο του δικτύου,
  - masquerading agent
  - denial of service
  - unauthorized access to data
- επίθεση ενός κόμβου του δικτύου σε πράκτορα,
  - masquerade
  - repudiation
  - unauthorized access
- επίθεση ενός πράκτορα σε άλλον πράκτορα.
  - masquerade
  - denial of service
  - eavesdropping
  - alteration

Κινητοί Πράκτορες

Απαιτήσεις Ασφάλειας

- data privacy,
- communications privacy,
- location privacy,
- integrity,
- accountability,
- availability,
- anonymity.

Κινητοί Πράκτορες

Ενέργειες για ασφάλεια

- η βάση του πράκτορα είναι ασφαλής,
- αποφυγή προγραμματιστικών λαθών,
- κρυπτογραφία,
- καταγραφή της ιστορίας του πράκτορα,
- οι πράκτορες ενημερώνουν τη βάση τους,
- παρακολούθηση των πρακτόρων από άλλους πράκτορες,

Κρίσιμη ερώτηση

Μπορούμε να λύσουμε τα ζητήματα ασφάλειας στους κινητούς πράκτορες και ταυτόχρονα να διατηρήσουμε τα πλεονεκτήματα (π.χ. αποτελεσματικότητα);

Κινητοί Πράκτορες

Τυπικός ορισμός του πράκτορα

- Ντετερμινιστικός αλγόριθμος
- Ένα ρομπότ με μνήμη το οποίο τρέχει έναν ντετερμινιστικό αλγόριθμο. Το ρομπότ μπορεί:

  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 *tokens*, κλπ.
- Πεπερασμένο αυτόματο (*Finite Moore Automaton*)

Κινητοί Πράκτορες

Πεπερασμένο αυτόματο $A = (X, Y, S, \delta, \lambda, S_0)$

- $X \subseteq C_A \times C_E$:

  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,
- $\delta : S \times X \rightarrow S$: η συνάρτηση μετάβασης,
- $\lambda : S \rightarrow Y$: μια συνάρτηση που καθορίζει τί πράξη θα κάνει ο πράκτορας.

Κινητοί Πράκτορες

Πεπερασμένο αυτόματο $A = (X, Y, S, \delta, \lambda, S_0)$

- Αρχικά ο πράκτορας A βρίσκεται σε κάποιον κόμβο u_0 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 το *port* από το οποίο έφτασε στον v και την κατάσταση του κόμβου).
- Τα παραπάνω σε συνδυασμό με την τρέχουσα κατάστασή του σ προκαλούν μια μετάβαση σε μια νέα κατάσταση σ' (σύμφωνα με την δ).
- Η νέα κατάσταση σ' καθορίζει μια πράξη $\lambda(\sigma')$ κλπ.

Κινητοί Πράκτορες

Ιδιότητες

- Συνήθως επιθυμούμε οι πράκτορες να σταματήσουν όταν τελειώσουν. Αυτό δεν είναι πάντα εύκολο να επιτευχθεί.
- Σε τέτοιες περιπτώσεις οι πράκτορες δεν ξέρουν εάν η αποστολή τους έχει τελειώσει. Από την άλλη, οι πράκτορες μπορεί να λειτουργούν για πάντα χωρίς να ολοκληρώνουν την αποστολή τους.
- π.χ. μια ομάδα από το πολύ 3 πράκτορες που μπορούν να επικοινωνούν χωρίς να χρειάζεται να συναντηθούν, είναι αδύνατον να εξερευνήσει όλα τα δυνατά γραφήματα (ούτε καν όλα τα δυνατά επίπεδα γραφήματα μέγιστου βαθμού 3).
- Οι πράκτορες εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο και σπάνε τις συμμετρίες τους είτε χρησιμοποιώντας τις ταυτότητές τους, είτε παρατηρώντας το περιβάλλον τους.

Κατανεμημένοι Υπολογισμοί - Αξιώματα

Καθυστερήσεις Επικοινωνίας

Εάν δεν υπάρχουν σφάλματα στο δίκτυο τότε θεωρούμε ότι οι επικοινωνίες μπορεί να καθυστερούν για πεπερασμένο χρόνο (αλλά συνήθως μη-προβλέψιμο (ασύγχρονα δίκτυα».

Τοπικός προσανατολισμός

Ο πράκτορας μπορεί να ξεχωρίσει τις ακμές που οδηγούν στους γείτονές του. (edge labelling or port labelling, locally consistent but not necessarily globally)

Κόστος και Πολυπλοκότητα

Επικοινωνία

- ο συνολικός αριθμός μηνυμάτων M που στάλθηκαν,
- ο λόγος M/V,
- ο λόγος M/E,
- ο συνολικός αριθμός των *bits* B που στάλθηκαν.

Κόστος και Πολυπλοκότητα

Χρόνος

- Η χρονική πολυπλοκότητα αναφέρεται στο χρόνο που μεσολαβεί από τη στιγμή που ο πρώτος πράκτορας ξεκινά την εκτέλεση του αλγόριθμου μέχρι τη στιγμή που ο τελευταίος πράκτορας τελειώνει.
- Η χρονική διάρκεια είναι μη-προβλέψιμη σε ασύγχρονα συστήματα, και για αυτό το λόγο, προκειμένου να εκτιμήσουμε έναν αλγόριθμο (ή να συγκρίνουμε αλγόριθμους) χρησιμοποιούμε την *ideal time complexity* η οποία είναι η χρονική πολυπλοκότητα που θα είχε ο αλγόριθμος εάν το σύστημα ήταν συγχρονισμένο με καθυστέρηση 1.

Τυπικοί περιορισμοί του μοντέλου

Τυπικοί περιορισμοί του μοντέλου

- Συνεκτικότητα: υπάρχει μονοπάτι από κάθε πράκτορα σε κάθε άλλον πράκτορα,
- Ολική Αξιοπιστία: δεν υπάρχουν σφάλματα στο δίκτυο,
- Ακμές δύο κατευθύνσεων.

Broadcasting

Το πρόβλημα

Ένας πράκτορας έχει μια πληροφορία που θέλει να μοιραστεί με όλους τους υπόλοιπους πράκτορες στο δίκτυο.

Το μοντέλο

- Οι πράκτορες βρίσκονται στους κόμβους του δικτύου και μπορούν να ανταλλάσουν μηνύματα.
- Η αναπαράσταση του δικτύου γίνεται από ένα μή-κατευθυνόμενο, συνδεδεμένο γράφημα.
- Το δίκτυο δεν έχει σφάλματα.

Broadcasting

Πολυπλοκότητα - κάτω φράγματα

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

Αλγόριθμος

1. Άν έχεις την πληροφορία (*initiator*) τότε στείλε την στους γείτονές σου;
2. Άν έλαβες την πληροφορία τότε στείλε την στους γείτονές σου;

Broadcasting

Αλγόριθμος

1. done:= FALSE;
2. **Repeat**
3. Άν έχεις την πληροφορία (*initiator*) τότε
4. {στείλε την στους γείτονές σου;
5. done:= TRUE; }
6. Άν έλαβες την πληροφορία τότε
7. {στείλε την στους γείτονές σου;
8. done:= TRUE; }
9. **Until** done;

Broadcasting

Ιδιότητες Αλγόριθμου

- Τώρα ο αλγόριθμος τερματίζει αλλά κανείς από τους πράκτορες δεν ξέρει πότε συμβαίνει αυτό.
- Ένας πράκτορας μπορεί να πάει στην κατάσταση done πολύ πριν τους άλλους.

Χρονική πολυπλοκότητα αλγόριθμου

$O(d(G))$, όπου $d(G) = \max_{u,v} sh.path(u, v)$.

Broadcasting

Πολυπλοκότητα μηνυμάτων αλγόριθμου

Κάθε πράκτορας u στέλνει ακριβώς μια φορά ένα μήνυμα μέσω όλων των ακμών που προσπίπτουν στον u. Άρα για κάθε ακμή (v_1, v_2) υπάρχει ένα μήνυμα που ταξιδεύει από τον v_1 στον v_2 και ένα μήνυμα που ταξιδεύει από τον v_2 στον v_1. Συνεπώς $M = 2|E|$.

Αλγόριθμος με μικρότερη πολυπλοκότητα μηνυμάτων

Εάν ένας πράκτορας λάβει το μήνυμα μέσω του port k τότε δεν στέλνει μήνυμα μέσω αυτού του port. $M = 2|E| - n + 1$.

Broadcasting

Ειδικές τοπολογίες δικτύων

- Δέντρο: $T = n - 1$, $M = n - 1$.
- Πλήρες γράφημα: $T = 1$, $M = n - 1$.

Common Knowledge

Στον προηγούμενο αλγόριθμο, στο τέλος όλοι οι πράκτορες ξέρουν την πληροφορία (απόδειξη με επαγωγή). Όμως δεν υπάρχει κάποια χρονική στιγμή κατά την οποία οι πράκτορες να γνωρίζουν ότι όλοι ξέρουν την πληροφορία.

