ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Άμεσοι Αλγόριθμοι

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σπυρίδωνος Ρ. Αντωνακόπουλου

Επιβλέπων: Ευστάθιος Κ. Ζάχος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2004

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Άμεσοι Αλγόριθμοι

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σπυρίδωνος Ρ. Αντωνακόπουλου

Επιβλέπων: Ευστάθιος Κ. Ζάχος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Ιουλίου 2004.

....	....	....
Ευστάθιος Ζάχος Καθηγητής Ε.Μ.Π.	Φώτω Αφράτη Καθηγήτρια Ε.Μ.Π.	Τιμολέων Σελλής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2004.

....

Σπυρίδων Ρ. Αντωνακόπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και
Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η παρούσα διπλωματική εργασία εστιάζει στην ερευνητική περιοχή των άμεσων (online) αλγορίθμων, δηλαδή εκείνων που καλούνται να επιλύσουν ένα πρόβλημα βελτιστοποίησης δίχως να έχουν εκ των προτέρων γνώση όλων των σχετικών δεδομένων.

Στο πρώτο κεφάλαιο ορίζονται οι βασικές έννοιες της ανταγωνιστικής ανάλυσης, η οποία αποτελεί το πλέον κοινά αποδεκτό κριτήριο αξιολόγησης της επίδοσης άμεσων αλγορίθμων. Ως παράδειγμα εφαρμογής των προηγουμένων, μελετώνται διεξοδικά δύο κλασικά online προβλήματα.

Το δεύτερο κεφάλαιο πραγματεύεται τεχνικές σχεδίασης αιτιοκρατικών άμεσων αλγορίθμων. Μάλιστα, έμφαση δίνεται στο σκεπτικό που υποκρύπτεται πίσω από καθεμία τεχνική, το οποίο συνήθως διατυπώνεται ως κάποια «αρχή». Επιπλέον, εξετάζονται οι κυριότερες μέθοδοι ανάλυσης online αλγορίθμων, οι οποίες έχουν χρησιμοποιηθεί ευρέως στη σχετική βιβλιογραφία.

Το τρίτο κεφάλαιο ασχολείται με randomized άμεσους αλγορίθμους, οι οποίοι παρουσιάζουν εξαιρετικό ενδιαφέρον διότι η αναμενόμενη επίδοσή τους ως προς την ανταγωνιστική ανάλυση είναι καλύτερη εκείνης των αντίστοιχων αιτιοκρατικών.

Τέλος, το τέταρτο κεφάλαιο εστιάζει στο διάσημο πρόβλημα των k-εξυπηρετητών. Συγκεκριμένα, παρατίθενται, αναλύονται και συγκρίνονται ορισμένοι από τους πιο γνωστούς αλγορίθμους που έχουν προταθεί για το πρόβλημα αυτό. Περιγράφεται επίσης η γενικότερη μέθοδος κατασκευής νέων τέτοιων αλγορίθμων μέσω αιτιοκρατικών και ιδίως πιθανοτικών εμφυτεύσεων μετρικών χώρων.

Λέξεις-κλειδιά: άμεσοι αλγόριθμοι, ανταγωνιστική ανάλυση, k-εξυπηρετητές, σελιδοποίηση, πιθανοτικές εμφυτεύσεις

Abstract

This diploma thesis focuses on the research area of online algorithms, i.e. those algorithms that are obliged to solve an optimization problem without having full *a priori* knowledge of the input data.

In the first chapter, we define the basic notions of competitive analysis, which is the most standard criterion for evaluating the performance of online algorithms. As an example, we apply the above in a thorough study of two classic online problems.

The second chapter deals with design techniques for deterministic online algorithms. Moreover, special emphasis is given to the rationale behind each technique, which is usually stated as a "principle." On top of that, we survey the most important methods for analyzing online algorithms, which have been used extensively in the relevant bibliography.

In the third chapter, we turn our attention to randomized online algorithms. The latter are of particular interest, because their expected performance under competitive analysis is better than that of corresponding deterministic algorithms.

Last but not least, the fourth chapter is devoted to the famous k-server problem. More specifically, we present, analyze and compare several of the well-known algorithms that have been proposed for this problem. In addition, we describe the general method for deriving new such algorithms via deterministic and probabilistic embeddings of metric spaces.

Keywords: online algorithms, competitive analysis, k-server, paging, probabilistic embeddings

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Η έννοια του online αλγορίθμου

Η μελέτη των αλγορίθμων συνδέεται κατά κανόνα με την Θεωρία Πολυπλοκότητας. Πιο συγκεκριμένα, αν ALG είναι κάποιος αλγόριθμος επιλύει το πρόβλημα Π, υποθέτουμε ότι δίνεται ένα οποιοδήποτε στιγμιότυπο I του Π και εξετάζουμε πόσους πόρους (χρόνος, χώρος κ.α.) απαιτεί ο ALG για να βρει τη λύση, συναρτήσει του μεγέθους της εισόδου, δηλαδή του στιγμιοτύπου. Έτσι μπορούμε να αποφανθούμε αν ο ALG είναι *αποδοτικός*, δηλαδή «καλός», ή το αντίθετο.

Εντούτοις, υπάρχουν πολλά πραγματικά προβλήματα στα οποία η είσοδος δεν είναι γνωστή στο σύνολό της τη στιγμή που ο αλγόριθμος αρχίζει την επίλυση, αλλά ενδεχομένως να αποκαλυφθεί σταδιακά στη συνέχεια. Τούτο συνήθως οφείλεται σε κάποιον από τους παρακάτω λόγους:

- Η είσοδος εξαρτάται από μελλοντικά γεγονότα, όπως είναι π.χ. οι αυριανές τιμές των χρηματιστηριακών δεικτών.
- Η είσοδος εξαρτάται από ορισμένες ενέργειες του ίδιου του αλγορίθμου. Λογου χάρη, καθώς ένα ρομπότ εξερευνά άγνωστους χώρους αποκτά νέες πληροφορίες, οι οποίες επηρεάζουν τις επόμενες κινήσεις του.

Σε αμφότερες τις περιπτώσεις διαπιστώνουμε ότι ο υπολογισμός είναι συνυφασμένος με τη χρονική εξέλιξη κάποιου συστήματος (στα παραδείγματά μας, του Χρηματιστηρίου ή του ρομπότ) το οποίο αντιδρά σε διαδοχικές μεταβολές του περιβάλλοντός του. Βέβαια, στην πρώτη περίπτωση το εν λόγω χαρακτηριστικό αποτελεί εγγενή ιδιότητα του προβλήματος, ενώ στη δεύτερη προκύπτει ως επακόλουθο της διαδικασίας υπολογισμού καθαυτής.

Το κρίσιμο ζήτημα, λοιπόν, είναι ότι ο αλγόριθμος καλείται να λάβει αποφάσεις υπό καθεστώς αβεβαιότητας, αφού γνωρίζει μόνο τμήμα της εισόδου, τη στιγμή που αγνοεί παντελώς – ή έστω σε μεγάλο βαθμό – το υπόλοιπο. Επομένως, οι αποφάσεις αυτές γενικά δεν θα είναι οι καλύτερες δυνατές, οπότε το μόνο στο οποίο μπορούμε να ελπίζουμε είναι να περιοριστεί όσο γίνεται ο τελικός αντίκτυπός τους στην επίδοση του αλγορίθμου.

Με τα παραπάνω επιχειρήσαμε να αποδώσουμε διαισθητικά τί συνιστά online πρόβλημα, υπολογισμό και αλγόριθμο, αντίστοιχα. Η περιγραφή δεν είναι εξαντλητική και οπωσδήποτε όχι απόλυτα συγκεκριμένη. Από την άλλη μεριά, όμως, μέχρι τώρα δεν έχουν διατυπωθεί τυπικοί ορισμοί για τους παραπάνω όρους, κυρίως διότι καλύπτουν έννοιες με πλήθος διαφορετικών παραλλαγών που δύσκολα κωδικοποιούνται αυστηρά.

Οι όροι offline πρόβλημα, υπολογισμός και αλγόριθμος, μπορούν να θεωρηθούν ως οι συμπληρωματικοί των προηγουμένων. Με άλλα λόγια, χρησιμοποιούνται σε περιπτώσεις που η είσοδος του προβλήματος είναι εξ ολοκλήρου καθορισμένη και γνωστή ήδη πριν ζητηθεί η επίλυσή του. Στα πλαίσια της Θεωρίας Πολυπλοκότητας, τούτο συνήθως αποτελεί σιωπηρή υπόθεση και επομένως ο προσδιορισμός "offline" καθίσταται περιττός. Δεν συμβαίνει όμως το ίδιο στην παρούσα εργασία, όπου μάλιστα είναι επιθυμητό να δοθεί έμφαση στην αντιπαραβολή online και offline εννοιών.

Σημείωση. Στην ελληνική βιβλιογραφία προτείνεται το επίθετο *άμεσος* ως εύστοχη μετάφραση του "online". Δυστυχώς, δεν έχει γίνει το ίδιο με τη λέξη "offline", οπότε προτιμήθηκε να γίνει χρήση των αγγλικών όρων σε όλη την έκταση της εργασίας.

1.1.1 Ανταγωνιστική ανάλυση

Ένα εύλογο ερώτημα που τίθεται είναι πώς θα αξιολογήσουμε την επίδοση ενός online αλγορίθμου. Στο εξής λοιπόν θα ασχοληθούμε αποκλειστικά με προβλήματα βελτιστοποίησης, τα οποία προσφέρουν ένα εγγενές μέτρο σύγκρισης της «ποιότητας» των λύσεων που παράγει κάθε αλγόριθμος. Τα προβλήματα αυτά χωρίζονται αδρομερώς σε δύο κατηγορίες: στα μεν στόχος είναι η ελαχιστοποίηση του *κόστους* και στα δε η μεγιστοποίηση του *κέρδους*, όπου τα μεγέθη αυτά ορίζονται αναλόγως του εκάστοτε προβλήματος.

Η ανάλυση της χειρότερης περίπτωσης, που αποτελεί κανόνα στη Θεωρία Πολυπλοκότητας, εν προκειμένω φαίνεται ακατάλληλη, αφού οποιαδήποτε μη τετριμμένη απόφαση που λαμβάνεται με μερική γνώση των δεδομένων εισόδου μπορεί να αποδειχθεί υπο-βέλτιστη για κάποιο στιγμιότυπο του προβλήματος. Συνεπώς, από αυτή την άποψη όλοι οι online αλγόριθμοι για το ίδιο πρόβλημα χαρακτηρίζονται εξίσου «κακοί». Μια άλλη προσέγγιση, που χρησιμοποιείται ευρέως στην πράξη, είναι αυτή της αναμενόμενης (μέσης) επίδοσης του αλγορίθμου για μια πιθανοτική κατανομή επί του συνόλου των στιγμιοτύπων.

Βέβαια, για να θεωρήσουμε μια κατάλληλη τέτοια κατανομή πρέπει να έχουμε αρκετές πληροφορίες σχετικά με τις εισόδους που θα κληθεί να επεξεργαστεί ο αλγόριθμος.

Τί γίνεται, όμως, αν θέλουμε να αποφύγουμε – ή δεν μπορούμε να κάνουμε – οποιεσδήποτε τέτοιες υποθέσεις; Η μέθοδος που έχει προταθεί είναι να συγκρίνουμε την επίδοση του online αλγορίθμου σε σύγκριση με τη βέλτιστη λύση για το συγκεκριμένο στιγμιότυπο. Γνωρίζουμε ότι τέτοια λύση βρίσκει ο καλύτερος δυνατός offline αλγόριθμος για το ίδιο πρόβλημα, άρα ο τελευταίος θα χρησιμοποιηθεί ως μέτρο αναφοράς. Όπως θα δούμε στη συνέχεια της παρούσας εργασίας, με τον τρόπο αυτό αναδεικνύεται η ποιότητα ενός online αλγορίθμου καλύτερα απ' ότι με την ανάλυση χειρότερης περίπτωσης.

Τα πρώτα δείγματα εφαρμογής της εν λόγω ιδέας εμφανίζονται ήδη πριν 40 περίπου χρόνια σε εργασία του Graham [Gra66]. Όμως, καθοριστική ώθηση στην μελέτη των online αλγορίθμων έδωσαν δύο μεταγενέστερες δημοσιεύσεις των Sleator και Tarjan [ST85a, ST85b], εκ των οποίων η πρώτη αναφέρεται στα προβλήματα ενημέρωσης λίστας και σελιδοποίησης μνήμης (βλέπε Ενότητα 1.3), ενώ η δεύτερη σε εξαρθρωμένα δένδρα (splay trees). Λίγο αργότερα, οι Karlin, Manasse, Rudolph και Sleator [KarlMRS88] εισήγαγαν και καθιέρωσαν τον όρο *ανταγωνιστική ανάλυση* (*competitive analysis*) για να περιγράφουν την προαναφερθείσα μέθοδο αξιολόγησης αλγορίθμων.

Οφείλουμε εδώ να τονίσουμε ότι μολονότι η ανταγωνιστική ανάλυση συνδέεται άρρηκτα με τα online προβλήματα, βρίσκει εφαρμογή και σε άλλες παρεμφερείς περιοχές, όπως είναι οι κατανεμημένοι αλγόριθμοι. Υπενθυμίζουμε ότι οι τελευταίοι εκτελούνται σε ένα σύνολο από υπολογιστικές μονάδες, συνδεδεμένες σε δίκτυο, με τα δεδομένα του προβλήματος να βρίσκονται διασκορπισμένα σε αυτές. Παρ' όλο που η επικοινωνία μεταξύ των μονάδων είναι εφικτή μέσω του δικτύου, συχνά η πλήρης ανταλλαγή πληροφοριών κρίνεται ασύμφορη ή υπερβολικά χρονοβόρα. Ως εκ τούτου, σε πολλές περιπτώσεις κατανεμημένοι αλγόριθμοι υποχρεούνται να δώσουν λύσεις έχοντας μόνο μια μερική εικόνα της κατάστασης του όλου συστήματος και άρα είναι υποψήφιοι προς ανταγωνιστική ανάλυση.

Κατόπιν των ανωτέρω, μπορούμε να γίνουμε πιο συγκεκριμένοι δίνοντας τους πρώτους ουσιώδεις ορισμούς, με βάση τους οποίους θα προχωρήσουμε στη μελέτη των online αλγορίθμων.

Ορισμός 1.1. *Ένας online αλγόριθμος* ALG *που επιλύει το πρόβλημα* Π *θα λέγεται* c-ανταγωνιστικός *αν για κάθε είσοδο* I *ισχύει:*

$$\text{κόστος}_{\text{ALG}}(I) \leq c \cdot \text{κόστος}_{\text{OPT}}(I) + c_0$$

$$\left[\text{ ή } \quad \text{κέρδος}_{\text{OPT}}(I) \leq c \cdot \text{κέρδος}_{\text{ALG}}(I) + c_0 \right],$$

όπου OPT *ο βέλτιστος offline αλγόριθμος για το* Π *και* c_0 *μη αρνητική σταθερά.*

Προφανώς θα αληθεύει ότι $c \geq 1$, ανεξαρτήτως του αν το πρόβλημα αφορά ελαχιστοποίηση κόστους ή μεγιστοποίηση κέρδους.