Κοινή γνώση

Η πληροφορία ρ είναι κοινή γνώση για την ομάδα 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...

Common Knowledge

Κοινή γνώση

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

Κατηγορίες κοινής γνώσης

- metric: (π.χ. αριθμός κόμβων),
- topological: ιδιότητες γραφήματος (π.χ. τοπολογία δακτυλίου),
- maps: τοπικοί ή πλήρεις χάρτες,
- other info: nodes' labelling

Wake-Up

Το πρόβλημα

Κάποιοι από τους πράκτορες βρίσκονται στην κατάσταση *awake* και οι υπόλοιποι στην κατάσταση *asleep*. Ο σκοπός είναι στο τέλος όλοι να βρίσκονται στην κατάσταση *awake*.

Το μοντέλο

- Οι πράκτορες βρίσκονται στους κόμβους του δικτύου και μπορούν να ανταλλάσουν μηνύματα.
- Η αναπαράσταση του δικτύου γίνεται από ένα μή-κατευθυνόμενο, συνδεδεμένο γράφημα.
- Το δίκτυο δεν έχει σφάλματα.

Το πρόβλημα *Broadcasting* είναι ειδική περίπτωση του προβλήματος *Wake-Up*.

Wake-Up

Πολυπλοκότητα - κάτω φράγματα

- $\Omega(n)$ χρόνος.
- $\Omega(|E|)$ μηνύματα ($2|E| \geq M \geq 2|E| - n + 1$).

Ειδικές τοπολογίες δικτύων

- Δέντρο: $M = n + k_* - 2$ μηνύματα, όπου k_* είναι ο αριθμός των πρακτόρων που αρχικά βρίσκονται στην κατάσταση *awake*.
- Πλήρες γράφημα: Το κάτω φράγμα για τον αριθμό των μηνυμάτων παραμένει $\Omega(n^2)$ σε αντίθεση με το πρόβλημα *Broadcasting* που σε αυτήν την περίπτωση έχει πολυπλοκότητα $n - 1$.

The Gathering Problem

Μια ομάδα από κινητούς πράκτορες καλείται να εκτελέσει διάφορες εργασίες σε ένα δίκτυο. Κατά την κατασκευή αλγορίθμων με αυτό το σκοπό, συχνά χρειάζεται να αντιμετωπιστούν πρωταρχικές λειτουργίες όπως *wakeup*, *traversal*, **gathering**, *election*.

The Gathering Problem

Κινητοί πράκτορες (**robots**), οι οποίοι αρχικά βρίσκονται σε διαφορετικά σημεία του δικτύου, πρέπει να συναντηθούν σε ένα σημείο (το οποίο δεν είναι προκαθορισμένο), και να παραμείνουν εκεί.

The Gathering Problem

Ένα πολύ αδύναμο - γενικό σενάριο

- δεν υπάρχει κάποια κεντρική αρχή
- οι πράκτορες λειτουργούν ασύγχρονα
- δεν υπάρχει κοινή γνώση κάποιας παραμέτρου του δικτύου
- οι πράκτορες δεν έχουν διαφορετικές ταυτότητες
- δεν υπάρχει κοινό σύστημα συντεταγμένων
- δεν είναι δυνατή η επικοινωνία μεταξύ των πρακτόρων
- οι πράκτορες έχουν μνήμη αλλά 'ξεχνούν'

The Gathering Problem

Δυσκολίες του προβλήματος

Τα ρομπότ

- πρέπει να σπάσουν τις συμμετρίες προκειμένου να συμφωνήσουν σε ένα κοινό σημείο συνάντησης,
- δεν μπορούν να επικοινωνήσουν αλλά πρέπει να αποφασίσουν για το πως θα κινηθούν μόνο παρατηρώντας το περιβάλλον τους,
- δεν έχουν μνήμη για να κρατήσουν παλιές παρατηρήσεις,
- δεν έχουν διαφορετικές ταυτότητες,
- λειτουργούν ασύγχρονα σε ένα ανώνυμο δίκτυο.

Δυνατά (περιορισμένα) μοντέλα...

[G. De Marco et al, 2005], [A. Dessmark et al, to appear], [D. Kowalski, A. Pelc, 2004]

Τα ρομπότ έχουν διαφορετικές ταυτότητες.

[P. Flocchini et al, 2004]

Τα ρομπότ έχουν στη διάθεσή τους **tokens**.

...και πιο γενικά μοντέλα

Ανώνυμα πανομοιότυπα ρομπότ: δεν στέλνουν μηνύματα και επικοινωνούν με το περιβάλλον μόνο παρατηρώντας το.
Σημείωση: Αυτό το μοντέλο χρησιμοποιήθηκε παλιότερα για τη μελέτη της περίπτωσης που τα ρομπότ μπορούν να κινούνται ελεύθερα στο επίπεδο.

[M. Cieliebak, 2004]

Τα ρομπότ έχουν μνήμη.

[H. Ando et al, 1999], [P. Flocchini, D. Peleg, G. Prencipe, ...]

Τα ρομπότ είναι **oblivious**, δηλαδή δεν μπορούν να κρατήσουν στη μνήμη τους παλιές παρατηρήσεις.

[H. Ando et al, 1999], [I. Suzuki, M. Yamashita, 1999]

Τα ρομπότ λειτουργούν συγχρονισμένα.

Πιο κοντά στο μοντέλο που θα εξετάσουμε.

[M. Cieliebak et al, 2004], [R. Cohen, D. Peleg, 2004]

Τα ρομπότ λειτουργούν ασύγχρονα.

[P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, 2005]

Η συνάντηση είναι δυνατή στο ασύγχρονο μοντέλο, εάν τα ρομπότ έχουν συμφωνήσει σε ένα κοινό σύστημα συντεταγμένων, ακόμη και εάν έχουν περιορισμένη ορατότητα.

[M. Cieliebak, 2003], [G. Prencipe, 2005]

Χωρίς κοινό σύστημα συντεταγμένων:

- η συνάντηση είναι δυνατή εάν τα ρομπότ έχουν επιπλέον την ικανότητα της ανίχνευσης πολλαπλότητας,
- χωρίς αυτή την ικανότητα η συνάντηση είναι αδύνατη.

Περιγραφή του μοντέλου (I)

Ένας αριθμός από πανομοιότυπους κινητούς πράκτορες (ρομπότ)

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

Κατά τη διάρκεια ενός κύκλου, ένα ρομπότ,

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

Περιγραφή του μοντέλου (II)

Κανόνες

1. Οι κύκλοι εκτελούνται ασύγχρονα για κάθε ρομπότ.
2. Οι κινήσεις είναι στιγμιαίες.
3. Τα ρομπότ είναι **oblivious**, δεν κρατούν στη μνήμη τους παλιές παρατηρήσεις.
4. Τα ρομπότ είναι ανώνυμα και εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο.

Περιγραφή του μοντέλου (III)

Παρατηρήσεις

- Οι αποφάσεις των ρομπότ μπορεί να βασίζονται σε πολύ παλιές παρατηρήσεις.
- Ο κόμβος στον οποίο πρόκειται το ρομπότ να κινηθεί, αποφασίζεται κατά τη διάρκεια της λειτουργίας *Compute* και η απόφαση αυτή βασίζεται αποκλειστικά στις θέσεις των άλλων ρομπότ έτσι όπως παρατηρήθηκαν κατά την προηγούμενη λειτουργία *Look*.

Αποτελέσματα

Η συνάντηση είναι αδύνατη για οποιοδήποτε αριθμό από ρομπότ σε αυτό το γενικό σενάριο. Προσθέτουμε την ικανότητα της ανίχνευσης πολλαπλότητας (*multiplicity detection*). Δηλαδή τα ρομπότ μπορούν να ανιχνεύσουν κατά τη διαδικασία *Look* εάν υπάρχουν πάνω από ένα ρομπότ στον ίδιο κόμβο.

Για περιττό αριθμό από ρομπότ:

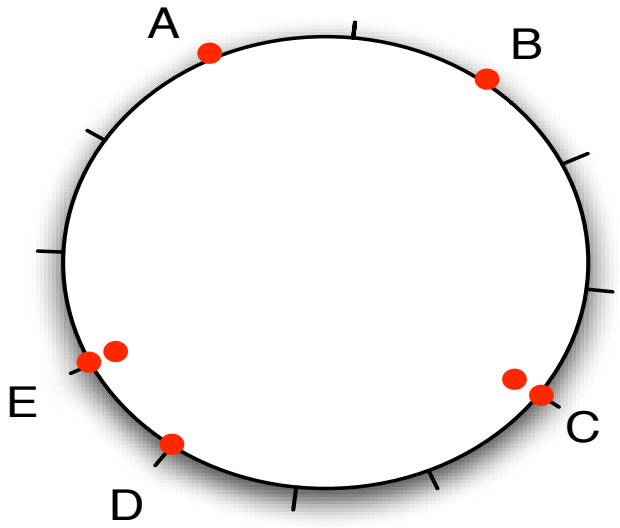
- Η συνάντηση είναι δυνατή αν και μόνο αν η αρχική σύνθεση (*configuration*) στο δακτύλιο είναι περιοδική.
- Δίνουμε έναν αλγόριθμο που οδηγεί τους πράκτορες σε συνάντηση σε κάθε τέτοια περίπτωση.

Αποτελέσματα

Για άρτιο αριθμό από ρομπότ:

- 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.

Η εικόνα ενός σχηματισμού



Ο σχηματισμός από το A
$((2,3,3,1,3),(5,9))$

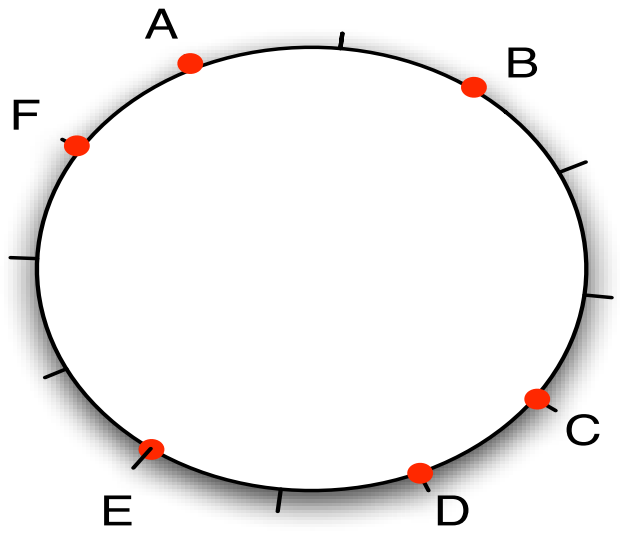
Ο σχηματισμός από το B
$((3,3,1,3,2),(3,7))$

Ο σχηματισμός από το C
$((3,1,3,2,3),(0,4))$

Η εικόνα που έχει το ρομπότ A

$$\{((2,3,3,1,3),(5,9)),((3,1,3,3,2),(3,7))\}$$

Ένας περιοδικός σχηματισμός



Περιοδικός σχηματισμός

Ένας σχηματισμός c χωρίς πολλαπλότητες καλείται περιοδικός εάν είναι παράθεση τουλάχιστον δύο αντιγράφων κάποιας υποακολουθίας p της c.

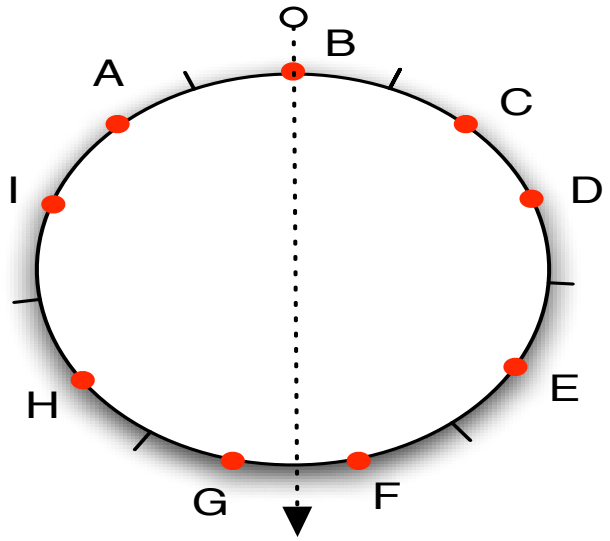
Ο σχηματισμός από το A

$(2, 3, 1, 2, 3, 1)$

Η εικόνα που έχει το ρομπότ A

$$\{(2, 3, 1, 2, 3, 1), (1, 3, 2, 1, 3, 2)\}$$

Ένας συμμετρικός (και περιοδικός σχηματισμός)



Συμμετρικός σχηματισμός

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.

Η εικόνα που έχει το ρομπότ A

$$\{(2, 2, 1, 2, 2, 1, 2, 2, 1), (1, 2, 2, 1, 2, 2, 1, 2, 2)\}$$

Ιδιότητες

Ορισμός

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

Λήμμα

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

Λήμμα

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

Αρνητικά αποτελέσματα

Λήμματα

1. Η συνάντηση για 2 ρομπότ είναι αδύνατη σε οποιοδήποτε σχηματισμό και οποιονδήποτε δακτύλιο.
2. Εάν δεν υπάρχει η ικανότητα της ανίχνευσης πολλαπλότητας τότε η συνάντηση οποιουδήποτε αριθμού $k > 1$ ρομπότ είναι αδύνατη σε οποιονδήποτε δακτύλιο.

Θεωρήματα

1. Η συνάντηση είναι αδύνατη για κάθε περιοδικό σχηματισμό.
2. Η συνάντηση είναι αδύνατη για κάθε συμμετρικό ακμής -ακμής σχηματισμό.

Σχηματισμοί με ακριβώς μία πολλαπλότητα

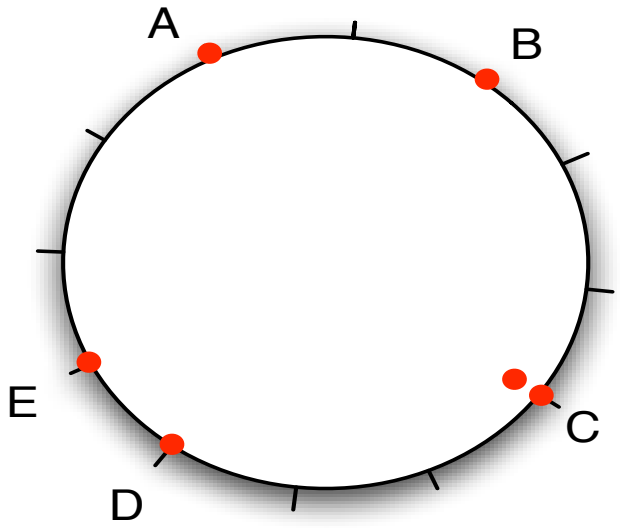
Αλγόριθμος ***Single-Multiplicity-Gathering***

if R is at the multiplicity **then** do not move
else
  if none of the segments between R
    and the multiplicity is free
  then do not move
  else move towards the multiplicity along the shortest
    of the free segments or along any of them
    in the case of equality.

Λήμμα

Ο αλγόριθμος *Single-Multiplicity-Gathering* οδηγεί μετά από πεπερασμένο χρόνο τα ρομπότ σε συνάντηση εάν ο αρχικός σχηματισμός είχε ακριβώς μία πολλαπλότητα.

Η συνάντηση σε σχηματισμούς με μία πολλαπλότητα



Ο σχηματισμός από το A
$((2,3,3,1,3),(5))$

Ο σχηματισμός από το B
$((3,3,1,3,2),(3))$

Ο σχηματισμός από το C
$((3,1,3,2,3),(0))$

Η εικόνα που έχει το ρομπότ A

$$\{((2,3,3,1,3),(5)),((3,1,3,3,2),(7))\}$$

Η συνάντηση σε ***rigid*** σχηματισμούς

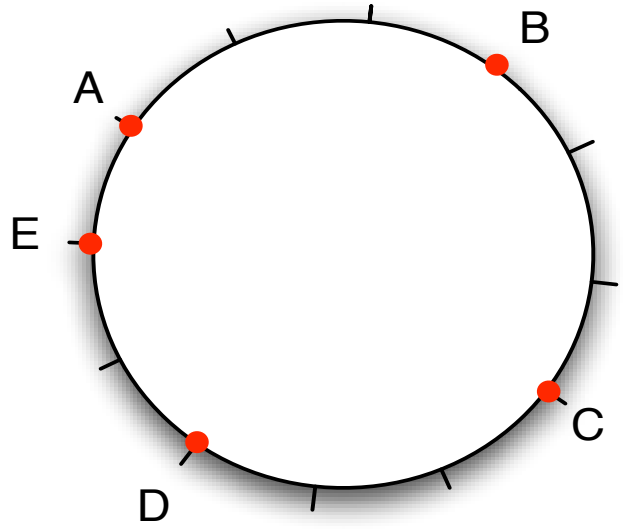
Αλγόριθμος ***Rigid-Gathering***

- elect a pair of neighboring robots A, B at a maximum distance
- choose the robot which has the other neighboring robot closer ($*$ ties can be broken easily $*$)
- move the elected robot towards the direction which increases the maximum distance