Ορισμός 1.2. *Ο* λόγος ανταγωνιστικότητας (competitive ratio) c_{ALG} *του online αλγορίθμου* ALG *προκύπτει ως:*

$$c_{\text{ALG}} \triangleq \inf \{c \mid \text{ο ALG είναι } c\text{-ανταγωνιστικός}\} \ .$$

Παρατηρήσεις

1. Οι παραπάνω ορισμοί μοιάζουν πολύ με εκείνους που χρησιμοποιούνται στη μελέτη προσεγγιστικών αλγορίθμων για NP-δύσκολα προβλήματα βελτιστοποίησης (βλέπε π.χ. [GJ79]). Τούτο δεν είναι τυχαίο, αφού κατά μία έννοια οι online αλγόριθμοι μπορούν όντως να θεωρηθούν προσεγγιστικοί. Το χαρακτηριστικό όμως που τους καθιστά μη βέλτιστους δεν είναι οι περιορισμοί στην χρονική πολυπλοκότητα, αλλά η αβεβαιότητα ως προς τα δεδομένα της εισόδου.

2. Στην περίπτωση που ο ALG ικανοποιεί τον Ορισμό 1.1 με $c_0 = 0$, θα αποκαλείται *αυστηρά c-ανταγωνιστικός*.

3. Ο ALG θα θεωρείται *ανταγωνιστικός* αν και μόνο αν ο λόγος c_{ALG} είναι ανεξάρτητος του μεγέθους του στιγμιοτύπου. Αντίθετα, επιτρέπεται να αποτελεί συνάρτηση διαφόρων παραμέτρων του προβλήματος, οι οποίες υποτίθενται σταθερές και εξ αρχής γνωστές.

4. Τέλος, ο αλγόριθμος που επιτυγχάνει τον καλύτερο λόγο ανταγωνιστικότητας μεταξύ όλων των online αλγορίθμων οι οποίοι λύνουν το ίδιο πρόβλημα Π θα ονομάζεται *ισχυρά ανταγωνιστικός*.

1.2 Δύο κλασικά online προβλήματα και η ανάλυσή τους

Ακολούθως θα παρουσιάσουμε και θα εξετάσουμε διεξοδικά δύο από τα πλέον παλαιά και γνωστά προβλήματα της περιοχής των online αλγορίθμων. Παρά τη σχετική απλότητα της διατύπωσής τους, η πλήρης ανάλυσή τους δεν είναι τετριμμένη (ειδικά του δεύτερου εξ αυτών) και αξιοποιεί τις αφηρημένες έννοιες που ορίσαμε στην προηγούμενη ενότητα.

1.2.1 Το πρόβλημα ενοικίασης εξοπλισμού σκι

Ας υποθέσουμε ότι έχουμε εκδράμει σε κάποιο χιονοδρομικό κέντρο για να κάνουμε σκι. Η ενοικίαση του απαραίτητου εξοπλισμού στοιχίζει 30 ευρώ ανά ημέρα, ενώ η αγορά του $30k$ ευρώ, όπου $k \geq 2$. Το ερώτημα είναι, φυσικά: ποια επιλογή μας συμφέρει; Αν γνωρίζαμε εκ των προτέρων πόσες ημέρες θα κάνουμε σκι, τότε η λύση θα ήταν προφανής. Εντούτοις, διάφοροι αστάθμητοι παράγοντες (λ.χ. καιρός) δεν μας επιτρέπουν να το προβλέψουμε αυτό εξαρχής. Όντας λίγο απαισιόδοξοι, θεωρούμε ότι κάθε πρωί μπορούμε με βεβαιότητα να ξέρουμε αν θα κάνουμε ή όχι σκι τη δεδομένη μέρα, αλλά για τις επόμενες δεν είμαστε καθόλου σίγουροι. Τοιουτοτρόπως, καλούμαστε να λύσουμε ένα online πρόβλημα.

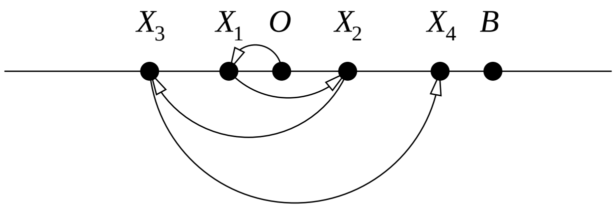
Μια σχετικά απλή στρατηγική θα ήταν να νοικιάσουμε τον εξοπλισμό κάθε μέρα που κάνουμε σκι μέχρι και την $(k-1)$-οστή και, αν μας δοθεί η ευκαιρία να κάνουμε σκι έστω και μια φορά ακόμη, να τον αγοράσουμε. Εύκολα διαπιστώνουμε ότι το χειρότερο ενδεχόμενο παρουσιάζεται όταν κάνουμε σκι επί ακριβώς k ημέρες. Τότε πληρώνουμε $(2k-1) \cdot 30$ αντί για $30k$ ευρώ, δηλαδή υφιστάμεθα $2 - \frac{1}{k}$ φορές μεγαλύτερο κόστος από το ελάχιστο δυνατό.

Εντούτοις, θα δείξουμε ότι ο παραπάνω τρόπος σκέψης είναι ο καλύτερος ως προς τη σύγκριση με την (ανέφικτη) βέλτιστη λύση που θα έδινε ένας offline αλγόριθμος έχοντας πλήρη γνώση του μέλλοντος. Τω όντι, ας αναλογιστούμε ποιες άλλες επιλογές έχουμε:

- Αν αγοράσουμε τον εξοπλισμό από την πρώτη μέρα αλλά δεν ξανακάνουμε σκι, θα έχουμε πληρώσει k φορές περισσότερα χρήματα.
- Αν νοικιάζουμε κάθε μέρα τον εξοπλισμό και κάνουμε πολλές ημέρες σκι, τότε πιθανόν να ξοδέψουμε ποσό πολλαπλάσιο της τιμής αγοράς του.
- Αν περιμένουμε την j-οστή μέρα που κάνουμε σκι για να προχωρήσουμε σε αγορά (ενώ μέχρι τότε αρκούμαστε στην ενοικίαση) και αυτή αποδειχθεί η τελευταία μέρα, πληρώνουμε κόστος $30(j-1) + 30k$ ευρώ, ενώ το ιδανικό θα ήταν $\min\{30j, 30k\}$ ευρώ. Ο λόγος μεταξύ των δύο αυτών ποσών ελαχιστοποιείται για $j = k$, οπότε καταλήγουμε στην προαναφερθείσα στρατηγική.

Συμπερασματικά, σύμφωνα με τους ορισμούς της προηγούμενης ενότητας, έχουμε βρει έναν online αλγόριθμο που είναι αυστηρά $\left(2 - \frac{1}{k}\right)$-ανταγωνιστικός και επιπροσθέτως ισχυρά ανταγωνιστικός.

Από τα παραπάνω διαφαίνεται μια γενικότερη αρχή, η επονομαζόμενη *αρχή του σκι* (*ski principle*). Μολονότι θα μπορούσαμε αφελώς να ισχυριστούμε



Σχήμα 1.1: Αναζητώντας τη γέφυρα

ότι είναι καλή ιδέα να λαμβάνει κανείς την απόφαση η οποία συνεπάγεται το ελάχιστο βραχυπρόθεσμο κόστος στην εκάστοτε περίπτωση, ένας online αλγόριθμος δεν πρέπει να λειτουργεί τόσο κοντόφθαλμα. Πράγματι, εκτελώντας κάποια στιγμή μια πράξη με υψηλό κόστος ενδέχεται να εξοικονομήσουμε πολύ περισσότερα στο μέλλον. Ουσιαστικά, η αρχή του σκι πρεσβεύει ότι άπαξ ο αλγόριθμος έχει χρεωθεί αρκετό κόστος ως συνέπεια πολλών σχετικά «φθηνών» πράξεων, τότε είναι λογικό να προχωρήσει σε μια πιο «ακριβή» ενέργεια, η οποία μπορεί να αποσβεσθεί από το συνολικό κόστος των υπολοίπων πράξεων.

1.2.2 Το πρόβλημα της γέφυρας

Έστω ότι βρισκόμαστε στην όχθη ενός ποταμού, η οποία είναι ευθεία. Γνωρίζουμε ότι σε κάποιο σημείο της όχθης υπάρχει μια γέφυρα, όμως δεν ξέρουμε πόσο απέχει από την τωρινή μας θέση, ούτε προς ποια κατεύθυνση πρέπει να κινηθούμε για να τη βρούμε. Το ζητούμενο είναι να εντοπίσουμε τη γέφυρα, ελαχιστοποιώντας την διαδρομή που διανύουμε αναζητώντας την.

Το πρόβλημα που διατυπώσαμε παραπάνω αναφέρεται στη βιβλιογραφία ως *πρόβλημα της γέφυρας*[1] (*bridge problem*). Θεωρείται online πρόβλημα διότι ο χάρτης της όχθης του ποταμού – που αποτελεί κατ' ουσίαν την είσοδο – μας αποκαλύπτεται σταδιακά καθώς την εξερευνούμε.

Είναι φανερό ότι, με όποιον τρόπο και αν ψάξουμε για τη γέφυρα, γενικά θα χρειαστεί να διατρέξουμε διαδρομή μεγαλύτερη από την απόσταση μεταξύ της γέφυρας και της αρχικής μας θέσης. Ειδικότερα, δεν είναι λογικό να κινούμαστε επ' άπειρον προς μία μόνο κατεύθυνση, διότι τότε υπάρχει σημαντική πιθανότητα να μη βρούμε ποτέ τη γέφυρα. Άρα, κατά διαστήματα θα πρέπει να κάνουμε μεταβολή, να επιστρέφουμε στο σημείο εκκίνησης και να συνεχίζουμε προς την αντίθετη κατεύθυνση, όπως φαίνεται στο Σχήμα 1.1.

Επομένως, η ελαχιστοποίηση της συνολικής διαδρομής ανάγεται στην κα-

[1] Επίσης απαντάται και ως *πρόβλημα της αγελάδας* (*cow problem*), στο οποίο μια μύωψ αγελάδα αναζητά κάποιο άνοιγμα σε έναν μακρύ φράχτη, ώστε να περάσει στην άλλη πλευρά, όπου βρίσκεται ένα καταπράσινο λιβάδι.

τάλληλη επιλογή των σημείων X_i. Επί παραδείγματι, έστω X_i τέτοια ώστε $\overrightarrow{OX}_{2i-1} = -\overrightarrow{OX}_{2i}$ και $|OX_{2i}| = i$, όπου O το σημείο εκκίνησής μας. Αν η γέφυρα βρίσκεται στο B, μεταξύ των σημείων X_{2k} και X_{2k+2} (για κάποιο $k \in \mathbb{N}^*$), δηλαδή $k < |OB| \leq k+1$, τότε το μήκος της διαδρομής που πρέπει να διανύσουμε ισούται με:

$$\begin{aligned} D &= |OX_1| + |X_1X_2| + \cdots + |X_{2k-1}X_{2k}| + |X_{2k}X_{2k+1}| + |X_{2k+1}B| = \\ &= 1 + 2 + \cdots + 2k + (2k+1) + (k+1+|OB|) = \\ &= 2k(k+1) + 1 + |OB| \approx \\ &\approx 2|OB|^2 . \end{aligned}$$

Ανάλογο αποτέλεσμα προκύπτει και σε κάθε άλλη περίπτωση. Συνεπώς, με τη μέθοδο αυτή η διαδρομή που διατρέχουμε αναζητώντας τη γέφυρα είναι περίπου ίση με το διπλάσιο του τετραγώνου της απόστασης της γέφυρας από την αρχική μας θέση. Η τετραγωνική αυτή εξάρτηση καθιστά τον παραπάνω αλγόριθμο όχι ιδιαίτερα ικανοποιητικό.

Ευτυχώς, όπως θα δούμε, υπάρχει τρόπος να βρούμε τη γέφυρα ούτως ώστε η διαδρομή D να μην υπερβαίνει ένα σταθερό πολλαπλάσιο της απόστασης $|OB|$. Συγκεκριμένα, επιλέγουμε τα σημεία X_i έτσι ώστε $|OX_i| = 2^{i-1}$. Αν $B \in X_kX_{k+2}$, $k \in \mathbb{N}^*$, τότε:

$$\begin{aligned} D &= |OX_1| + |X_1X_2| + \cdots + |X_{k-1}X_k| + |X_kX_{k+1}| + |X_{k+1}B| = \\ &= 1 + 3 + \cdots + 3 \cdot 2^{k-1} + 3 \cdot 2^k + (2^{k+1} + |OB|) = \\ &= 2^{k+3} - 2 + |OB| \leq \\ &\leq 9 \cdot |OB| - 2 , \end{aligned}$$

αφού $2^k < |OB| \leq 2^{k+2}$. Η παραπάνω σχέση δεν ισχύει όταν η γέφυρα βρίσκεται ανάμεσα στο σημείο εκκίνησης O και είτε το X_1 είτε το X_2. Στην πρώτη περίπτωση $D = |OB|$, ενώ στη δεύτερη $D = 2 + |OB|$. Άρα γενικά έχουμε:

$$D \leq 9 \cdot |OB| + 2 . \tag{1.1}$$

Επομένως, ο αλγόριθμος που περιγράψαμε είναι 9-ανταγωνιστικός, αφού η βέλτιστη offline λύση δεν είναι άλλη από το να καλύψουμε απόσταση $|OB|$ προς την κατεύθυνση της γέφυρας ώστε να φθάσουμε σε αυτή.

Στη συνέχεια αποδεικνύουμε ότι δεν υπάρχει online αλγόριθμος με λόγο ανταγωνιστικότητας μικρότερο από 9. Κατ' αρχάς, παρατηρούμε ότι οι χειρότερες περιπτώσεις για οποιονδήποτε τέτοιο αλγόριθμο εμφανίζονται όταν η γέφυρα βρίσκεται σε μια απειροελάχιστη απόσταση ϵ από κάποιο σημείο X_k και προς την κατεύθυνση που απομακρύνεται από το O. Τότε, θέτοντας

$|OX_i| = x_i$ έχουμε:

$$\begin{aligned}\frac{D}{|OB|} &= \frac{|OX_1| + |X_1X_2| + \cdots + |X_kX_{k+1}| + |X_{k+1}B|}{|OB|} = \\ &= \frac{x_1 + (x_1 + x_2) + \cdots + (x_k + x_{k+1}) + (x_{k+1} + x_k + \epsilon)}{x_k + \epsilon} \to \\ &\to \frac{2x_1 + 2x_2 + \cdots + 2x_{k+1} + x_k}{x_k},\end{aligned}$$

καθώς $\epsilon \to 0$. Πρέπει λοιπόν να ελαχιστοποιήσουμε την

$$f(x_1, x_2, \ldots, x_i, \ldots) \triangleq \sup_{n \geq 1} \left\{ 1 + 2\frac{\sum_{i=1}^{n+1} x_i}{x_n} \right\}. \tag{1.2}$$

Για το σκοπό αυτό, υποθέτουμε ότι $0 < x_1 \leq x_2 \leq \cdots \leq x_i \leq \cdots$ χωρίς βλάβη της γενικότητας. Θα δείξουμε λοιπόν ότι υπάρχουν $x'_1, x'_2, \ldots, x'_i, \ldots$ τέτοια ώστε $f(x_1, x_2, \ldots, x_i, \ldots) \geq f(x'_1, x'_2, \ldots, x'_i, \ldots)$ και επίσης

$$c^* = \frac{x'_1 + x'_2}{x'_1} = \frac{x'_1 + x'_2 + x'_3}{x'_2} = \cdots = \frac{x'_1 + x'_2 + \cdots + x'_{i+1}}{x'_i} = \cdots, \tag{1.3}$$

για κάποιο $c^* > 0$.

Θεωρούμε την παρακάτω άπειρη οικογένεια μετασχηματισμών:

$$\begin{array}{l|l|c|lcl|c} \hat{x}_1 = x_1 + \epsilon_1 & \hat{x}_1 = x_1(1+\epsilon_2) & & \hat{x}_1 & = & x_1(1+\epsilon_i) & \\ \hat{x}_2 = x_2 - \epsilon_1 & \hat{x}_2 = x_2(1+\epsilon_2) & & \hat{x}_2 & = & x_2(1+\epsilon_i) & \\ & \hat{x}_3 = x_3 - \epsilon_2(x_1 + x_2) & \cdots & & \vdots & & \cdots \\ & & & \hat{x}_{i-1} & = & x_{i-1}(1+\epsilon_i) & \\ & & & \hat{x}_i & = & x_i - \epsilon_i\left(\sum_{k=1}^{i-1} x_k\right) & \end{array}$$

Από τους όρους μεταξύ των οποίων λαμβάνουμε το supremum στη σχέση (1.2), ο πρώτος μετασχηματισμός επηρεάζει μόνο τους δύο που προκύπτουν για $n = 1$ και $n = 2$. Μάλιστα, ανάλογα με το πρόσημο του ϵ_1, ο ένας όρος θα αυξηθεί και ο άλλος θα μειωθεί. Αντίστοιχα, ο δεύτερος μετασχηματισμός μεταβάλλει μόνο τους όρους για $n = 2$ και $n = 3$ κ.ο.κ. Έτσι, με κατάλληλη εφαρμογή των παραπάνω μετασχηματισμών μπορούμε να εξισώσουμε όλους τους προαναφερθέντες όρους, δηλαδή να ικανοποιείται η (1.3), χωρίς να αυξηθεί το supremum τους.

Επιπλέον, θέτουμε $S_n = \sum_{i=1}^{n} x'_i$ και $S_0 = 0$. Προφανώς η ακολουθία S_n είναι γνησίως αύξουσα. Παρατηρούμε ακόμη ότι $S_{i+1} = c^*(S_i - S_{i-1})$. Για να επιλύσουμε την αναδρομική αυτή σχέση βρίσκουμε τη χαρακτηριστική εξίσωσή της, που είναι η $x^2 - c^*x + c^* = 0$ και έχει διακρίνουσα $\Delta = (c^*)^2 - 4c^*$. Αν η διακρίνουσα είναι αρνητική, δηλαδή οι λύσεις της εξίσωσης είναι μιγαδικές, τότε η ακολουθία S_n θα προκύψει ταλαντευόμενη, λαμβάνοντας θετικές και αρνητικές τιμές. Άτοπο, αφού S_n γνησίως μονότονη. Συνεπώς:

$$\Delta \geq 0 \iff c^*(c^*-4) \geq 0 \iff \left(c^* \geq 4 \quad \text{ή} \quad c^* \leq 0\right),$$

οπότε, αφού c^* θετικός, θα έχουμε $c^* \geq 4$ και τελικά:

$$f(x_1, x_2, \ldots, x_i, \ldots) \geq 1 + 2c^* \geq 9\,.$$

Συμπεραίνουμε λοιπόν ότι ο αλγόριθμος που αναφέραμε προηγουμένως είναι ισχυρά ανταγωνιστικός.

1.3 Άλλα ενδιαφέροντα online προβλήματα

Τα πλέον χαρακτηριστικά online προβλήματα είναι εκείνα στα οποία η είσοδος μπορεί να αναπαρασταθεί ως μια χρονική ακολουθία γεγονότων $\sigma = r_1, r_2, \ldots$, καθένα εκ των οποίων απαιτεί κάποια άμεση ενέργεια. Πέραν του προβλήματος ενοικίασης εξοπλισμού σκι (βλέπε Υποενότητα 1.2.1), παρακάτω θα αναφέρουμε συνοπτικά μερικά ακόμη γνωστά προβλήματα τέτοιας μορφής. Μάλιστα, τα σημαντικότερα από αυτά θα εξεταστούν εκτενώς σε επόμενα κεφάλαια.

Ενημέρωση Λίστας (List Update)

Μια συχνά χρησιμοποιούμενη δομή δεδομένων είναι η απλή συνδεδεμένη λίστα, που έχει ως κύριο χαρακτηριστικό τη γραμμική πρόσβαση στα δεδομένα. Δηλαδή, για να προσπελάσουμε το i-οστό στοιχείο της θα πρέπει να περάσουμε από όλα τα προηγούμενα, άρα μπορούμε να θεωρήσουμε ότι η ενέργεια αυτή έχει κόστος i. Προφανώς, η εισαγωγή ή η διαγραφή ενός στοιχείου από τη λίστα επίσης έχουν κόστος ανάλογο της θέσης αυτού.

Ας υποθέσουμε επιπλέον ότι αφού προσπελαστεί κάποιο στοιχείο επιτρέπεται να μετακινηθεί οπουδήποτε πλησιέστερα στην αρχή της λίστας με αμελητέο κόστος[2]. Έτσι, η πρόσβαση στο συγκεκριμένο στοιχείο καθίσταται «φθηνότερη», ενώ το αντίθετο συμβαίνει για εκείνα τα στοιχεία που μετατοπίζονται κατά μια θέση πιο πίσω εξαιτίας της λειτουργίας αυτής.

Έστω λοιπόν ότι οι αιτήσεις προσπέλασης στη λίστα σχηματίζουν τη χρονική ακολουθία σ. Η απόφαση για το αν ένα στοιχείο που μόλις προσπελάστηκε συμφέρει να μετακινηθεί πρέπει να ληφθεί χωρίς γνώση των μελλοντικών αιτήσεων, άρα αντιμετωπίζουμε ένα online πρόβλημα.

[2] Η εν λόγω υπόθεση εξαρτάται και από τον τρόπο υλοποίησης της λίστας. Εδώ παρουσιάζουμε μια σχετικά απλουστευμένη περίπτωση.

Σελιδοποίηση Μνήμης (Paging)

Κεντρικό ζήτημα στην περιοχή της Αρχιτεκτονικής Υπολογιστών είναι η διαχείριση της λανθάνουσας (cache) μνήμης, όπως αντίστοιχα στο πεδίο των Λειτουργικών Συστημάτων είναι η λειτουργία της εικονικής (virtual) μνήμης, απ' όπου προέρχεται και ο όρος *σελιδοποίηση*. Ουσιαστικά, το πρόβλημα είναι το ίδιο και στις δύο περιπτώσεις: διαθέτουμε μια «αργή» μνήμη μεγάλης χωρητικότητας και μια μικρότερη «γρήγορη» μνήμη[3]. Η δεύτερη χρησιμοποιείται για προσωρινή αποθήκευση ενός περιορισμένου μέρους των περιεχομένων της πρώτης, ώστε να μπορούμε να έχουμε ταχεία πρόσβαση σε αυτά.

Έστω λοιπόν ότι η αργή μνήμη διαθέτει n θέσεις αποθήκευσης (στη βιβλιογραφία αναφέρονται ως *σελίδες*) και η γρήγορη μνήμη μόνο $k < n$ τέτοιες, αλλιώς το πρόβλημα καθίσταται τετριμμένο. Θεωρούμε ότι λαμβάνουμε μια χρονική ακολουθία σ από αιτήσεις προσπέλασης σε σελίδες της αργής μνήμης.

Αν τα περιεχόμενα μιας σελίδας που ζητείται υπάρχουν εκείνη τη στιγμή στην γρήγορη μνήμη, η αντίστοιχη αίτηση ικανοποιείται χωρίς να υποστούμε κόστος. Σε αντίθετη περίπτωση, πρέπει να εκτελεστεί ανάγνωση της σελίδας από την αργή μνήμη, με μοναδιαίο κόστος. Τα μόλις ανακτηθέντα δεδομένα μεταφέρονται στη γρήγορη μνήμη, ώστε να είναι διαθέσιμα για μελλοντική αναφορά. Αν υπάρχει ελεύθερος χώρος εκεί, έχει καλώς. Ειδάλλως, θα πρέπει πρώτα να τον δημιουργήσουμε, απομακρύνοντας κάποια ήδη αποθηκευμένη σελίδα.

Η επιλογή της σελίδας η οποία θα αντικατασταθεί σε κάθε τέτοια περίπτωση είναι καθοριστική, διότι ενδέχεται λίγο αργότερα να λάβουμε αίτηση για εκείνα ακριβώς τα δεδομένα που εκδιώξαμε από τη γρήγορη μνήμη προηγουμένως. Πρόκειται λοιπόν για ένα online πρόβλημα, στο οποίο στόχος είναι η ελαχιστοποίηση των προσπελάσεων της αργής μνήμης.

k-Εξυπηρετητές (k-Server)

Έστω $k \in \mathbb{N}^*$ και $\mathcal{M} = (M, d)$ μετρικός χώρος, όπου M σύνολο σημείων με $|M| > k$ και d μετρική επί του M. Θεωρούμε ότι υπάρχουν k κινητοί *εξυπηρετητές* (*servers*), τοποθετημένοι αρχικά σε κάποια σημεία του χώρου. Η είσοδος του προβλήματος είναι μία ακολουθία αιτήσεων $\sigma = r_1, r_2, \ldots$, όπου κάθε αίτηση r_i αποτελείται απλώς από ένα σημείο του χώρου. Για να ικανοποιηθεί, πρέπει στο σημείο εκείνο να βρίσκεται τουλάχιστον ένας εξυπηρετητής. Αν τη χρονική στιγμή που διατυπώνεται η αίτηση δεν ισχύει κάτι τέτοιο, κάποιος από τους εξυπηρετητές οφείλει να μετακινηθεί στο ζητούμενο σημείο. Η ενέργεια αυτή έχει κόστος ίσο με την απόσταση που διανύει ο εξυπηρετητής,

[3]Το ζεύγος («αργή» μνήμη, «γρήγορη» μνήμη) αναφέρεται είτε στο (κύρια μνήμη, λανθάνουσα μνήμη) είτε στο (βοηθητική μνήμη, κύρια μνήμη).

όπως καθορίζεται από την μετρική d. Όπως είναι φανερό, επιθυμούμε να ελαχιστοποιήσουμε το συνολικό κόστος για την διαδοχική ικανοποίηση όλων των αιτήσεων της σ.

Σημείωση. Το εν λόγω online πρόβλημα διατυπώθηκε για πρώτη φορά από τους Manasse, McGeoch και Sleator [MMS88], παρουσιάζει δε ιδιαίτερο ενδιαφέρον λόγω της γενικότητάς του και έχει αποτελέσει το αντικείμενο αξιόλογης ερευνητικής προσπάθειας, που συνεχίζεται μέχρι σήμερα.

Μάλιστα, το πρόβλημα σελιδοποίησης μνήμης ανάγεται στο πρόβλημα των k-εξυπηρετητών. Για να το δείξουμε αυτό, αρκεί να θεωρήσουμε μετρικό χώρο $\mathcal{M} = (M, d)$ με $|M| = n$, στον οποίο η απόσταση μεταξύ δύο οποιωνδήποτε σημείων ισούται πάντοτε με 1· τουτ' έστιν, ο $\mathcal{M}$ είναι *ομοιόμορφος*. Εύκολα διαπιστώνουμε ότι το πρόβλημα των k-εξυπηρετητών επί ενός τέτοιου μετρικού χώρου ισοδυναμεί με εκείνο της σελιδοποίησης.

Μετρικά Συστήματα Εργασιών (Metrical Task Systems)

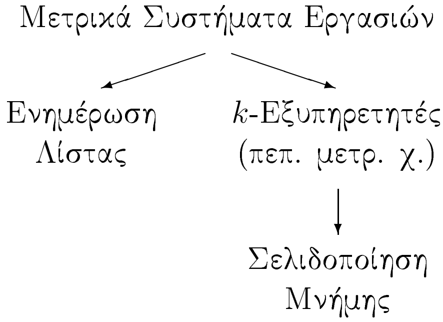
Έστω $\mathcal{M} = (M, d)$ μετρικός χώρος πεπερασμένου πλήθους σημείων, ο οποίος μοντελοποιεί αφηρημένα κάποιο σύστημα, το οποίο έχει τη δυνατότητα να εκτελεί διάφορες εργασίες. Δηλαδή, κάθε σημείο του $\mathcal{M}$ αναπαριστά μια κατάσταση του συστήματος, ενώ η απόσταση μεταξύ δύο οποιωνδήποτε σημείων είναι ανάλογη προς το κόστος μετάβασης μεταξύ των αντιστοίχων καταστάσεων. Καθεμία εργασία r περιγράφεται από ένα διάνυσμα $\langle r(1), r(2), \ldots, r(N)\rangle$, όπου $N = |M|$ και $r(j) \in \mathbb{R}^+ \cup \{\infty\}$ ισούται με το κόστος εκτέλεσης της εργασίας όταν το σύστημα βρίσκεται στην κατάσταση j. Τοιουτοτρόπως, *μετρικό σύστημα εργασιών* είναι ένα ζεύγος $(\mathcal{M}, \mathcal{R})$, όπου $\mathcal{R}$ κάποιο σύνολο επιτρεπτών εργασιών.

Υποθέτουμε, λοιπόν, ότι η αρχική κατάσταση του συστήματος είναι η s_0 και ότι δίνεται ακολουθία εργασιών $\sigma = r_1, r_2, \ldots$ προς εκτέλεση. Όταν ανατίθεται μια εργασία r, έχουμε τη δυνατότητα πρώτα να μεταβάλουμε την κατάσταση του συστήματος και κατόπιν να εκτελέσουμε την r. Αν λοιπόν $s[i]$ είναι η κατάσταση υπό την οποία εκτελέστηκε η εργασία r_i, το συνολικό κόστος για την ακολουθία σ είναι:

$$\text{κόστος}(\sigma) \triangleq \underbrace{\sum_{i=1}^{n} d(s[i-1], s[i])}_{\substack{\text{συνολικό κόστος}\\ \text{μεταβάσεων}}} + \underbrace{\sum_{i=1}^{n} r_i(s[i])}_{\substack{\text{συνολικό κόστος}\\ \text{επεξεργασίας}}} ,$$

το οποίο φυσικά επιθυμούμε να ελαχιστοποιήσουμε.

Το *πρόβλημα της μηχανής παγωτού* (*ice cream machine problem*) αποτελεί ένα απλό παράδειγμα μετρικού συστήματος εργασιών. Έστω ότι έχουμε



Σχήμα 1.2: «Γενεαλογικό δένδρο» online προβλημάτων

μια μηχανή που παρασκευάζει δύο ειδών παγωτά: βανίλια και σοκολάτα. Το κόστος παρασκευής ενός κιλού παγωτού βανίλιας είναι 1€, ενώ για τη σοκολάτα 2€. Βέβαια, η μηχανή μπορεί να παράγει μόνο έναν από τους δύο τύπους παγωτού ανά πάσα στιγμή, ανάλογα με τις ρυθμίσεις της. Η αλλαγή των ρυθμίσεων αυτών δεν είναι εύκολη διαδικασία και κοστίζει 1€.