Λήμμα

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

Η συνάντηση σε ***rigid*** σχηματισμούς



Ο σχηματισμός από το A
$(3, 3, 3, 2, 1)$

Ο σχηματισμός από το B
$(3, 3, 2, 1, 3)$

Ο σχηματισμός από το C
$(3, 2, 1, 3, 3)$

Η εικόνα που έχει το ρομπότ A

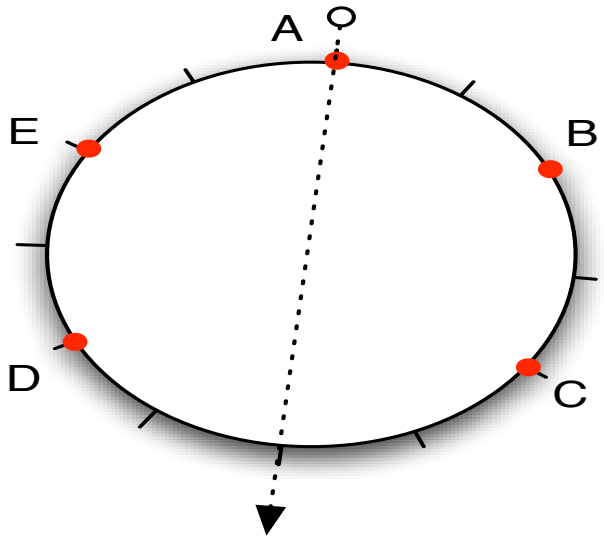
$$\{(3, 3, 3, 2, 1), (1, 2, 3, 3, 3)\}$$

Η συνάντηση περιττού αριθμού από ρομπότ

Algorithm Odd-Gathering

if the configuration is periodic **then** gathering is impossible
else
    if the configuration has a single multiplicity
    then Single-Multiplicity-Gathering
    else
        if the configuration is rigid **then** Rigid-Gathering
        else
            if R is axial **then** move (to any of the adjacent nodes)

Η συνάντηση περιττού αριθμού από ρομπότ



Ο σχηματισμός από το A
$(2, 2, 4, 2, 2)$

Ο σχηματισμός από το B
$(2, 4, 2, 2, 2)$

Ο σχηματισμός από το C
$(4, 2, 2, 2, 2)$

Η εικόνα που έχει το ρομπότ A

$$\{(2, 2, 4, 2, 2), (2, 2, 4, 2, 2)\}$$

Η συνάντηση περιττού αριθμού από ρομπότ

Έστω C ένας συμμετρικός σχηματισμός από έναν περιττό αριθμό από ρομπότ, χωρίς πολλαπλότητες. Έστω C' ο σχηματισμός που προκύπτει από τον C μετά την μετακίνηση του ρομπότ που βρίσκεται πάνω στον άξονα συμμετρίας, σε κάποιον από τους γειτονικούς κόμβους.

Λήμμα

Εάν ο σχηματισμός C' δεν έχει πολλαπλότητες τότε δεν είναι περιοδικός.
Απόδειξη: Αφού ο σχηματισμός C είναι συμμετρικός, ο C' 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' ήταν περιοδικός και έστω d η περίοδός του με μήκος p. Τότε $d_1 = a+1$, $d_p = a-1$. Αλλά από τη συμμετρία του C προκύπτει $d_p = d_1$. Άτοπο.

Η συνάντηση περιττού αριθμού από ρομπότ

Λήμμα

Μετά από το πολύ k κινήσεις, ο σχηματισμός C' που προκύπτει είτε έχει ακριβώς μία πολλαπλότητα είτε είναι *rigid*.
Απόδειξη: Έστω x η απόσταση μεταξύ του ρομπότ που βρίσκεται στον άξονα συμμετρίας και κάποιου γειτονικού ρομπότ.

1. Υπάρχει ακριβώς μια τιμή που εμφανίζεται στον C περιττό αριθμό από φορές.
2. Στον C' αυτή η τιμή είναι είτε $x-1$ είτε $x+1$.
3. Ο συνολικός αριθμός από εμφανίσεις στον C' ακεραίων διαφορετικής ομοτιμίας από αυτήν την τιμή είναι αυστηρά μικρότερος από ότι στον C.

Θεώρημα

Η συνάντηση ενός περιττού αριθμού από ρομπότ είναι εφικτή αν και μόνο αν ο αρχικός σχηματισμός δεν είναι περιοδικός.

Συμπεράσματα

Για έναν περιττό αριθμό από ρομπότ

- η συνάντηση είναι εφικτή αν και μόνο αν στον αρχικό σχηματισμό, τα ρομπότ μπορούν να εκλέξουν έναν κόμβο στον οποίο βρίσκεται κάποιο ρομπότ.

Για έναν άρτιο αριθμό από ρομπότ

- η συνάντηση είναι αδύνατη όταν

  είτε ο αριθμός των ρομπότ είναι 2, ή
  ο αρχικός σχηματισμός είναι περιοδικός, ή
  υπάρχει μια συμμετρία ακμής -ακμής.
- η συνάντηση είναι εφικτή για όλους τους *rigid* σχηματισμούς.

Ανοικτά ερωτήματα

- Είναι η συνάντηση εφικτή για συμμετρικούς, μή-περιοδικούς σχηματισμούς από έναν άρτιο αριθμό από ρομπότ με κάποια συμμετρία κόμβου-ακμής;
- Άλλες ικανότητες αντί για την ανίχνευση πολλαπλότητας (π.χ. περιορισμένη μνήμη);
- Άλλες τοπολογίες δικτύου;

Εικασίες για την τοπολογία του δακτυλίου

- Για έναν άρτιο αριθμό από περισσότερα από 2 ρομπότ, η συνάντηση είναι εφικτή αν και μόνο αν ο αρχικός σχηματισμός δεν είναι περιοδικός και δεν έχει συμμετρία ακμής -ακμής.
- Για έναν οποιονδήποτε αριθμό από περισσότερα από 2 ρομπότ, η συνάντηση είναι εφικτή αν και μόνο αν στον αρχικό σχηματισμό τα ρομπότ μπορούν να εκλέξουν έναν κόμβο (στον οποίο δεν βρίσκεται κατ΄ ανάγκη ρομπότ).

Rendezvous

Το πρόβλημα της συνάντησης (***Rendezvous***)

Πώς πρέπει να κινηθούν δύο mobile agents έτσι ώστε να συναντηθούν (rendezvous) σε κάποιο κόμβο του δικτύου;

Ενδιαφέρουσες ερωτήσεις

Ποιο είναι το πιο `αδύναμο' σενάριο για το οποίο η συνάντηση είναι δυνατή;

- τι ικανότητες πρέπει να έχουν οι πράκτορες (π.χ. να μπορούν να αφήνουν μηνύματα στους κόμβους, να αφήνουν σημάδια (*tokens*) στους κόμβους)
- μνήμη (π.χ. για να μετρούν), γνώση (π.χ. τον αριθμό των κόμβων του δικτύου)

Rendezvous

Διάφορα μοντέλα

[Yu, Yung, 1996]: Μη-επιλύσιμο σε γενικά γραφήματα αν οι πράκτορες χρησιμοποιούν τον ίδιο ντετερμινιστικό αλγόριθμο.

- [Baston, Gal, 2001], [Dessmark, Fraigniaud, Pelc, 2003]: Πιθανοτικοί αλγόριθμοι ή διαφορετικοί ντετερμινιστικοί αλγόριθμοι και μνήμη.
- [Barriere et al, 2003], [Dobrev et al, 2004]: Ανώνυμοι πράκτορες που αφήνουν μηνύματα στους κόμβους.
- [Kranakis et al, 2003], [Sawchuk, 2004], [Gasieniec et al, 2006]: Ανώνυμοι πράκτορες που αφήνουν *tokens* στους κόμβους ενός δακτύλιου.

Το μοντέλο (I)

Δύο ολόιδιοι κινητοί πράκτορες

1. είναι αρχικά τοποθετημένοι σε ένα ανώνυμο, σύγχρονο και προσανατολισμένο *nxm* torus .
2. Εκτελούν τον ίδιο ντετερμινιστικό αλγόριθμο και έχουν στη διάθεσή τους πανομοιότυπα **tokens**.
3. δεν γνωρίζουν καμμία παράμετρο του δικτύου.
4. γνωρίζουν τον αριθμό των tokens που έχουν.

Το μοντέλο (II)

Μια χρονική στιγμή ο πράκτορας βρίσκεται σε έναν κόμβο v του δικτύου και μπορεί

- να παραμείνει εκεί ή να κινηθεί σε κάποιο γειτονικό κόμβο,
- να δει τον αριθμό των tokens στο v,
- να αφήσει (ή να πάρει) ένα ή περισσότερα tokens στο (από το) v.

Άλλες ικανότητες του πράκτορα

Ανάλογα με την μνήμη που έχει, ο πράκτορας μπορεί να μετρήσει τους κόμβους που επισκέπτεται κλπ.