Από την άλλη μεριά, έχουμε τη δυνατότητα να αγοράσουμε παγωτό από εξωτερικό προμηθευτή, όμως η τιμή της βανίλιας είναι 2€/κιλό και της σοκολάτας 4€/κιλό. Αν λοιπόν δεχόμαστε διαδοχικές παραγγελίες για παγωτό πότε του ενός και πότε του άλλου είδους, θέλουμε να τις ικανοποιήσουμε με το μικρότερο δυνατό κόστος, αλλάζοντας τις ρυθμίσεις της μηχανής όποτε αυτό μας συμφέρει.

Σημείωση. Τα μετρικά συστήματα εργασιών ορίστηκαν από τους Borodin, Linial και Saks [BorLS87]. Τονίζεται ότι το πρόβλημα ενημέρωσης λίστας, όπως και εκείνο των k-εξυπηρετητών (σε πεπερασμένους μετρικούς χώρους), αποτελούν υποπεριπτώσεις μετρικών συστημάτων εργασιών. Έτσι, οι σχέσεις μεταξύ των προβλημάτων που εκθέσαμε μέχρι τώρα στην παρούσα υποενότητα απεικονίζονται στο Σχήμα 1.2.

Κεφάλαιο 2

Σχεδίαση και Ανάλυση Αιτιοκρατικών Online Αλγορίθμων

2.1 Τεχνικές σχεδίασης

Λίγα χρόνια πριν, οι Irani και Karlin [IK97] σχολίαζαν ότι «η σχεδίαση online αλγορίθμων εξακολουθεί να αποτελεί, σε μεγάλο βαθμό, μια τέχνη». Το ίδιο μπορούμε να ισχυριστούμε και σήμερα. Μολαταύτα, υπάρχουν ορισμένα επαναλαμβανόμενα «μοτίβα» που έχουν αξιοποιηθεί επιτυχώς σε ποικίλα προβλήματα. Μεταξύ αυτών διακρίνουμε τα εξής:

- Άπληστοι (greedy) αλγόριθμοι
- Εξισορρόπηση κόστους (balancing costs)
- Αύξηση του offline κόστους
- Συνδυασμός online αλγορίθμων

Στη συνέχεια, θα αναλύσουμε το καθένα ξεχωριστά.

2.1.1 Άπληστοι αλγόριθμοι

Κατά μια ευρεία έννοια, όλοι οι online αλγόριθμοι των οποίων η είσοδος αποτελείται από μια ακολουθία αιτήσεων $\sigma = r_1, r_2, \ldots$ είναι *άπληστοι*. Τω όντι, κάθε ενέργειά τους εξαρτάται από την τρέχουσα αίτηση, την τρέχουσα κατάσταση και το ιστορικό των προηγούμενων αιτήσεων. Ουσιαστικά, σε κάθε βήμα ο αλγόριθμος αποτιμά τα παραπάνω δεδομένα σύμφωνα με κάποια σταθερά κριτήρια, ώστε να αποφασίσει ποιος είναι ο καλύτερος τρόπος για να εξυπηρετήσει την εκάστοτε αίτηση. Εντούτοις, χρήζουν ιδιαίτερης μνείας ορισμένοι αλγόριθμοι στους οποίους ο χαρακτηρισμός «άπληστος» ταιριάζει αβίαστα.

Τοπικά άπληστος (local greedy) αλγόριθμος

Πρόκειται για τον πιο κλασικό άπληστο αλγόριθμο, διότι επιλέγει πάντοτε την ενέργεια που ελαχιστοποιεί το κόστος εξυπηρέτησης της τρέχουσας αίτησης· πράττει δηλαδή ακριβώς το αντίθετο από αυτό που υπαγορεύει η αρχή του σκι. Συνεπώς, σε πολλές περιπτώσεις η απόδοσή του απογοητεύει παντελώς – επί παραδείγματι, για το πρόβλημα των k-εξυπηρετητών δεν είναι ανταγωνιστικός. Αποδεικνύεται όμως ισχυρά ανταγωνιστικός στην online κατασκευή δένδρων Steiner, σε ορισμένα προβλήματα εξισορρόπησης φορτίου και όχι μόνο. Υπάρχουν επιπροσθέτως και τοπικά άπληστοι αλγόριθμοι με βάρη. Σε αυτή την κατηγορία ανήκει η τεχνική εκθετικής συνάρτησης, η οποία εφαρμόζεται στη δρομολόγηση εικονικών κυκλωμάτων.

Αλγόριθμος αναδρομικής εξέτασης (retrospective)

Ο εν λόγω αλγόριθμος υπολογίζει σε ποιες καταστάσεις θα ήταν δυνατό να βρίσκεται ο βέλτιστος offline αλγόριθμος αν η τρέχουσα αίτηση ήταν και η τελευταία της ακολουθίας εισόδου. Ακολούθως μετακινείται στην «πλησιέστερη» εξ αυτών των καταστάσεων, δηλαδή σε εκείνη που απαιτεί το μικρότερο κόστος μετάβασης. Τον αλγόριθμο αυτόν εφαρμόσαμε στο πρόβλημα ενοικίασης εξοπλισμού σκι της Υποενότητας 1.2.1, όπου διαπιστώσαμε μάλιστα ότι έχει τον καλύτερο δυνατό λόγο ανταγωνιστικότητας. Δυστυχώς, δεν ισχύει το ίδιο όσον αφορά το πρόβλημα των k-εξυπηρετητών, για το οποίο δεν είναι ανταγωνιστικός. Πέραν τούτων, μια ενδιαφέρουσα εκδοχή του αλγορίθμου αναπτύχθηκε από τους Kalyanasundaram και Pruhs [KalyP91] και ανεξάρτητα από τους Khuller, Mitchell και Vazirani [KhulMV91], για online ταιριάσματα ελαχίστου βάρους σε διμερείς γράφους.

Αλγόριθμος συνάρτησης έργου (work function)

Ο διάσημος αυτός αλγόριθμος [BorLS87, CL92, KoutP94] προκύπτει ως γραμμικός συνδυασμός των δύο προηγουμένων και λειτουργεί ως εξής: Ορίζουμε $w_{\sigma_t}(s)$ το βέλτιστο κόστος για την εξυπηρέτηση των αιτήσεων $r_1, r_2, \ldots, r_t$ με κατάληξη την κατάσταση s. Αν μετά την t-οστή αίτηση ο αλγόριθμος βρίσκεται στην κατάσταση s_t, τότε θα εξυπηρετήσει την $(t+1)$-οστή αίτηση αφού πρώτα μετακινηθεί στην κατάσταση s_{t+1} που ελαχιστοποιεί την ποσότητα

$$w_{\sigma_{t+1}}(s_{t+1}) + d(s_t, s_{t+1}) \; . \tag{2.1}$$

Γνωρίζουμε ότι ο αλγόριθμος συνάρτησης έργου παρουσιάζει τον βέλτιστο (ή σχεδόν τον βέλτιστο) λόγο ανταγωνιστικότητας για πολλά online προβλήματα, με κυριότερο εκείνο των k-εξυπηρετητών. Η απόδειξη του τελευταίου παρατίθεται στην Ενότητα 4.3.

Βέβαια, σε άλλες περιπτώσεις ο παραπάνω αλγόριθμος δεν αποδίδει ικανοποιητικά. Επί παραδείγματι, δεν είναι ανταγωνιστικός για το πρόβλημα της Διαστρωματωμένης Διάσχισης Γράφου (Layered Graph Traversal), όπως έδειξε ο Burley [Bur93]. Μάλιστα, στην ίδια εργασία αποδεικνύεται ότι μία παραλλαγή του αλγορίθμου είναι ανταγωνιστική: αρκεί να αντικαταστήσουμε την προς ελαχιστοποίηση ποσότητα (2.1) με την

$$\alpha \cdot w_{\sigma_{t+1}}(s_{t+1}) + d(s_t, s_{t+1}) \ ,$$

για κάποιο $\alpha > 1$. Πάντως, ο προσδιορισμός των online προβλημάτων για τα οποία καθένας από τους προαναφερθέντες άπληστους αλγορίθμους είναι ανταγωνιστικός παραμένει ένα σπουδαίο ανοικτό πρόβλημα.

Παρατήρηση. Ένα πρακτικό μειονέκτημα που εμφανίζεται κατά την υλοποίηση των αλγορίθμων αναδρομικής εξέτασης και συνάρτησης έργου είναι η ανάγκη για σημαντικό χώρο μνήμης κατά τον υπολογισμό του βέλτιστου κόστους $w_t(s)$. Ειδικότερα, ο απαιτούμενος χώρος προκύπτει συνήθως ανάλογος του συνολικού πλήθους καταστάσεων, το οποίο είναι μικρό μόνο για σχετικά εύκολα προβλήματα όπως αυτό της ενοικίασης εξοπλισμού σκι.

2.1.2 Εξισορρόπηση κόστους

Η αρχή της εξισορρόπησης κόστους λίγο διαφέρει από την αρχή του σκι που διατυπώσαμε προηγουμένως. Ας υποθέσουμε, χάριν απλότητας, ότι ο online αλγόριθμος έχει δύο επιλογές για την εξυπηρέτηση μιας αίτησης. Όπως είναι φυσικό, για κάποιες πιθανές ακολουθίες μελλοντικών αιτήσεων η μία επιλογή θα αποδειχθεί μακροπρόθεσμα πιο συμφέρουσα, ενώ για τις υπόλοιπες θα συμβεί το αντίθετο. Εντούτοις, η εξισορρόπηση κόστους πρεσβεύει ότι ο αλγόριθμος πρέπει να φροντίσει ώστε, είτε πραγματοποιηθεί το ένα είτε το άλλο ενδεχόμενο, το τελικό κόστος να είναι (χονδρικά) το ίδιο.

Ένα παράδειγμα αλγορίθμου που αξιοποιεί την παραπάνω ιδέα είναι ο επονομαζόμενος BALANCE [MMS88], ο οποίος αποδεικνύεται βέλτιστος για το πρόβλημα των k-εξυπηρετητών σε μετρικούς χώρους με $k+1$ ακριβώς σημεία. Ο BALANCE συγκρατεί στη μνήμη του τη συνολική διαδρομή D_i που έχει διανύσει ο i-οστός εξυπηρετητής. Αν d_i είναι η απόσταση του τελευταίου από το σημείο της επόμενης αίτησης, ο BALANCE θα μετακινήσει στο σημείο αυτό τον εξυπηρετητή για τον οποίο η ποσότητα $D_i + d_i$ ελαχιστοποιείται.

Προφανώς, επιδίωξη του αλγορίθμου είναι όλοι οι εξυπηρετητές να διανύσουν περίπου την ίδια συνολική απόσταση. Τοιουτοτρόπως αποφεύγεται, σε μεγάλο βαθμό, το ενδεχόμενο μόνο ένα υποσύνολο των εξυπηρετητών να είναι ενεργό. Για να κατανοήσουμε τη σημασία του γεγονότος αυτού, αρκεί να αναλογιστούμε γιατί ο τοπικά άπληστος αλγόριθμος δεν τα καταφέρνει τόσο καλά στο ίδιο πρόβλημα. Πράγματι, ας θεωρήσουμε ότι έχουμε δύο εξυπηρετητές και έστω ότι δύο σημεία A, B με πολύ μικρή απόσταση μεταξύ τους εμφανίζονται εναλλάξ σε διαδοχικές αιτήσεις. Ο τοπικά άπληστος αλγόριθμος θα μετακινήσει αρχικά τον πλησιέστερο εξυπηρετητή a, ο οποίος στη συνέχεια θα παλινοδεί διαρκώς ανάμεσα στα A και B. Εν τω μεταξύ, ο πιο απομακρυσμένος εξυπηρετητής b θα παραμένει άεργος, επειδή η μεταφορά του σε κάποιο από τα A, B επισύρει υψηλό βραχυπρόθεσμο κόστος. Ο BALANCE όμως αποτρέπει μια τέτοια εξέλιξη, αφού μόλις η συνολική διαδρομή που έχει διανύσει ο a αυξηθεί αρκετά, ο αλγόριθμος θα μετακινήσει τον b.

Ακολούθως αποδεικνύεται ότι ο BALANCE είναι k-ανταγωνιστικός σε μερικούς χώρους με $k+1$ ακριβώς σημεία.

Θεώρημα 2.1. [MMS88] *Έστω $\mathcal{M} = (M, d)$ μετρικός χώρος με $|M| = k+1$ και c_{BAL} ο λόγος ανταγωνιστικότητας του* BALANCE *για το πρόβλημα των k-εξυπηρετητών στον $\mathcal{M}$. Τότε $c_{\text{BAL}} \leq k$.*

Απόδειξη. Ανά πάσα στιγμή, υπάρχει ακριβώς ένα σημείο του $\mathcal{M}$ το οποίο δεν καλύπτεται από κάποιον εξυπηρετητή. Ονομάζουμε το σημείο αυτό *οπή*. Χωρίς βλάβη της γενικότητας υποθέτουμε ότι κάθε νέα αίτηση αφορά την εκάστοτε οπή. Άλλωστε, μια αίτηση προς οποιοδήποτε διαφορετικό σημείο σίγουρα δεν κοστίζει τίποτε για τον BALANCE, ενώ πιθανόν να αυξήσει το βέλτιστο offline κόστος.

Αριθμούμε λοιπόν τα σημεία του χώρου $\{1, 2, \ldots, k+1\}$. Έστω s_i η κατάσταση κατά την οποία η οπή βρίσκεται στο σημείο i και $w_t(s_i)$ το βέλτιστο κόστος εξυπηρέτησης των πρώτων t αιτήσεων με κατάληξη την s_i. Σημειωτέον ότι το κόστος μετάβασης από την κατάσταση s_i στην s_j είναι $d(i, j)$ και άρα $w_t(s_i) \leq w_t(s_j) + d(i, j)$.

Συμβολίζουμε με h^t τη θέση της οπής μετά την ικανοποίηση της t-οστής αίτησης και με D_i^t τη συνολική διαδρομή που έχει διανύσει ο εξυπηρετητής που βρίσκεται στο σημείο $i \neq h^t$ εκείνη τη στιγμή. Φυσικά, η επόμενη αίτηση θα αφορά το σημείο h^t. Ισχυριζόμαστε ότι:

$$i \neq h^t \implies D_i^t \leq w_t(s_i)\,. \tag{2.2}$$

Θα αποδείξουμε την παραπάνω πρόταση με επαγωγή στο t.

- Για $t = 0$ προφανώς αληθεύει.

- Έστω ότι για κάποιο $u \geq 0$ ισχύει $i \neq h^u \Rightarrow D_i^u \leq w_u(s_i)$. Αν $min \neq h^u$ το σημείο που ελαχιστοποιεί την ποσότητα $D_i^u + d(i, h^u)$, ο BALANCE θα στείλει στην οπή h^u τον εξυπηρετητή που βρίσκεται στο min, ώστε να ικανοποιήσει την $(u+1)$-οστή αίτηση. Τότε $D_{h^u}^{u+1} \leftarrow D_{min}^u + d(min, h^u)$ και επιπλέον:

$$w_{u+1}(s_{h^u}) \longleftarrow \min_{i \neq h^u} \{w_u(s_i) + d(i, h^u)\} \ .$$

Λόγω της επαγωγικής υπόθεσης προκύπτει ότι $D_{h^u}^{u+1} \leq w_{u+1}(s_{h^u})$. Επίσης, για όλες τις τιμές του j πλην των min και h^u ισχύει $D_j^{u+1} = D_j^u$ ενώ $w_{u+1}(s_j) \geq w_u(s_j)$.