Rendezvous

Μπορούμε να σχεδιάσουμε

έναν ντετερμινιστικό αλγόριθμο που να οδηγεί τους δύο πράκτορες σε συνάντηση μέσα σε πεπερασμένο χρόνο; (ανεξάρτητα από τις αρχικές τους θέσεις ή το μέγεθος του δικτύου).

Απαιτήσεις για συνάντηση στη χειρότερη περίπτωση

- Ποιός είναι ο ελάχιστος χρόνος (ή μικρότερη διαδρομή) που απαιτείται;
- Ποια είναι η ελάχιστη μνήμη που απαιτείται;
- Ποιός είναι ο ελάχιστος αριθμός από tokens;

Trade-offs

Μεταξύ της μνήμης και του αριθμού από tokens.

Rendezvous With Detection vs Rendezvous Without Detection.

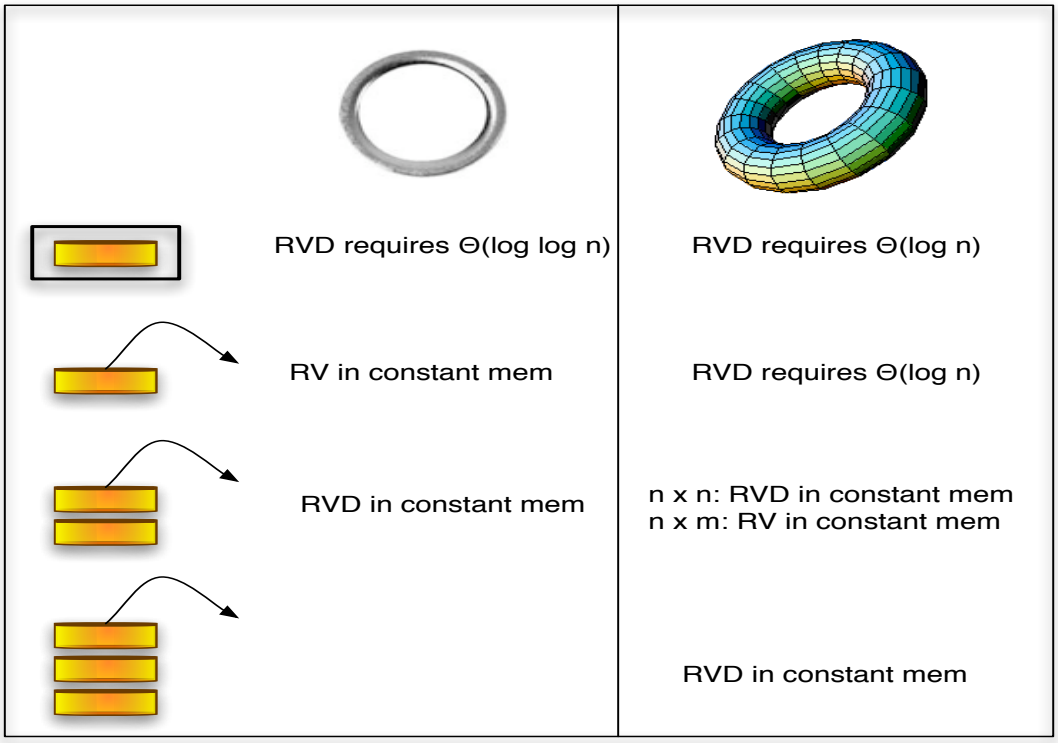
Λήμμα

Έστω $(0, 0)$ η αρχική θέση του πράκτορα A στο torus. Εάν η αρχική θέση του πράκτορα B είναι είτε $(n/2, 0)$ ή $(0, m/2)$ ή $(n/2, m/2)$ τότε είναι αδύνατη η συνάντηση (ανεξάρτητα από τον αριθμό των tokens ή της μνήμης που έχουν οι πράκτορες).

Ορισμοί

- **Rendezvous Without Detection (RV)**: οι πράκτορες συναντιούνται αν η αρχική τους απόσταση είναι διαφορετική από $(n/2, 0)$, $(0, m/2)$ και $(n/2, m/2)$.
- **Rendezvous With Detection (RVD)**: αλλιώς σταματούν και αναγνωρίζουν ότι η συνάντηση είναι αδύνατη.

Ring vs Torus



Κάτω φράγματα μνήμης για συνάντηση

Δύο πράκτορες σε ένα $n \times n$ **torus:**

- Χρειάζονται $\Omega(\log n)$ μνήμη ο καθένας αν έχουν στη διάθεσή τους έναν σταθερό αριθμό από ακίνητα tokens.
- Χρειάζονται $\Omega(\log n)$ μνήμη ο καθένας αν έχουν στη διάθεσή τους ένα κινητό token.

Αλγόριθμοι συνάντησης

Με ένα ακίνητο **token** και $O(\log n + \log m)$ μνήμη

RVD σε $O(n \cdot m)$ βήματα.

Με δύο κινητά **tokens** και σταθερή μνήμη

- **RVD** σε $n \times n$ torus μετά από $O(n^2)$ βήματα,
- **RV** σε $n \times m$ torus μετά από $O(n^4 + m^4)$ βήματα.

Με τρία κινητά **tokens** και σταθερή μνήμη

RVD σε $O(n^2 + m^2)$ βήματα.

Ένας πράκτορας σε ένα $n \times n$ **Torus**

Λήμμα

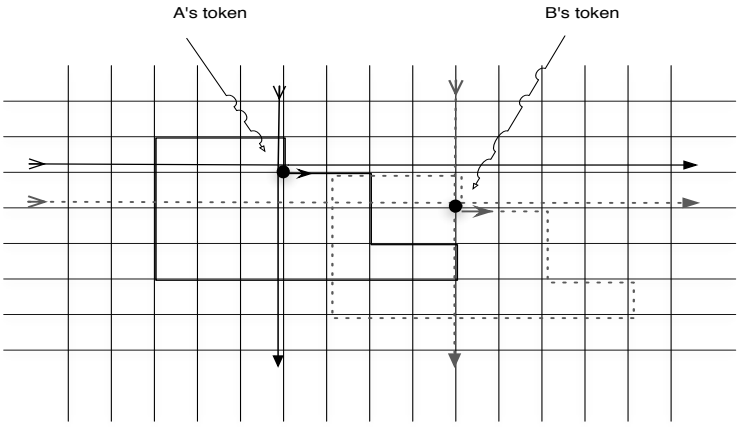
Ένας πράκτορας χωρίς **tokens** χρειάζεται τουλάχιστον $\Omega(\log n)$ μνήμη για να επισκεφτεί όλους τους κόμβους.

Ιδέα: Μετά από το πολύ n επαναλήψεις της πρώτης κατάστασης που επαναλαμβάνεται, ο πράκτορας δεν επισκέπτεται καινούριους κόμβους.

Λήμμα

Ένας πράκτορας που έχει στη διάθεσή του έναν σταθερό αριθμό από ακίνητα tokens χρειάζεται τουλάχιστον $\Omega(\log n)$ μνήμη για να επισκεφτεί όλους τους κόμβους.

Δύο πράκτορες με ένα ακίνητο **token** χρειάζονται τουλάχιστον $\Omega(\log n)$ μνήμη για να συναντηθούν.



Ιδέα: Μετά τον A, τοποθέτησε τον πράκτορα B έτσι ώστε:

- Ο πράκτορας B αφήνει το token σε κόμβο που δεν θα επισκεφτεί ποτέ ο πράκτορας A και
- Ο πράκτορας B δεν επισκέπτεται ποτέ τον κόμβο που βρίσκεται το token που έχει αφήσει ο A.

Σταθερός αριθμός από ακίνητα **tokens** – ένα κινητό **token.**

Λήμμα

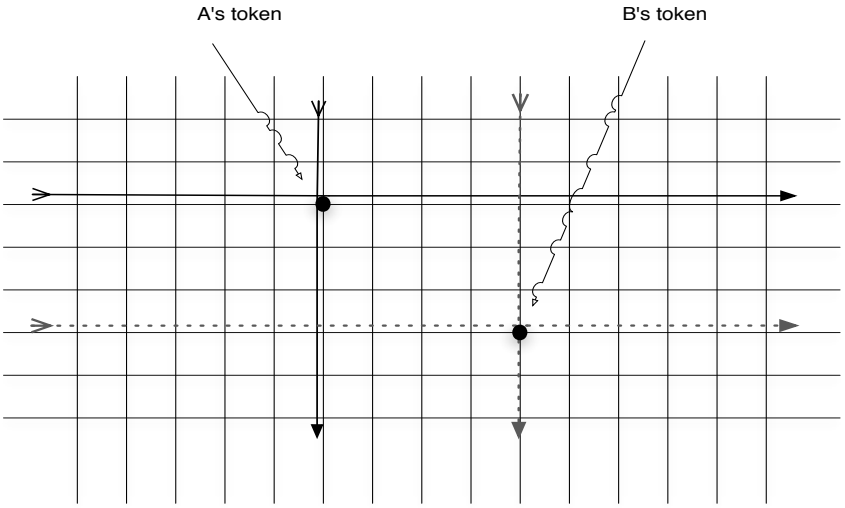
Δύο πράκτορες με ένα σταθερό αριθμό από ακίνητα tokens χρειάζονται τουλάχιστον $\Omega(\log n)$ μνήμη για ραντεβού.