Άρα η πρόταση (2.2) είναι αληθής. Επομένως, για το συνολικό κόστος του BALANCE μέχρι και την t-οστή αίτηση έχουμε:

$$\begin{aligned}
\text{κόστος}_{\text{BAL}}(r_1, r_2, \ldots, r_t) &= \sum_{i \neq h^t} D_i^t \leq \\
&\leq \sum_{i \neq h^t} w_t(s_i) \leq \\
&\leq k \left[\min_{1 \leq i \leq k+1} \{w_t(s_i)\} + \max_{i \neq j} \{d(i,j)\} \right] = \\
&= k \left[\min_{1 \leq i \leq k+1} \{w_t(s_i)\} \right] + k \left[\max_{i \neq j} \{d(i,j)\} \right] \leq \\
&\leq k \cdot \text{κόστος}_{\text{OPT}}(r_1, r_2, \ldots, r_t) + b \,,
\end{aligned}$$

όπου b σταθερά ανεξάρτητη του μήκους της ακολουθίας εισόδου. □

Σε άλλη ενότητα του κεφαλαίου θα δούμε ότι ο καλύτερος δυνατός λόγος ανταγωνιστικότητας για το συγκεκριμένο πρόβλημα είναι τω όντι k.

Όπως θα ήταν αναμενόμενο, η εξισορρόπηση κόστους δεν αποτελεί πανάκεια. Λόγου χάρη, εύκολα διαπιστώνουμε ότι αν ο μετρικός χώρος του προβλήματος των k-εξυπηρετητών περιλαμβάνει περισσότερα από $k+1$ σημεία, τότε ο BALANCE δεν είναι πλέον ανταγωνιστικός, ακόμη και στην απλή περίπτωση που $k = 2$. Ένα κατάλληλο αντιπαράδειγμα, συνοδευόμενο από μια δεύτερη απόδειξη του Θεωρήματος 2.1, παρουσιάζεται στην Ενότητα 4.2.

Μια παραλλαγή του αλγορίθμου, που προτάθηκε από τους Irani και Rubinfeld [IR91], μετακινεί κάθε φορά τον εξυπηρετητή που ελαχιστοποιεί την ποσότητα $D_i + 2d_i$ και είναι 10-ανταγωνιστική για δύο εξυπηρετητές. Όμως, οι Chrobak και Larmore [CL91a] έδειξαν ότι ο λόγος ανταγωνιστικότητας της παραλλαγής αυτής φράσσεται κάτω από το 6.

Τέλος, ο Kleinberg [Klei94] μελέτησε γενικευμένους BALANCE αλγορίθμους, στους οποίους κριτήριο επιλογής εξυπηρετητή είναι η ελαχιστοποίηση του αθροίσματος $D_i + f(d_i)$, όπου f οποιαδήποτε πραγματική συνάρτηση. Απέδειξε λοιπόν ότι ο λόγος ανταγωνιστικότητας τέτοιων αλγορίθμων για δύο εξυπηρετητές έχει κάτω φράγμα $\frac{5+\sqrt{7}}{2} \approx 3,82$.

2.1.3 Αύξηση του offline κόστους

Εφόσον το βέλτιστο offline κόστος αποτελεί μέτρο σύγκρισης, ένας καλός online αλγόριθμος μπορεί να φροντίσει ώστε τα μη ευνοϊκά για εκείνον στιγμιότυπα εισόδου να επισύρουν και αύξηση του κόστους αυτού.

Παράδειγμα εφαρμογής της αρχής αυτής είναι μια κατηγορία αλγορίθμων για το πρόβλημα της σελιδοποίησης, γνωστοί ως αλγόριθμοι σήμανσης (*marking algorithms*). Ένας οποιοσδήποτε τέτοιος αλγόριθμος λειτουργεί σε *φάσεις*. Στην αρχή μιας φάσης, όλες οι σελίδες που φυλάσσονται στη γρήγορη μνήμη δεν είναι *σεσημασμένες*[1] (*marked*). Αν κάποια από αυτές ζητηθεί, τότε σημειώνεται. Αν πάλι μια αίτηση καταστήσει υποχρεωτική την προσπέλαση της αργής μνήμης, έχουμε δύο ενδεχόμενα:

- Υπάρχουν μη σεσημασμένες σελίδες στη γρήγορη μνήμη. Τότε επιλέγεται μια εξ αυτών προς απομάκρυνση, με τρόπο που καθορίζεται από τον αλγόριθμο, και αντικαθίσταται από τη μόλις ανακτηθείσα σελίδα, η οποία και σημειώνεται.

- Δεν υπάρχουν μη σεσημασμένες σελίδες στη γρήγορη μνήμη. Στην περίπτωση αυτή η τρέχουσα φάση λήγει, οπότε αρχίζει η επόμενη. Όλες οι προϋπάρχουσες σημάνσεις αίρονται και η εκκρεμούσα αίτηση εξυπηρετείται σύμφωνα με τα παραπάνω.

Σημειωτέον ότι η αρχή και το τέλος των φάσεων εξαρτώνται αποκλειστικά από την ακολουθία των αιτήσεων και όχι από την επιλογή των προς αντικατάσταση σελίδων.

Το σκεπτικό με το οποίο ενεργούν οι αλγόριθμοι σήμανσης είναι ότι, στην χειρότερη περίπτωση, οποιοσδήποτε αιτιοκρατικός online αλγόριθμος θα αναγκαστεί να προσπελαύνει την αργή μνήμη σε κάθε αίτηση. Συνεπώς, μπορεί μόνο να ελπίζει ότι θα διαχειριστεί σωστά τη γρήγορη μνήμη, δηλαδή ότι θα εκτελέσει τις κατάλληλες αντικαταστάσεις, ώστε οι δυσμενείς γι' αυτόν ακολουθίες να μην είναι δυνατό να ικανοποιηθούν με μικρό κόστος από κάποιον offline αλγόριθμο. Η εν λόγω ιδέα τεκμηριώνεται με το ακόλουθο θεώρημα.

[1] Εκ συμβάσεως, ο κενός χώρος θεωρείται επίσης μη σεσημασμένος.

Θεώρημα 2.2. [KarlMRS88] *Κάθε αλγόριθμος σήμανσης για το πρόβλημα σελιδοποίησης μνήμης είναι k-ανταγωνιστικός.*

Απόδειξη. Έστω S_i το διάστημα από τη δεύτερη αίτηση της i-οστής φάσης μέχρι και την πρώτη αίτηση της επόμενης. Αρκεί να παρατηρήσουμε ότι ο βέλτιστος αλγόριθμος OPT θα αναγκαστεί να εκτελέσει τουλάχιστον μια προσπέλαση της αργής μνήμης σε κάθε τέτοιο διάστημα.

Πράγματι, στην αρχή του διαστήματος S_i, ο OPT διαθέτει στη γρήγορη μνήμη τα περιεχόμενα της σελίδας που μόλις είχε ζητηθεί από την πρώτη αίτηση της i-οστής φάσης. Αν χρειαστεί να ανακτήσει δεδομένα από την αργή μνήμη πριν το τέλος της φάσης, ο παραπάνω ισχυρισμός αληθεύει ούτως ή άλλως. Αν όχι, σημαίνει ότι η γρήγορη μνήμη του OPT περιείχε ήδη εκ των προτέρων όλες τις σελίδες που ζητήθηκαν κατά τη φάση αυτή, οι οποίες φυσικά είναι k το πλήθος. Η πρώτη αίτηση της $(i+1)$-οστής φάσης, όμως, δεν θα αφορά καμία από εκείνες τις σελίδες, άρα τότε ο OPT θα προσπελάσει υποχρεωτικά την αργή μνήμη.

Παρομοίως διαπιστώνουμε ότι σε κάθε φάση ένας marking αλγόριθμος εκτελεί το πολύ k ανακτήσεις από την αργή μνήμη. Τούτο συμβαίνει διότι τα περιεχόμενα μιας σελίδας που έχει ήδη ζητηθεί παραμένουν αποθηκευμένα στη γρήγορη μνήμη τουλάχιστον μέχρι το τέλος της φάσης. Άρα, ποτέ δεν χρειάζεται να μεταφερθούν τα ίδια δεδομένα από την αργή μνήμη παραπάνω από μία φορά κατά τη διάρκεια μιας φάσης. Αφού ζητούνται ακριβώς k διαφορετικές σελίδες σε κάθε φάση, η προηγούμενη διαπίστωση ισχύει.

Συνδυάζοντας όλα τα προαναφερθέντα, έπεται τελικά ότι ο marking αλγόριθμος θα προσπελάσει την αργή μνήμη το πολύ k φορές περισσότερες από τον OPT, modulo κάποια σταθερά. $\square$

Όπως βλέπουμε, ένας αλγόριθμος σήμανσης χρησιμοποιεί το πρόσφατο παρελθόν ώστε να προβλέψει, σε κάποιο βαθμό, το μέλλον. Οι σεσημασμένες σελίδες έχουν ζητηθεί πιο πρόσφατα από όλες τις μη σεσημασμένες. Υποθέτοντας ότι μια αίτηση που μόλις εξυπηρετήθηκε είναι πιθανό να επαναληφθεί προσεχώς στην ακολουθία, ο αλγόριθμος δεν απομακρύνει τις σεσημασμένες σελίδες από τη γρήγορη μνήμη. Με άλλα λόγια, εκμεταλλεύεται την ιδιότητα της *χρονικής τοπικότητας αναφοράς*.

Ένας τέτοιος αλγόριθμος θα παρουσιάζει καλή επίδοση, στα πλαίσια της ανταγωνιστικής ανάλυσης. Τω όντι, είναι φανερό πως ο βέλτιστος offline αλγόριθμος ικανοποιεί με χαμηλό κόστος ακολουθίες που εμφανίζουν τοπικότητα αναφοράς, ενώ δεν μπορεί να κάνει το ίδιο για ακολουθίες στις οποίες πολλές διαφορετικές σελίδες ζητούνται διαδοχικά. Για να είναι ανταγωνιστικός, ο online αλγόριθμος οφείλει να παρουσιάζει καλή επίδοση στις περιπτώσεις που ευνοούν τον offline αλγόριθμο, δηλαδή ακριβώς στις ακολουθίες με τοπικότητα αναφοράς.

Δεν αποτελεί λοιπόν έκπληξη το γεγονός ότι ο ευρύτατα χρησιμοποιούμενος στην πράξη αλγόριθμος LRU ανήκει στην κατηγορία των αλγορίθμων σήμανσης. Ο LRU λειτουργεί ως εξής: όταν απαιτηθεί αντικατάσταση μιας από τις σελίδες που είναι αποθηκευμένες στη γρήγορη μνήμη, επιλέγει εκείνη στην οποία δεν έχει γίνει αναφορά για το μεγαλύτερο διάστημα κατά το παρελθόν.

Η γενική ιδέα που περιγράψαμε αξιοποιείται επίσης στο πρόβλημα ενημέρωσης λίστας και συγκεκριμένα στον αλγόριθμο MTF (Move-To-Front – «Μετακίνησε Στην Αρχή» [ST85a]), τον οποίο θα αναλύσουμε στη συνέχεια. Από τη σύζευξη της εν λόγω τεχνικής και της αρχής εξισορρόπησης κόστους προκύπτει ο σχετικά εξεζητημένος αλγόριθμος Διάσχισης (Traversal algorithm [BorLS87]).

2.1.4 Συνδυασμός online αλγορίθμων

Έστω $\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_m$ ένα σύνολο online αλγορίθμων για το πρόβλημα Π. Θεωρούμε ότι το Π μπορεί να αναπαρασταθεί ως μετρικό σύστημα εργασιών και έχει σύνολο στιγμιοτύπων $\mathcal{I}$. Κάθε αλγόριθμος ALG_i επιτυγχάνει λόγο ανταγωνιστικότητας α συγκρινόμενος με τον βέλτιστο offline αλγόριθμο, αλλά μόνο για κάποιο υποσύνολο $\mathcal{I}_i$ των δυνατών εισόδων. Αν ισχύει $\bigcup \mathcal{I}_i \supseteq \mathcal{I}$, τότε είναι δυνατό να κατασκευαστεί ένας σύνθετος αιτιοκρατικός online αλγόριθμος ALG^* για το Π, χρησιμοποιώντας βέβαια τους m επιμέρους αλγορίθμους. Ο λόγος ανταγωνιστικότητας του ALG^* θα είναι $\alpha(2em+1)$, όπου $e \approx 2,72$ η βάση των φυσικών λογαρίθμων. Η μέθοδος αυτή συνδυασμού online αλγορίθμων αναφέρεται και ως *τελεστής* MIN, ο οποίος εν προκειμένω εφαρμόζεται επί των $\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_m$.

Οι Fiat, Rabani και Ravid [FiRR90] παρατήρησαν ότι ο τελεστής MIN ταυτίζεται ουσιαστικά με τη λύση του *προβλήματος των m μονοπατιών* (m *cow paths problem*), το οποίο είχε εξεταστεί στα [BaeYCR88] και [PY89]. Πρόκειται για γενίκευση του προβλήματος της γέφυρας (βλέπε Υποενότητα 1.2.2) και ορίζεται ως εξής. Από κάποιο σημείο O ξεκινούν m μονοπάτια, καθένα με άγνωστο αλλά πεπερασμένο μήκος. Στόχος είναι να φθάσουμε στο τέλος ενός από αυτά, διανύοντας συνολικά το πολύ ένα σταθερό πολλαπλάσιο του μήκους του μικρότερου μονοπατιού.

Ακολούθως θα δείξουμε πώς λειτουργεί ο MIN στην περίπτωση των μετρικών συστημάτων εργασιών. Έστω s η αρχική κατάσταση για όλους τους αλγορίθμους και $\sigma = \langle T^1, T^2, \ldots, T^N \rangle$ μια ακολουθία εργασιών. Συμβολίζουμε σ_j το αρχικό τμήμα της σ με μήκος j. Επιπλέον, $\text{κόστος}_{\text{ALG}_i}(s, \sigma_j)$ είναι το κόστος που απαιτείται για την εκτέλεση των πρώτων j εργασιών από τον ALG_i, ξεκινώντας από την s, και $\text{κατάσταση}_{\text{ALG}_i}(s, \sigma_j)$ η κατάσταση στην οποία θα βρίσκεται αμέσως μετά.

Ο αλγόριθμος $\text{ALG}^* = MIN\{\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_m\}$ μιμείται σε κάθε περίπτωση κάποιον από τους m επιμέρους αλγορίθμους. Κατά συνέπεια, μετά την εξυπηρέτηση της ακολουθίας σ_j θα καταλήξει στην $\text{κατάσταση}_{\text{ALG}_i}(s, \sigma_j)$, για κάποιο i. Αρχικά, ο ALG^* ορίζει ένα φράγμα $B = \frac{m}{m-1}$ και επιλέγει αυθαίρετα τον αλγόριθμο που θα εφαρμόσει.