Λήμμα

Δύο πράκτορες με ένα κινητό token ο καθένας χρειάζονται τουλάχιστον $\Omega(\log n)$ μνήμη για ραντεβού.

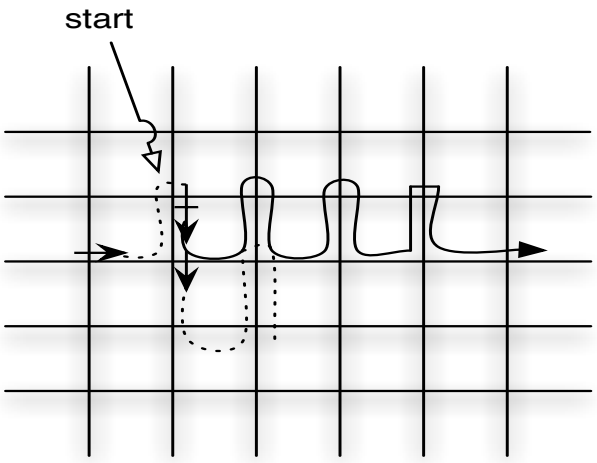
Ιδέα: τοποθέτησε τους πράκτορες έτσι ώστε σε οποιαδήποτε φάση που ξεκινά όταν οι πράκτορες μετακινούν τα tokens τους, και διαρκεί μέχρι να τα μετακινήσουν ξανά, οι πράκτορες δεν συναντούν ξένα tokens.

RVD σε $n \times n$ **Torus** με ένα ακίνητο **token** και $O(\log n)$ μνήμη σε $O(n^2)$ βήματα.



- Ίδιος δακτύλιος: μέτρα τους κόμβους οριζόντια και κάθετα.
- Διαφορετικός δακτύλιος: ψάξε έναν προς έναν τους οριζόντιους δακτύλιους.

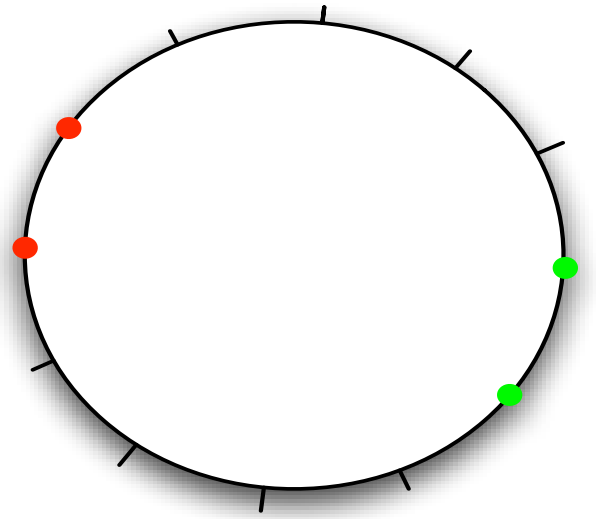
RVD σε $n \times n$ **Torus** με δύο κινητά **tokens** και σταθερή μνήμη σε $O(n^2)$ βήματα (I).



1. Οριζόντια ανίχνευση

Απόφαση για το εάν οι πράκτορες έχουν ξεκινήσει στον ίδιο ή σε διαφορετικούς δακτύλιους.

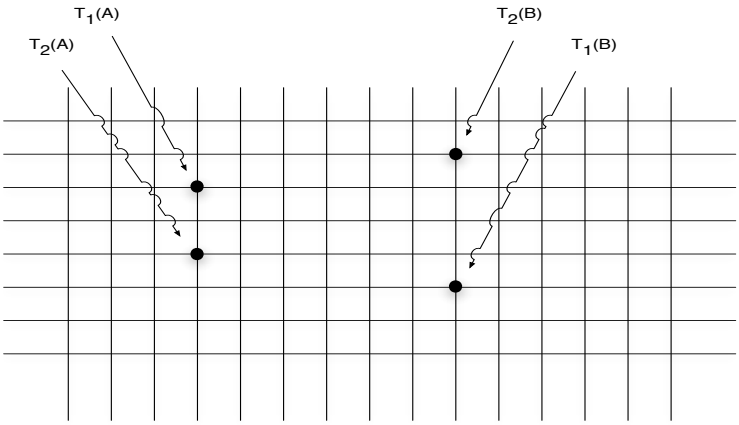
RVD σε $n \times n$ **Torus** με δύο κινητά **tokens** και σταθερή μνήμη σε $O(n^2)$ βήματα (II).



2. **RVD** σε δακτύλιο

Εάν οι πράκτορες έχουν ξεκινήσει στον ίδιο δακτύλιο τότε RVD.

RVD σε $n \times n$ **Torus** με δύο κινητά **tokens** και σταθερή μνήμη σε $O(n^2)$ βήματα (III).

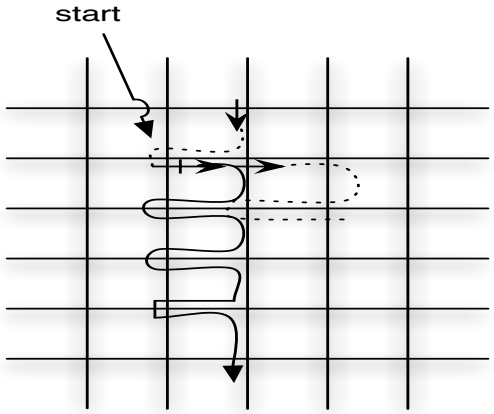


3. Σημάδεψε ένα μονοπάτι

Σημάδεψε ένα μονοπάτι με τα tokens και προσπάθησε να συναντήσεις τον άλλον.

Αν δεν πετύχεις τότε πρέπει να ισχύει $d_y = n/2$.

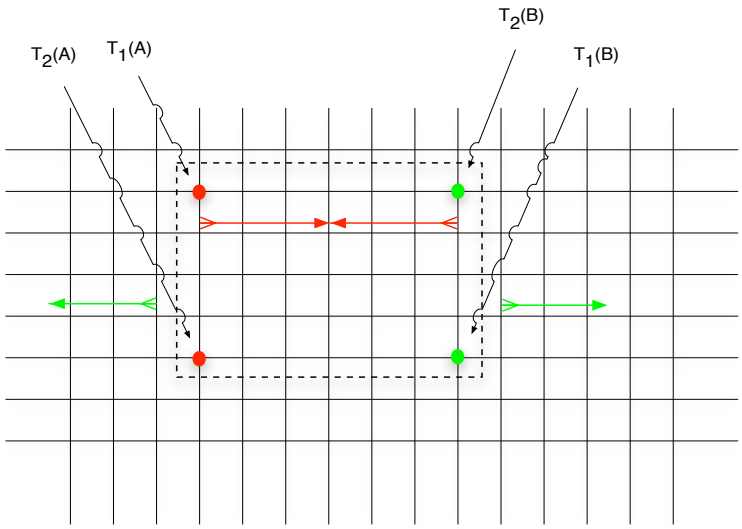
RVD σε $n \times n$ **Torus** με δύο κινητά **tokens** και σταθερή μνήμη σε $O(n^2)$ βήματα (**IV**).



4. Κάθετη ανίχνευση

Ανίχνευσε κάθετα και προσπάθησε να συναντήσεις τον άλλον.
Αν δεν πετύχεις τότε πρέπει επίσης να ισχύει $d_x = n/2$.

RV σε $n \times m$ **Torus** με δύο κινητά **tokens** και σταθερή μνήμη σε $O(n^4 + m^4)$ βήματα.



Διαφορετικός δακτύλιος;

1. Σημάδεψε ένα παραλληλόγραμμο,
2. Προσπάθησε να το μικρύνεις

Μπορεί να μετασχηματιστεί σε **RVD** αλγόριθμο όταν:

$\frac{n-1}{10} \leq m \leq 2n + 17$

RVD σε $n \times m$ **Torus** με τρία κινητά **tokens** και σταθερή μνήμη σε $O(n^2 + m^2)$ βήματα.

Διαφορετικός δακτύλιος;

1. Σημάδεψε ένα παραλληλόγραμμο,
2. Μετακίνησε το τρίτο token πάνω στο παραλληλόγραμμο (όπως ο RVD αλγόριθμος για δακτύλιο με δύο tokens)

Συμπεράσματα - Ανοικτά ερωτήματα

Συμπεράσματα στο **torus**

- Οποιοσδήποτε σταθερός αριθμός από ακίνητα tokens είναι λιγότερο δυνατός από δύο κινητά tokens.
- Η ιεραρχία καταρρέει στα τρία tokens.