Έστω τώρα ότι τη στιγμή που ανατίθεται η j-οστή εργασία ο ALG^* μιμείται τον ALG_k και άρα βρίσκεται στην $\text{κατάσταση}_{\text{ALG}_k}(s, \sigma_{j-1})$. Εν πρώτοις υπολογίζεται η ποσότητα

$$\texttt{mincost}(j) \triangleq \min_{1 \le i \le m} \left\{ \text{κόστος}_{\text{ALG}_i}(s, \sigma_j) \right\} .$$

Πριν εκτελέσει την T_j, ο αλγόριθμος συγκρίνει το $\text{κόστος}_{\text{ALG}_k}(s, \sigma_j)$ με το γινόμενο $B \cdot \texttt{mincost}(j)$. Αν διαπιστώσει ότι το πρώτο δεν υπερβαίνει το δεύτερο, τότε συνεχίζει κανονικά. Ειδάλλως, μεταβαίνει στην $\text{κατάσταση}_{\text{ALG}_{k'}}(s, \sigma_j)$ και εφαρμόζει στο εξής τον $\text{ALG}_{k'}$, όπου $k' = (k + 1) \bmod m$. Ύστερα από κάθε τέτοια αλλαγή, η τιμή του B ενημερώνεται θέτοντας $B \leftarrow \frac{m}{m-1} B$.

Παραθέτουμε άνευ αποδείξεως το ακόλουθο θεώρημα, το οποίο συνάγεται εκ των αποτελεσμάτων του [BaeYCR88]:

Θεώρημα 2.3. *Δεδομένων οποιωνδήποτε m αλγορίθμων για ένα μετρικό σύστημα εργασιών, έστω $\text{ALG}^* = MIN\{\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_m\}$. Για οποιαδήποτε ακολουθία εργασιών σ και αρχική κατάσταση s, ισχύει:*

$$\text{κόστος}_{\text{ALG}^*}(s, \sigma) \le (2em + 1) \cdot \min_{1 \le i \le m} \left\{ \text{κόστος}_{\text{ALG}_i}(s, \sigma) \right\} .$$

Οι Azar, Broder και Manasse [AzBM93] εφηύραν μια βελτιωμένη εκδοχή του τελεστή MIN. Ειδικότερα, αν κάθε αλγόριθμος ALG_i έχει λόγο ανταγωνιστικότητας α_i για ακολουθίες του συνόλου $\mathcal{I}_i$, τότε με εφαρμογή του τελεστή αυτού προκύπτει ένας $\left(8 \cdot \sum_{i=1}^{m} \alpha_i\right)$-ανταγωνιστικός online αλγόριθμος. Μάλιστα, οι ίδιοι αποδεικνύουν ότι κανένας αιτιοκρατικός συνδυασμός των m αυτών αλγορίθμων δεν μπορεί να έχει λόγο ανταγωνιστικότητας καλύτερο από $\sum_{i=1}^{m} \alpha_i$.

Τέλος, στην Υποενότητα 3.1.1 θα εξετάσουμε τον randomized τελεστή MIN, του οποίου η επίδοση προκύπτει ιδιαίτερα ενδιαφέρουσα.

2.2 Μέθοδοι ανάλυσης

Στα πλαίσια της ανταγωνιστικής ανάλυσης ενός online αλγορίθμου ALG που επιλύει το πρόβλημα Π τίθενται οι εξής στόχοι:

- Ο ALG να αποδειχθεί c-ανταγωνιστικός, για κάποιο επιθυμητό συντελεστή c.

- Να τεκμηριωθεί ότι υπάρχει c' ώστε ο συγκεκριμένος αλγόριθμος αποκλείεται να είναι καλύτερος από c'-ανταγωνιστικός.

- Να δειχθεί ότι υπάρχει c'' ώστε κανένας online αλγόριθμος για το Π δεν μπορεί να έχει λόγο ανταγωνιστικότητας μικρότερο από c''.

Αν c_{ALG} ο λόγος ανταγωνιστικότητας του ALG, τότε προφανώς θα έχουμε $c'' \leq c' \leq c_{\text{ALG}} \leq c$. Με άλλα λόγια, λοιπόν, επιδιώκουμε να βρούμε άνω και κάτω φράγματα του c_{ALG}. Σημειωτέον ότι η ανάλυση του προβλήματος της γέφυρας και εκείνου της ενοικίασης εξοπλισμού σκι που ήδη παρουσιάσαμε (Ενότητα 1.2) ακολουθούν τις προαναφερθείσες κατευθυντήριες γραμμές. Εντούτοις, οι μέθοδοι που χρησιμοποιήσαμε εκεί δεν είναι ιδιαίτερα αντιπροσωπευτικές ούτε γενικεύονται εύκολα, σε αντίθεση με αυτές που θα αναφέρουμε παρακάτω.

2.2.1 Συναρτήσεις δυναμικού

Η πλέον κοινή τεχνική απόδειξης άνω φραγμάτων για τον λόγο ανταγωνιστικότητας είναι η κατασκευή μιας *συνάρτησης δυναμικού* (*potential function*), που εφαρμόζεται κυρίως σε προβλήματα με είσοδο μια χρονική ακολουθία γεγονότων $\sigma = \sigma_1, \sigma_2, \ldots$. Για να μελετήσουμε έναν online αλγόριθμο ALG με αυτόν τον τρόπο, δίνουμε ως είσοδο στον ALG και τον βέλτιστο offline αλγόριθμο OPT μια ακολουθία σ, αποτελούμενη από m στοιχεία, την οποία αμφότεροι εκτελούν ταυτόχρονα. Η συνάρτηση δυναμικού Φ αντιστοιχεί τις καταστάσεις στις οποίες βρίσκονται οι ALG και OPT την εκάστοτε στιγμή με έναν πραγματικό αριθμό.

Συγκεκριμένα, έστω Φ_i η τιμή της συνάρτησης δυναμικού όταν οι ALG και OPT έχουν επεξεργαστεί τα πρώτα i στοιχεία της ακολουθίας. Υποθέτουμε ότι το πρόβλημα που εξετάζουμε αφορά ελαχιστοποίηση κόστους, αφού εξάλλου η περίπτωση της μεγιστοποίησης κέρδους είναι εντελώς ανάλογη. Συμβολίζουμε λοιπόν με t_i και o_i τα κόστη που υφίστανται οι δύο αλγόριθμοι, αντίστοιχα, κατά το i-οστό βήμα της λειτουργίας τους. Το *κόστος μετ' απόσβεσης* (*amortized cost*) a_i του ALG για το βήμα αυτό ορίζεται ως $a_i = t_i + \Phi_i - \Phi_{i-1}$.

Επομένως, ένα άνω φράγμα του συνολικού κόστους του ALG για την σ μπορεί να αποδειχθεί φράσσοντας το κόστος μετ' απόσβεσης σε κάθε βήμα. Έστω ότι βρίσκουμε μια συνάρτηση δυναμικού, τέτοια ώστε για κάθε i με $1 \leq i \leq m$ να ισχύει $a_i \leq c \cdot o_i$ και επιπλέον να υπάρχει σταθερά b ανεξάρτητη της εισόδου ώστε $\Phi_i \geq b$. Τότε:

$$\text{κόστος}_{\text{ALG}}(\sigma) = \sum_{i=1}^{m} t_i = (\Phi_0 - \Phi_m) + \left(\Phi_m - \Phi_0 + \sum_{i=1}^{m} t_i \right) =$$

$$
\begin{aligned}
&= (\Phi_0 - \Phi_m) + \sum_{i=1}^{m} (t_i + \Phi_i - \Phi_{i-1}) = \\
&= (\Phi_0 - \Phi_m) + \sum_{i=1}^{m} a_i \leq \\
&\leq (\Phi_0 - \Phi_m) + c \sum_{i=1}^{m} o_i \leq \\
&\leq (\Phi_0 - b) + c \cdot \text{κόστος}_{\text{OPT}}(\sigma)\,, \qquad (2.3)
\end{aligned}
$$

όπου Φ_0 είναι σταθερά που εξαρτάται μόνο από τις αρχικές καταστάσεις των ALG και OPT.

Η χρήση συνάρτησης δυναμικού για τη μελέτη μιας ακολουθίας ενεργειών ονομάζεται *ανάλυση απόσβεσης* (*amortized analysis*) [Tar85], επειδή το κόστος κάθε λειτουργίας δεν λογίζεται αφ' εαυτού, αλλά λαμβάνεται ο αριθμητικός μέσος όρος του κόστους επί όλων των λειτουργιών της ακολουθίας. Εν προκειμένω, τούτο είναι χρήσιμο διότι ενδέχεται μια μεμονωμένη ενέργεια του ALG να προκύψει εξαιρετικά «ακριβή» σε σχέση με την αντίστοιχη του OPT, ενώ εμείς ενδιαφερόμαστε να δείξουμε ότι το συνολικό online κόστος είναι το πολύ κάποιο σταθερό πολλαπλάσιο του συνολικού βελτίστου κόστους.

Φυσικά, η εύρεση μιας κατάλληλης συνάρτησης δυναμικού αποτελεί τον ακρογωνιαίο λίθο της μεθόδου αυτής. Κατά κανόνα, η συνάρτηση δυναμικού ποσοτικοποιεί τη διαφορά μεταξύ της εκάστοτε κατάστασης του online και του offline αλγορίθμου. Δύο τέτοια παραδείγματα θα εξετάσουμε ακολούθως.

Επανερχόμαστε στο πρόβλημα ενημέρωσης λίστας, που ορίσαμε στην Ενότητα 1.3. Ο αλγόριθμος MTF (Move-To-Front) σε κάθε βήμα μετακινεί το στοιχείο που μόλις προσπελάστηκε στην αρχή της λίστας, όπως φανερώνει και το όνομά του. Με τις κινήσεις αυτές επιδιώκει να εξασφαλίσει ότι αν υποστεί υψηλό κόστος για την ικανοποίηση μιας ακολουθίας αιτήσεων, ανάλογα αυξημένο θα είναι και το βέλτιστο κόστος. Πράγματι, θα αποδείξουμε ότι είναι 2-ανταγωνιστικός.

Κατ' αρχάς, έστω ℓ_i^{MTF} και ℓ_i^{OPT} οι λίστες των MTF και OPT αντίστοιχα, όπως έχουν διαμορφωθεί μετά την i-οστή αίτηση. Ορίζουμε τη συνάρτηση δυναμικού Φ_i ως το πλήθος των *αναστροφών* μεταξύ των ℓ_i^{MTF} και ℓ_i^{OPT}. Αναστροφή θεωρείται ένα διατεταγμένο ζεύγος στοιχείων (x_j, x_k), όπου το x_j βρίσκεται πριν το x_k στην ℓ_i^{MTF}, αλλά μετά από τούτο στην ℓ_i^{OPT}. Λόγου χάρη, η λίστα $\langle x_1, x_2, \ldots, x_m \rangle$ εμφανίζει $\binom{m}{2}$ αντιστροφές σε αντιπαραβολή με την $\langle x_m, x_{m-1}, \ldots, x_1 \rangle$.

Υποθέτουμε λοιπόν ότι η i-οστή αίτηση αφορά ανάγνωση του στοιχείου x. Τη στιγμή που αυτή λαμβάνεται, οι ℓ_{i-1}^{MTF} και ℓ_{i-1}^{OPT} έχουν ως φαίνεται στο

$$\ell_{i-1}^{\text{MTF}} : \ \text{αρχή} \rightarrow \begin{array}{|c|c|c|}\hline \quad A \quad & x & \qquad B \qquad \\ \hline \end{array}$$

$$\ell_{i-1}^{\text{OPT}} : \ \text{αρχή} \rightarrow \begin{array}{|c|c|c|}\hline \qquad C \qquad & x & \ D \ \\ \hline \end{array}$$

Σχήμα 2.1: Οι λίστες των MTF και OPT, αμέσως προτού ζητηθεί το στοιχείο x.

Σχήμα 2.1, όπου A, B, C και D σύνολα στοιχείων (πιθανώς κενά). Για να προσπελάσει το x, ο MTF υφίσταται κόστος $t_i = |A|$, ενώ αντίστοιχα ο OPT $o_i = |C|$. Επιπλέον, αφού το x μετακινείται στην αρχή από τον MTF, δημιουργούνται $|A \cap C|$ νέες αναστροφές ενώ αίρονται $|A \cap D|$ άλλες. Συνεπώς, θα έχουμε:

$$\Phi_i - \Phi_{i-1} \leq |A \cap C| - |A \cap D| \,. \tag{2.4}$$

Η ανωτέρω σχέση θα ισχύει ως ισότητα αν ο OPT αφήσει το x στην προηγούμενη θέση του και ως ανισότητα αν το μεταφέρει πλησιέστερα στην αρχή της λίστας. Αληθεύει ακόμη για εισαγωγή ενός νέου στοιχείου ή απόπειρα προσπέλασης ενός στοιχείου που δεν υπάρχει στη λίστα. Σε τέτοιες περιπτώσεις αρκεί να θεωρήσουμε ότι το εν λόγω στοιχείο υπάρχει ήδη και βρίσκεται αμέσως μετά το (πραγματικό) τέλος της λίστας. Εν τέλει, αν η i-οστή αίτηση αφορά διαγραφή του x, τότε δεν προκύπτουν νέες αναστροφές, οπότε $\Phi_i - \Phi_{i-1} = -|A \cap D|$ και άρα η (2.4) ισχύει πάλι ως ανισότητα.

Από τα προαναφερθέντα συμπεραίνουμε ότι:

$$\begin{aligned} |A| = |A \cap C| + |A \cap D| &\iff |A| + |A \cap C| = 2|A \cap C| + |A \cap D| \implies \\ &\implies |A| + |A \cap C| \leq 2|C| + |A \cap D| \iff \\ &\iff |A| + |A \cap C| - |A \cap D| \leq 2|C| \implies \\ &\overset{(2.4)}{\implies} t_i + \Phi_i - \Phi_{i-1} \leq 2o_i \,. \end{aligned} \tag{2.5}$$

Συνδυάζοντας τις (2.3) και (2.5) προκύπτει η 2-ανταγωνιστικότητα του MTF. Αν επιπροσθέτως κάνουμε την (λογική) υπόθεση ότι οι αρχικές λίστες των δύο αλγορίθμων είναι ακριβώς ίδιες, δηλαδή $\ell_0^{\text{MTF}} \equiv \ell_0^{\text{OPT}}$, τότε $\Phi_0 = 0$ και έτσι αποδεικνύεται ότι:

Θεώρημα 2.4. [ST85a] *Ο* MTF *είναι αυστηρά 2-ανταγωνιστικός.*

Παρατήρηση. Το παραπάνω αποτέλεσμα έχει ιστορική σημασία, καθ' ότι συνιστά το πρώτο δείγμα εφαρμογής της μεθόδου της συνάρτησης δυναμικού.

Γνωρίζουμε ήδη ότι ο αλγόριθμος LRU είναι k-ανταγωνιστικός για το πρόβλημα σελιδοποίησης μνήμης, αφού πρόκειται για αλγόριθμο σήμανσης. Αν επιθυμούμε να αποδείξουμε το ίδιο με χρήση συνάρτησης δυναμικού, πρέπει πρώτα να ορίσουμε αυστηρά τη λειτουργία του LRU.

Έστω λοιπόν ότι σε κάθε σελίδα p της αργής μνήμης αντιστοιχεί μια μεταβλητή $a[p]$, η οποία αποτελεί ένα μέτρο του πόσο πρόσφατα προσπελάστηκε η p. Ισχύει $a[p] = 0$ αν και μόνο αν η p δεν περιέχεται εκείνη τη στιγμή στη γρήγορη μνήμη. Αν ζητηθεί κάποια σελίδα p', τότε για κάθε σελίδα p με $a[p] > a[p']$ ο αλγόριθμος θέτει $a[p] \leftarrow a[p] - 1$, ενώ ταυτόχρονα $a[p'] \leftarrow k$. Σε περίπτωση που η γρήγορη μνήμη είναι γεμάτη και δεν περιέχει την p', τότε (και μόνον τότε) θα υπάρχει ακριβώς μια σελίδα p'' της οποίας η μεταβλητή $a[p'']$ μηδενίζεται. Κατά συνέπεια, η p'' απομακρύνεται από τη γρήγορη μνήμη και στη θέση της αποθηκεύεται η p'. Ειδάλλως, δεν χρειάζεται να γίνει καμία αντικατάσταση, είτε επειδή η p' βρίσκεται ήδη στη γρήγορη μνήμη, είτε διότι η τελευταία διαθέτει ελεύθερο χώρο.

Το σύνολο των μεταβλητών $\{a[p]\}$ επαρκεί για να καθορίσει πλήρως την κατάσταση στην οποία βρίσκεται ο LRU. Έστω λοιπόν S το σύνολο των σελίδων που έχει ο βέλτιστος αλγόριθμος OPT στη δική του γρήγορη μνήμη. Ορίζουμε τη συνάρτηση δυναμικού ως

$$\Phi \triangleq \sum_{p \in S} (k - a[p]) \, .$$

Αφού $a[p] \le k$ για κάθε p, έπεται ότι $\Phi \ge 0$. Επιπλέον, υποθέτουμε ότι αρχικά οι γρήγορες μνήμες των δύο αλγορίθμων είναι κενές και άρα $\Phi_0 = 0$. Ως συνήθως, συμβολίζουμε με t_i και o_i το κόστος εξυπηρέτησης της i-οστής αίτησης από τον LRU και τον OPT, αντίστοιχα. Το ζητούμενο αποτέλεσμα προκύπτει από το ακόλουθο λήμμα:

Λήμμα 2.5. [KarlMRS88] *Για κάθε $i > 0$ ισχύει*

$$t_i + \Phi_i - \Phi_{i-1} \le k \cdot o_i \, .$$

Απόδειξη. Θεωρούμε ότι κάθε αίτηση εξυπηρετείται σε δύο στάδια: πρώτα ικανοποιείται από τον OPT και κατόπιν από τον LRU. Υποθέτουμε ότι στην i-οστή αίτηση ζητείται η σελίδα p· θα δείξουμε ότι το λήμμα αληθεύει *για κάθε στάδιο ξεχωριστά.*

Στο πρώτο στάδιο ο LRU δεν υφίσταται κόστος, αφού δεν εκτελεί καμία λειτουργία. Άρα, η μεταβολή της συνάρτησης δυναμικού κατά το στάδιο αυτό πρέπει να είναι το πολύ ίση με $k \cdot o_i$. Πράγματι, αν η p βρίσκεται ήδη στη γρήγορη μνήμη του OPT, τότε $o_i = 0$ και η τιμή της συνάρτησης δυναμικού δεν αλλάζει. Σε αντίθετη περίπτωση, ο offline αλγόριθμος θα ανακτήσει την p από την αργή μνήμη με κόστος $o_i = 1$ και θα την αποθηκεύσει στη γρήγορη, απομακρύνοντας πιθανόν κάποια άλλη σελίδα. Επομένως, η μεταβολή στη συνάρτηση δυναμικού δεν υπερβαίνει το $k - a[p] \le k = k \cdot o_i$.

Στο δεύτερο στάδιο, είναι η σειρά του OPT να παραμείνει άεργος. Τώρα, το άθροισμα του t_i και της μεταβολής της συνάρτησης δυναμικού πρέπει να είναι μη θετικό. Αν η p υπάρχει ήδη στη γρήγορη μνήμη του LRU, τότε $t_i = 0$. Έστω ότι πριν την ικανοποίηση της αίτησης ισχύει $a[p] = j > 0$ και θεωρούμε το σύνολο $X = \{p' \mid a[p'] > j\}$. Από τον ορισμό του αλγορίθμου προκύπτει ότι $|X| \leq k - j$. Γνωρίζουμε ακόμη ότι $p \in S$, άρα η συνάρτηση δυναμικού θα μειωθεί κατά $k - j$ επειδή $a[p] \leftarrow k$. Όμως, για κάθε $p' \in X$ έχουμε $a[p'] \leftarrow a[p'] - 1$. Κατά συνέπεια, η μεταβολή της συνάρτησης δυναμικού ισούται με $-(k-j) + 1 \cdot |X \cap S| \leq -(k-j) + |X| = 0$.

Αντιθέτως, αν ο LRU αναγκαστεί να μεταφέρει την p από την αργή μνήμη, τότε φυσικά $t_i = 1$. Συμβολίζουμε $X = \{p' \mid a[p'] > 0\}$, οπότε $|X| = k$. Εντούτοις, $|X \cap S| \leq k - 1$, διότι $p \notin X$ και $p \in S$. Εφόσον $a[p] \leftarrow k$ και $a[p'] \leftarrow a[p'] - 1$, για κάθε $p' \in X$, η συνάρτηση δυναμικού θα μεταβληθεί κατά $-k + 1 \cdot |X \cap S| \leq -k + (k-1) = -1$.

Από όλα τα ανωτέρω διαπιστώνουμε ότι το υπ' όψη λήμμα ισχύει σε κάθε περίπτωση. □

Η απόδειξη που μόλις παρουσιάσαμε ακολουθεί μια ελαφρώς διαφορετική μεθοδολογία σε σχέση με την προηγούμενη, η οποία αναφερόταν στον αλγόριθμο MTF. Ειδικότερα, χρησιμοποιήθηκε η επονομαζόμενη τεχνική των *εναλλασσόμενων κινήσεων* (*interleaving moves* [BorEY98]). Σύμφωνα με αυτή, για να αποδειχθεί η c-ανταγωνιστικότητα ενός online αλγορίθμου ALG, αρκεί να βρεθεί συνάρτηση δυναμικού Φ τέτοια ώστε για κάθε δυνατή ακολουθία γεγονότων:

1. Αν μόνο ο βέλτιστος offline αλγόριθμος OPT προβαίνει σε κάποια ενέργεια κατά τη διάρκεια του i-οστού γεγονότος και υφίσταται κόστος x, τότε $\Delta\Phi = \Phi_i - \Phi_{i-1} \leq cx$, δηλαδή η συνάρτηση δυναμικού αυξάνει κατά cx το πολύ.

2. Αν μόνο ο ALG προβαίνει σε κάποια ενέργεια κατά τη διάρκεια του i-οστού γεγονότος και υφίσταται κόστος x, τότε $\Delta\Phi = \Phi_i - \Phi_{i-1} \leq -x$, δηλαδή η συνάρτηση δυναμικού ελαττώνεται κατά τουλάχιστον x.

3. Υπάρχει σταθερά b ανεξάρτητη της ακολουθίας γεγονότων ώστε $\Phi_i \geq b$, για κάθε i.

Παρατήρηση. Αξίζει να σημειωθεί ότι η διαμέριση της (ταυτόχρονης) λειτουργίας των ALG και OPT σε γεγονότα απαιτεί κάποια προσοχή, προκειμένου να είναι εφικτή η εφαρμογή της εν λόγω μεθόδου. Λόγου χάρη, στην απόδειξη του Λήμματος 2.5 θεωρήσαμε ότι κάθε αίτηση εξυπηρετείται πρώτα από τον OPT και ύστερα από τον LRU, δηλαδή καθορίσαμε ρητά τη σειρά με την οποία λαμβάνουν χώρα τα δύο αυτά γεγονότα. Αν υποθέταμε ακριβώς το αντίστροφο, δεν θα μπορούσαμε να ικανοποιήσουμε τις συνθήκες 1 και 2 που προαναφέραμε. Βέβαια, και πάλι θα ήταν δυνατό να αποδειχθεί το ζητούμενο, αλλά με λιγότερο κομψό τρόπο.

Κατασκευή συναρτήσεων δυναμικού

Όπως ίσως να υποψιαζόταν κανείς, η εύρεση χρήσιμων συναρτήσεων δυναμικού αποτελεί συνήθως μια δύσκολη υπόθεση. Επομένως, μάλλον προκαλεί έκπληξη το γεγονός ότι για σημαντικές κατηγορίες προβλημάτων γνωρίζουμε συστηματικούς τρόπους κατασκευής τους, που συνεπάγεται επίσης ότι η ύπαρξή τους είναι εξασφαλισμένη. Επί παραδείγματι, στη συνέχεια θα αναλύσουμε μια τεχνική η οποία εφαρμόζεται σε μετρικά συστήματα εργασιών και προτάθηκε από τον Daniel D. Sleator.

Θεωρούμε ένα οποιοδήποτε μετρικό σύστημα εργασιών $(\mathcal{M}, \mathcal{R})$ και κατασκευάζουμε το γράφο G ως εξής. Κάθε κορυφή αντιστοιχεί σε ένα ζεύγος $(s_{\text{ALG}}, s_{\text{OPT}})$, όπου s_{ALG} είναι μια κατάσταση του online αλγορίθμου ALG (η οποία φυσικά περιλαμβάνει και τις εσωτερικές μεταβλητές του τελευταίου) και s_{OPT} μια κατάσταση του βέλτιστου offline αλγορίθμου OPT. Υπάρχει κατευθυνόμενη ακμή e από την κορυφή (s_i, s_j) προς την $(s_k, s_\ell))$ αν και μόνο αν για κάποια εργασία $r_e \in \mathcal{R}$:

- ο ALG μετακινείται από την s_i στην s_k προτού εξυπηρετήσει την r_e, και
- ο OPT έχει τη δυνατότητα να εκτελέσει την r_e μεταβαίνοντας πρώτα από την s_j στην s_ℓ.

Έστω ότι θέλουμε να αποδείξουμε πως ο ALG είναι c-ανταγωνιστικός. Τότε, σε κάθε ακμή προσθέτουμε ένα βάρος $w(e)$ τέτοιο ώστε

$$w(e) = c \cdot \underbrace{[d(s_j, s_\ell) + r_e(s_\ell)]}_{\substack{\text{κόστος της } r_e \\ \text{για τον OPT}}} - \underbrace{[d(s_i, s_k) + r_e(s_k)]}_{\substack{\text{κόστος της } r_e \\ \text{για τον ALG}}},$$

όπου $e \equiv ((s_i, s_j), (s_k, s_\ell))$. Αν s_0 η αρχική κατάσταση αμφοτέρων των αλγορίθμων, συμβολίζουμε G' τον υπογράφο του G ο οποίος είναι *προσιτός* από την κορυφή $v_0 \equiv (s_0, s_0)$, ακολουθώντας βέβαια τη φορά των ακμών.

Για κάθε κύκλο C του G' θεωρούμε την ποσότητα

$$q(C) \triangleq \frac{\displaystyle\sum_{e \in C} w(e)}{\displaystyle\sum_{e \in C} [d(s_j, s_\ell) + r_e(s_\ell)]}, \tag{2.6}$$

όπου πάλι $e \equiv ((s_i, s_j), (s_k, s_\ell))$. Συμβολίζουμε C' τον κύκλο για τον οποίο η (2.6) ελαχιστοποιείται. Ας υποθέσουμε τώρα ότι ο G' έχει κύκλους αρνητικού βάρους, δηλαδή $q(C') < 0$. Κατά μήκος τέτοιων κύκλων ο ALG υφίσταται συνολικό κόστος μεγαλύτερο από c φορές εκείνο του OPT.

Υπάρχει λοιπόν άπειρη οικογένεια από ακολουθίες εργασιών οι οποίες κατ' αρχάς οδηγούν τους δύο αλγορίθμους σε ζεύγος καταστάσεων που αντιπροσωπεύει κορυφή του C' και κατόπιν φροντίζουν ώστε οι τελευταίοι να «εκτελούν» οσοδήποτε μεγάλο πλήθος «περιφορών» του C'. Το ότι οι αλγόριθμοι θα συμπεριφερθούν με αυτό τον τρόπο είναι εγγυημένο, αφ' ενός διότι ο ALG είναι αιτιοκρατικός, άρα προβλέψιμος, και αφ' ετέρου επειδή οι εν λόγω κινήσεις του OPT αποδεικνύονται μακροπρόθεσμα οι πιο συμφέρουσες.

Τοιουτοτρόπως, για την παραπάνω οικογένεια ακολουθιών βρίσκουμε ένα κάτω φράγμα του λόγου ανταγωνιστικότητας του online αλγορίθμου ίσο με $c - q$, που υπερβαίνει το μέγιστο επιθυμητό. Συνεπώς, αν ο ALG είναι όντως c-ανταγωνιστικός, τότε δεν υπάρχουν κύκλοι αρνητικού βάρους στον G'.

Όμως, σε έναν κατευθυνόμενο γράφο χωρίς κύκλους αρνητικού βάρους μπορούμε να ορίσουμε μια πραγματική συνάρτηση Φ επί του συνόλου των κορυφών, τέτοια ώστε αν $e \equiv (u, v)$ και

$$w'(e) = w(e) + \Phi(u) - \Phi(v)\,,$$

τότε $w'(e) \geq 0$ για κάθε ακμή e [Tar83]. Στην περίπτωση του G', μια κατάλληλη Φ μπορεί να κατασκευαστεί εύκολα ως εξής. Πρώτα θέτουμε $\Phi(v_0) \triangleq 0$· για κάθε άλλη κορυφή v του G', η $\Phi(v)$ ισούται με το ελάχιστο βάρος μονοπατιού επί του G' με αφετηρία την v_0 και προορισμό την v. Η προκύπτουσα Φ δεν είναι άλλη από τη ζητούμενη συνάρτηση δυναμικού, χάρη στην οποία τεκμηριώνεται η c-ανταγωνιστικότητα του ALG.

Αξιοποιώντας την ανωτέρω διατύπωση, είναι εφικτή η εύρεση του λόγου c με χρήση δυναμικού προγραμματισμού. Αρκεί να θεωρήσουμε ένα γραμμικό πρόγραμμα με σύνολο μεταβλητών $\{c\} \cup \{\Phi(v) \mid v \in V(G')\}$ και να απαιτήσουμε την ελαχιστοποίηση του c υπό τους περιορισμούς του συνόλου $\{w'(e) \geq 0 \mid e \in E(G')\}$. Το παραπάνω τέχνασμα έχει χρησιμοποιηθεί επιτυχώς στην απόδειξη άνω φραγμάτων για μικρού μεγέθους στιγμιότυπα του προβλήματος των k-εξυπηρετητών [KarlMMO90, LR94]. Μολαταύτα, αν επιχειρήσουμε να το εφαρμόσουμε σε πιο γενικά και ενδιαφέροντα προβλήματα, διαπιστώνουμε ότι ο γράφος G' προκύπτει απαγορευτικά μεγάλος. Επομένως, η πρακτική του αξία είναι εξαιρετικά περιορισμένη.