Ανοικτά ερωτήματα

- Τρία κινητά tokens έχουν αυστηρά περισσότερη δύναμη από δύο; (Με άλλα λόγια, είναι RVD δυνατόν με δύο κινητά tokens και σταθερή μνήμη σε ένα $n \times m$ torus;)
- Ασύγχρονο torus χωρίς προσανατολισμό;

Το πρόβλημα της μαύρης τρύπας.

Μαύρη τρύπα:

Μία εξαιρετικά καταστρεπτική διαδικασία η οποία βρίσκεται σε κάποιον κόμβο του δικτύου και είναι ικανή να καταστρέψει όλους τους πράκτορες που επισκέπτονται αυτόν τον κόμβο χωρίς να αφήσει κανένα ίχνος

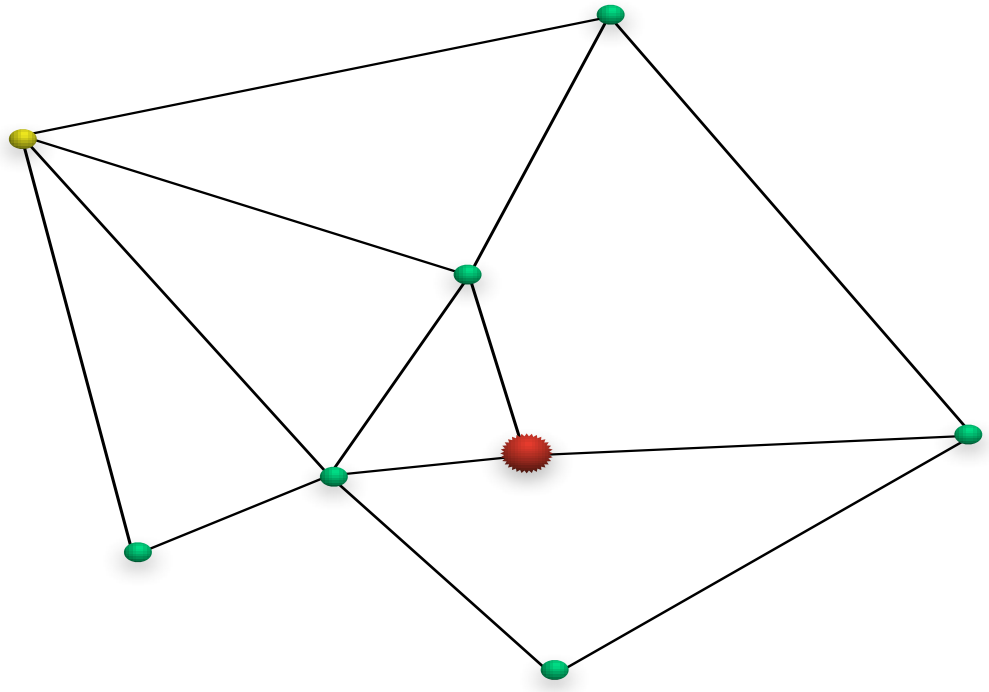
- Δίνεται ένα γράφημα στο οποίο υπάρχει το πολύ μία μαύρη τρύπα,
- Μία ομάδα από πράκτορες βρίσκεται σε κάποιον κόμβο u_s του γραφήματος.

Το πρόβλημα της μαύρης τρύπας.

Το πρόβλημα της μαύρης τρύπας.

Θέλουμε να σχεδιάσουμε έναν ντετερμινιστικό αλγόριθμο που μετά από πεπερασμένο χρόνο θα οδηγήσει τους πράκτορες στην ανακάλυψη της μαύρης τρύπας. Δηλαδή μετά από πεπερασμένα βήματα πρέπει τουλάχιστον ένας από τους πράκτορες να επιστρέψει στον κόμβο u_s και να αναφέρει τον κόμβο στον οποίον υπάρχει η μαύρη τρύπα (ή να αναφέρει ότι το δίκτυο είναι καθαρό).

Το πρόβλημα της μαύρης τρύπας.



Ένα μοντέλο.

Ασύγχρονο δίκτυο.

Οι πράκτορες μπορούν να γράψουν μηνύματα στους κόμβους που επισκέπτονται.

- Πόσοι πράκτορες χρειάζονται για να εντοπίσουν τη μαύρη τρύπα;
- Τί γνώση χρειάζεται να έχουν οι πράκτορες;
- Ποιός είναι ο χρόνος που απαιτείται;
- Πόσοι πράκτορες θα χαθούν μέσα στην μαύρη τρύπα;

Μερικά αποτελέσματα στο προηγούμενο μοντέλο.

Μερικά αποτελέσματα στο προηγούμενο μοντέλο.

- Το δίκτυο πρέπει να είναι 2−connected
- Είναι αδύνατον να αποφασιστεί αν υπάρχει μια μαύρη τρύπα στο δίκτυο.
- Εάν δεν υπάρχει καμμία γνώση για την τοπολογία του δικτύου εκτός από τα n, Δ τότε χρειάζονται $\Delta + 1$ πράκτορες που ξεκινούν από τον ίδιο κόμβο. Ανακαλύπτουν την μαύρη τρύπα μετά από $\Theta(n^2)$ βήματα.
- Εάν έχουν αίσθηση της κατεύθυνσης τότε αρκούν δύο πράκτορες που ξεκινούν από τον ίδιο κόμβο. Ανακαλύπτουν την μαύρη τρύπα μετά από $\Theta(n^2)$ βήματα.

Μερικά αποτελέσματα στο προηγούμενο μοντέλο.

Μερικά αποτελέσματα στο προηγούμενο μοντέλο.

- Εάν έχουν γνώση της τοπολογίας του δικτύου τότε αρκούν δύο πράκτορες που ξεκινούν από τον ίδιο κόμβο. Ανακαλύπτουν την μαύρη τρύπα μετά από $\Theta(n \log n)$ βήματα.
- Εάν η τοπολογία είναι προσανατολισμένος δακτύλιος αρκούν δύο πράκτορες ακόμη και αν δεν ξεκινούν από τον ίδιο κόμβο.
- Εάν η τοπολογία είναι δακτύλιος χωρίς προσανατολισμό και οι πράκτορες δεν ξεκινούν από τον ίδιο κόμβο, τότε χρειάζονται τρεις και αρκούν.

Το μοντέλο που θα εξετάσουμε.

Το μοντέλο που θα εξετάσουμε.

Δύο πράκτορες με διαφορετικές ταυτότητες ξεκινούν από τον ίδιο κόμβο. Έχουν έναν χάρτη του δικτύου. Το δίκτυο είναι συγχρονισμένο. Υπάρχει το πολύ μία μαύρη τρύπα. Οι πράκτορες έχουν αρκετή μνήμη και μπορούν να επικοινωνήσουν μόνο όταν συναντιούνται σε κάποιον κόμβο (δεν μπορούν δηλαδή να στείλουν μηνύματα ο ένας στον άλλον ή να αφήσουν μηνύματα σε κόμβους).

Παρατηρήσεις

- Χρειάζονται οπωσδήποτε δύο πράκτορες για να ανακαλύψουν την μαύρη τρύπα.
- Οι πράκτορες μπορούν να αποφασίσουν αν υπάρχει μαύρη τρύπα ή όχι στο δίκτυο.

Ιδιότητες του αλγόριθμου.

Υποθέτοντας

ότι η διάσχιση μιας ακμής διαρκεί μια χρονική μονάδα ορίζουμε σαν κόστος ενός Black Hole Search (BHS) scheme τον χρόνο που χρειάζεται μέχρι την αναφορά της τοποθεσίας της μαύρης τρύπας (ή την αναφορά ότι δεν υπάρχει μαύρη τρύπα) για το χειρότερο δυνατό σενάριο (αρχικός κόμβος, θέση της μαύρης τρύπας).

Ερώτηση:

Πόσο γρήγορα μπορούν να ανακαλύψουν οι πράκτορες την μαύρη τρύπα (ή να αποφασίσουν ότι δεν υπάρχει μαύρη τρύπα στο δίκτυο) ανεξάρτητα από τον κόμβο που ξεκινούν;

Να σχεδιαστεί ένας ντετερμινιστικός αλγόριθμος ο οποίος:

Δεδομένου ενός γραφήματος και ενός αρχικού κόμβου, επιστρέφει ένα BHS scheme ελάχιστου κόστους.

Αποτελέσματα.

Αποτελέσματα.

1. 5/3 προσεγγιστικός αλγόριθμος για γενικά δέντρα,
2. βέλτιστοι αλγόριθμοι για ειδικές κατηγορίες δέντρων,
3. NP-hard σε γενικά γραφήματα,
4. $\frac{7}{2} \rightarrow 3\frac{3}{8}$ προσεγγιστικός αλγόριθμος για γενικά γραφήματα,
5. APX-hard σε γενικά γραφήματα,
6. εάν υπάρχει ένα υποσύνολο κόμβων για τους οποίους είναι γνωστό ότι είναι ασφαλείς τότε υπάρχει προσεγγιστικός αλγόριθμος με παράγοντα 6.

Το μοντέλο.

BHS-Scheme

Ένα σχήμα αναζήτησης μαύρης τρύπας για είσοδο (G, s) είναι ένα ζεύγος από ακολουθίες κινήσεων για κάθε έναν από τους δύο πράκτορες με τις εξής ιδιότητες:

- Η διάσχιση μιας ακμής διαρκεί μια χρονική μονάδα,
- Στο τέλος του σχήματος υπάρχει τουλάχιστον ένας πράκτορας ο οποίος γνωρίζει ακριβώς την θέση της μαύρης τρύπας,
- Όλοι οι πράκτορες που έχουν επιζήσει πρέπει να επιστρέψουν στον κόμβο s.

Ένας κόμβος θεωρείται εξερευνημένος όταν όλοι οι επιζώντες πράκτορες ξέρουν αν είναι ασφαλής ή όχι.
Η περιοχή των κόμβων που είναι εξερευνημένοι μεγαλώνει στα σημεία συνάντησης.
Οι ακολουθίες κινήσεων μεταξύ των σημείων συνάντησης καλούνται φάσεις.

Βασικές ιδιότητες και τεχνικές αναζήτησης

Βασικές ιδιότητες

- Έναν ανεξερεύνητο κόμβο δεν μπορούν να τον επισκεφτούν και οι δύο πράκτορες,
- Κατά τη διάρκεια μιας φάσης, ένας πράκτορας μπορεί να επισκεφτεί το πολύ έναν ανεξερεύνητο κόμβο,
- Στο τέλος μιας φάσης η εξερευνημένη περιοχή αυξάνεται κατά έναν ή δύο κόμβους.

Τεχνικές αναζήτησης

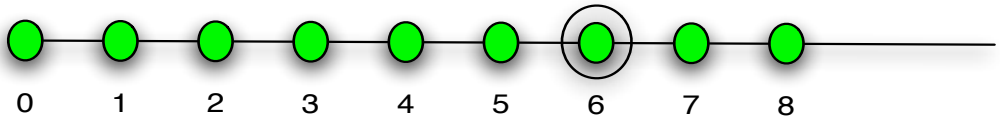
- **Probe:** ένας από τους πράκτορες επισκέπτεται έναν ανεξερεύνητο κόμβο και επιστρέφει στην εξερευνημένη περιοχή για να συναντήσει τον άλλον πράκτορα.
- **Split:** οι δύο πράκτορες επισκέπτονται έναν ανεξερεύνητο κόμβο ο καθένας και επιστρέφουν στην εξερευνημένη περιοχή για να συναντηθούν.

Η περίπτωση της γραμμής

Η περίπτωση της γραμμής

Ένας βέλτιστος αλγόριθμος για αναζήτηση μαύρης τρύπας σε το πολύ $4n - 8$ κινήσεις.

- Split(s-1, s+1);
- Split(s-2, s+2);
- Walk(s-1);
- Walk-and-Probe(1);
- Split(0, s+3);
- Walk(s+2);
- Walk-and-Probe(n);

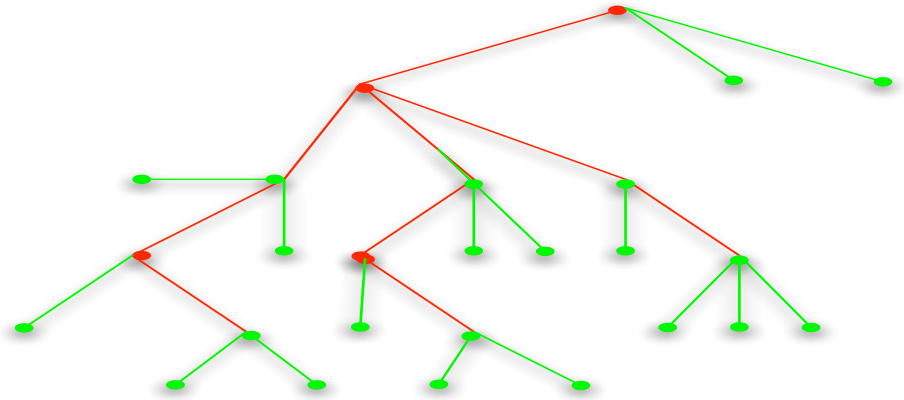


Η περίπτωση του θαμνώδους δέντρου

Η περίπτωση του θαμνώδους δέντρου

Ένας βέλτιστος αλγόριθμος για αναζήτηση μαύρης τρύπας χρησιμοποιώντας μόνο τη διαδικασία Split.

- κόκκινη ακμή: απαιτούνται τουλάχιστον 6 διασχίσεις
- πράσινη ακμή: απαιτούνται τουλάχιστον 2 διασχίσεις

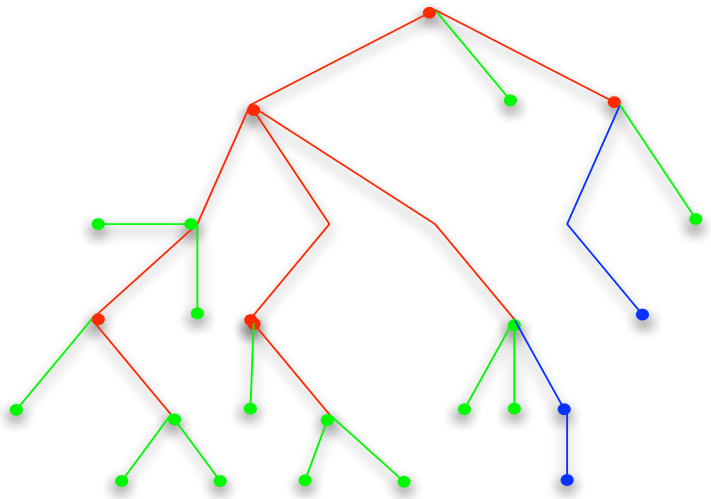


Η περίπτωση του γενικού δέντρου

Η περίπτωση του γενικού δέντρου

Ένας 5/3 προσεγγιστικός αλγόριθμος $(d + 4\beta + 2\rho\ (+1))$

- μπλε κλαδί: απαιτούνται τουλάχιστον 6 διασχίσεις



Η περίπτωση του γενικού γραφήματος

Το πρόβλημα του κύκλου **Hamilton**

- Είσοδος: δίνεται ένα επίπεδο γράφημα $G = (V, E)$, του οποίου κάθε κόμβος έχει βαθμό 3, και μια ακμή $(x, y) \in E$.
- Ερώτηση: υπάρχει κύκλος Hamilton στο γράφημα G που να περιέχει την ακμή (x, y);

Το πρόβλημα απόφασης της αναζήτησης μαύρης τρύπας

- Είσοδος: δίνεται ένα γράφημα $G' = (V', E')$, ένας αρχικός κόμβος $s \in V'$ και ένας θετικός ακέραιος X.
- Ερώτηση: υπάρχει ένα σχήμα αναζήτησης μαύρης τρύπας για το γράφημα G' με αρχικό κόμβο το s και κόστος το πολύ X;

Ανοιχτά προβλήματα

Ανοιχτά προβλήματα

- Είναι NP-hard το πρόβλημα σε γενικά δέντρα;
- Καλύτεροι προσεγγιστικοί αλγόριθμοι σε γραφήματα,
- Περισσότερες μαύρες τρύπες,
- Περισσότεροι πράκτορες.

Βιβλιογραφία

- N. Santoro, Design and Analysis of Distributed Algorithms, *Wiley 2007*.
- W. Jansen and T. Karygiannis, Mobile Agent Security, *Special Publication 800-19, National Institute of Standards and Technology*, USA, August 1999.
- R. Klasing, E. Markou and A. Pelc, Gathering Asynchronous Oblivious Mobile Robots in a Ring, *Theoretical Computer Science*, 390 (2008), pp. 27-39.
- E. Kranakis, D. Krizanc and E. Markou, Mobile Agent Rendezvous in a Synchronous Torus, *Proc. 7th Latin American Theoretical Informatics Symposium (LATIN' 06)*, March 2006, Valdivia, Chile, LNCS 3887, pp. 653-664.
- J. Czyzowicz, D. Kowalski, E. Markou and A. Pelc, Searching for a Black Hole in Synchronous Tree Networks, *Combinatorics, Probability & Computing*, 16 (4) (2007), pp. 595-619.