Έτσι λοιπόν, οι συναρτήσεις δυναμικού κατά κανόνα ανακαλύπτονται μέσω δοκιμών και αποτυχιών. Στις περισσότερες περιπτώσεις, θέτουμε κάποια διαισθητικά κριτήρια τα οποία αποτελούν μέτρο της «απόστασης», δηλαδή της διαφοράς, μεταξύ της κατάστασης του online αλγορίθμου και εκείνης του βέλτιστου offline αλγορίθμου. Αναμένουμε (ή μάλλον ελπίζουμε) ότι ένας κατάλληλος γραμμικός συνδυασμός των κριτηρίων αυτών θα παράξει την προσδοκώμενη συνάρτηση δυναμικού.

Μάλιστα, οι Bern, Greene και Ragunathan [BerGR93] αυτοματοποίησαν μέρος της διαδικασίας αναζήτησης συναναρτήσεων δυναμικού για προβλήματα μοιραζόμενης λανθάνουσας μνήμης. Ήτοι, αφ' ότου καθόρισαν τις συνιστώσες από τις οποίες έκριναν ότι θα έπρεπε να αποτελείται η Φ, χρησιμοποίησαν γραμμικό προγραμματισμό για την εύρεση των βελτίστων συντελεστών ώστε να επιτευχθεί ο ιδανικός λόγος ανταγωνιστικότητας.

2.2.2 Κάτω φράγματα

Για την απόδειξη κάτω φραγμάτων του λόγου ανταγωνιστικότητας, είναι συνήθως βολικό να θεωρούμε μία κακόβουλη φανταστική οντότητα, τον *αντίπαλο* (*adversary*), ο οποίος αναμετράται με τον υπό εξέταση online αλγόριθμο ALG που επιλύει το πρόβλημα Π. Συγκεκριμένα, ο αντίπαλος έχει τη δυνατότητα να κατασκευάζει στιγμιότυπα του Π και να υπολογίζει τη βέλτιστη λύση τους – με offline τρόπο φυσικά, αφού γνωρίζει όλη την εκάστοτε είσοδο! Επιπλέον, ξέρει επακριβώς πώς λειτουργεί ο ALG. Στόχος του, λοιπόν, είναι να δημιουργήσει ένα στιγμιότυπο στο οποίο ο online αλγόριθμος θα παρουσιάσει τη χειρότερη δυνατή επίδοση ως προς την ανταγωνιστική ανάλυση.

Συχνά αποδίδουμε προσδιορισμούς στον αντίπαλο, ανάλογα με τη στρατηγική που ακολουθεί. Ιδιαίτερο ενδιαφέρον παρουσιάζει ο λεγόμενος *απηνής αντίπαλος* (*cruel adversary*). Πρόκειται για μια έννοια που δεν ορίζεται αυστηρά, όπως και άλλες που έχουμε ήδη συναντήσει. Χονδρικά όμως, ένας τέτοιος αντίπαλος επιδιώκει σε κάθε βήμα να αυξάνει το κόστος (ή να μειώνει το κέρδος) του online αλγορίθμου, άνευ αρνητικού αντικτύπου στον offline αλγόριθμο, όπου αυτό είναι εφικτό. Η συμπεριφορά του απηνή αντιπάλου αναδεικνύεται εναργέστερα στην απόδειξη του παρακάτω θεωρήματος.

Θεώρημα 2.6. [ST85a] *Έστω* ALG *οποιοσδήποτε αιτιοκρατικός online αλγόριθμος για το πρόβλημα σελιδοποίησης μνήμης. Ο λόγος ανταγωνιστικότητας του* ALG *αποκλείεται να είναι μικρότερος από k.*

Απόδειξη. Υποθέτουμε ότι αρχικά ο ALG και ο βέλτιστος offline αλγόριθμος OPT έχουν στη γρήγορη μνήμη το ίδιο σύνολο σελίδων A, με $|A| = k$. Ο απηνής αντίπαλος αυτοπεριορίζεται, περιλαμβάνοντας στην ακολουθία εισόδου μόνο αιτήσεις που αφορούν σελίδες του συνόλου $A \cup \{p\}$, όπου $p \notin A$ μια αυθαίρετη σελίδα. Φροντίζει όμως ώστε κάθε φορά να ζητά κάποια σελίδα που δεν βρίσκεται εκείνη τη στιγμή στη γρήγορη μνήμη του ALG, αφού εξάλλου πάντοτε υπάρχει τουλάχιστον μία τέτοια. Η ακολουθία σ που σχηματίζεται μπορεί να έχει οσοδήποτε μεγάλο μήκος και ισχύει $\text{κόστος}_{\text{ALG}}(\sigma) = |\sigma|$.

Απομένει να δείξουμε ότι:

$$\text{κόστος}_{\text{OPT}}(\sigma) \leq \left\lceil \frac{|\sigma|}{k} \right\rceil .$$

Υπενθυμίζεται ότι η γρήγορη μνήμη του offline αλγορίθμου είναι διαρκώς πλήρης. Άρα, κάθε φορά που ανακτά μια σελίδα p' από την αργή μνήμη υποχρεούται να απομακρύνει κάποια άλλη. Ο OPT αντικαθιστά εκείνη τη σελίδα p'' στην οποία δεν πρόκειται να γίνει αναφορά για το μεγαλύτερο διάστημα στο μέλλον.[2] Ο αλγόριθμος θα χρειαστεί να προσπελάσει την αργή μνήμη ξανά μόνο όταν ζητηθεί η p''. Όμως, προτού συμβεί αυτό θα μεσολαβήσουν αιτήσεις για όλες τις $k-1$ υπόλοιπες σελίδες που παραμένουν στη γρήγορη μνήμη και ενδεχομένως κάποιες για τη νέα σελίδα p'. Συνεπώς, αν μια αίτηση αναγκάσει τον OPT να καταβάλει κόστος 1, οι επόμενες $k-1$ (τουλάχιστον) θα ικανοποιηθούν με μηδενικό κόστος. □

Είναι αξιοπρόσεκτο το ότι ο αντίπαλος διατηρεί το βέλτιστο offline κόστος σε χαμηλά επίπεδα διατυπώνοντας αιτήσεις για μόλις $k+1$ διαφορετικές σελίδες. Από την άλλη μεριά, όπως έχουμε ήδη δει, υπάρχουν k-ανταγωνιστικοί αλγόριθμοι για το πρόβλημα σελιδοποίησης, οι οποίοι είναι επομένως οι καλύτεροι δυνατοί σύμφωνα με την ανάλυση αυτή.

Στην ανωτέρω απόδειξη βοήθησε καθοριστικά το γεγονός ότι η στρατηγική που εφαρμόζει ο OPT μπορεί να περιγραφεί πολύ απλά. Έτσι, η εύρεση του offline κόστους κατέστη σχετικά εύκολη, πράγμα που δεν συμβαίνει συνήθως. Μάλιστα, πολλά online προβλήματα που έχουν μελετηθεί είναι NP-δύσκολα, δηλαδή ο υπολογισμός της βέλτιστης λύσης τους (με τις μέχρι σήμερα γνώσεις μας) συνιστά μια ιδιαίτερα πολύπλοκη διαδικασία. Ευτυχώς, δεν είναι οπωσδήποτε αναγκαίο να προσδιορίσουμε το ελάχιστο δυνατό offline κόστος, αλλά αρκεί να βρούμε ένα κατάλληλο άνω φράγμα αυτού.

Ένας τρόπος για να επιτύχουμε κάτι τέτοιο είναι να λάβουμε το μέσο όρο του κόστους μιας συλλογής αλγορίθμων. Ήτοι, αναζητούμε ένα σύνολο $\{\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_m\}$ αλγορίθμων, τέτοιων ώστε όταν ένας απηνής αντίπαλος κατασκευάζει μια ακολουθία σ με σκοπό να μεγιστοποιήσει το κόστος ενός online αλγορίθμου ALG, το άθροισμα από τα κόστη των m αλγορίθμων για είσοδο σ δεν υπερβαίνει ένα σταθερό πολλαπλάσιο του αντίστοιχου κόστους του ALG. Με άλλα λόγια:

$$\sum_{i=1}^{m} \text{κόστος}_{\text{ALG}_i}(\sigma) \leq c \cdot \text{κόστος}_{\text{ALG}}(\sigma) ,$$

για κάποιο $c > 0$.

[2]Παράβαλε με τη λειτουργία του online αλγορίθμου LRU.

Εύλογα, κάποιος από τους m αλγορίθμους έχει κόστος μικρότερο ή ίσο του μέσου όρου του κόστους όλων. Επειδή εξ ορισμού η επίδοση του βελτίστου αλγορίθμου είναι τουλάχιστον εξίσου καλή με εκείνη οποιουδήποτε εκ των $\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_m$, ο λόγος ανταγωνιστικότητας του ALG φράσσεται κάτω από το $\frac{m}{c}$.

Όπως έχουμε δει, το πρόβλημα των k-εξυπηρετητών σε ομοιόμορφο μετρικό χώρο είναι ισοδύναμο με το πρόβλημα σελιδοποίησης, για το οποίο δείξαμε ότι ο λόγος ανταγωνιστικότητας κάθε online αλγορίθμου είναι $\geq k$. Χρησιμοποιώντας την τεχνική που μόλις παρουσιάσαμε, θα γενικεύσουμε το αποτέλεσμα αυτό για οποιοδήποτε μετρικό χώρο.

Θεώρημα 2.7. [MMS88] *Έστω* ALG *οποιοσδήποτε αιτιοκρατικός online αλγόριθμος για το πρόβλημα των k-εξυπηρετητών, σε μετρικό χώρο $\mathcal{M}$ με $k+1$ ή περισσότερα σημεία. Ο λόγος ανταγωνιστικότητας του* ALG *αποκλείεται να είναι μικρότερος από k.*

Απόδειξη. Ο αντίπαλος επιλέγει ακριβώς $k+1$ σημεία: εκείνα στα οποία αρχικά βρίσκονται οι k εξυπηρετητές, τα οποία συμβολίζουμε $p_1, p_2, \ldots, p_k$ και ένα ακόμη, έστω p_{k+1}. Οι αιτήσεις του αντιπάλου αφορούν αποκλειστικά σημεία του συνόλου $\{p_1, p_2, \ldots, p_k, p_{k+1}\}$. Όντας απηνής, ο αντίπαλος πάντοτε ζητά το εκάστοτε σημείο όπου ο ALG δεν έχει εξυπηρετητή. Έτσι δημιουργεί ακολουθία αιτήσεων μήκους N, το οποίο μπορεί να είναι απεριόριστα μεγάλο. Αν συμβολίσουμε $\sigma = p_{r_1}, p_{r_2}, \ldots, p_{r_N}$ την εν λόγω ακολουθία και c_ℓ το κόστος ικανοποίησης της ℓ-οστής αίτησης, τότε για $\ell = 1, 2, \ldots, N-1$ ισχύει $c_\ell = d\big(p_{r_\ell}, p_{r_{\ell+1}}\big)$. Θα έχουμε επομένως:

$$\text{κόστος}_{\text{ALG}}(\sigma) = \sum_{\ell=1}^{N-1} d\big(p_{r_\ell}, p_{r_{\ell+1}}\big) + c_N \,.$$

Για να φράξουμε άνω το βέλτιστο κόστος ικανοποίησης της σ, θα κατασκευάσουμε k αλγορίθμους $\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_k$, τέτοιους ώστε

$$\text{κόστος}_{\text{ALG}}(\sigma) \geq \sum_{i=1}^{k} \text{κόστος}_{\text{ALG}_i}(\sigma) + b \,, \tag{2.7}$$

όπου b σταθερά. Οι αλγόριθμοι αυτοί λειτουργούν ως εξής. Η πρώτη αίτηση της σ αφορά προφανώς το σημείο p_{k+1}. Τότε, για όλα τα i, ο ALG_i ικανοποιεί την αίτηση μετακινώντας τον εξυπηρετητή που βρίσκεται στο σημείο p_i.

Φροντίζουμε ώστε για το υπόλοιπο της ακολουθίας να ισχύουν οι παρακάτω αναλλοίωτες ιδιότητες:

- Κάθε αλγόριθμος ALG_i έχει έναν εξυπηρετητή στο σημείο που ζητήθηκε πιο πρόσφατα.

- Ανά πάσα στιγμή, αν $i \neq j$, τότε το σημείο στο οποίο δεν υπάρχει εξυπηρετητής του ALG_i διαφέρει από το αντίστοιχο για τον ALG_j.

Έστω λοιπόν ότι φθάνει αίτηση για το σημείο p_{r_ℓ}. Ακριβώς ένας από τους k αλγορίθμους μετακινεί τον εξυπηρετητή που βρίσκεται στο σημείο $p_{r_{\ell-1}}$ ώστε να την ικανοποιήσει, ενώ οι υπόλοιποι δεν είναι υποχρεωμένοι να κάνουν κάποια ενέργεια. Εύκολα διαπιστώνουμε ότι τοιουτοτρόπως αμφότερες οι αναλλοίωτες διατηρούνται.

Το συνολικό κόστος των $\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_k$ ισούται με

$$\begin{aligned}\sum_{i=1}^{k} \text{κόστος}_{\text{ALG}_i}(\sigma) &= \sum_{i=1}^{k} d(p_i, p_{k+1}) + \sum_{\ell=1}^{N-1} d\big(p_{r_\ell}, p_{r_{\ell+1}}\big) \\ &= \sum_{i=1}^{k} d(p_i, p_{k+1}) + \text{κόστος}_{\text{ALG}}(\sigma) - c_N\,.\end{aligned}$$

Όμως $c_N > 0$, άρα η (2.7) αληθεύει για $b = -\sum_{i=1}^{k} d(p_i, p_{k+1})$, που είναι ανεξάρτητο της σ. Επιπλέον, $\text{κόστος}_{\text{OPT}}(\sigma) \leq \text{κόστος}_{\text{ALG}_i}(\sigma)$, για κάθε i. Κατά συνέπεια:

$$k \cdot \text{κόστος}_{\text{OPT}}(\sigma) \leq k \cdot \min_i \big\{\text{κόστος}_{\text{ALG}_i}(\sigma)\big\} \leq \sum_{i=1}^{k} \text{κόστος}_{\text{ALG}_i}(\sigma)\,,$$

οπότε

$$\text{κόστος}_{\text{ALG}}(\sigma) \geq k \cdot \text{κόστος}_{\text{OPT}}(\sigma) + b\,.$$

Αφού ο αντίπαλος μπορεί να δημιουργήσει μια ακολουθία αιτήσεων με οσοδήποτε υψηλό συνολικό κόστος για τον ALG, το υπ' όψη θεώρημα ισχύει. $\square$

Κεφάλαιο 3

Randomized Online Αλγόριθμοι

Μέχρι τώρα, όλοι οι online αλγόριθμοι που εξετάσαμε ήταν αιτιοκρατικοί (deterministic), δηλαδή η συμπεριφορά τους για κάθε δεδομένη είσοδο ήταν μονοσήμαντα ορισμένη. Μπορούμε όμως να θεωρήσουμε και randomized αλγορίθμους, οι οποίοι έχουν τη δυνατότητα να παίρνουν αποφάσεις με τυχαίο τρόπο κατά τη λειτουργία τους. Με άλλα λόγια, υποθέτουμε ότι ρίχνουν (εικονικά) νομίσματα και ενεργούν αναλόγως των αποτελεσμάτων. Εύλογα, τα κόστη ή τα κέρδη τέτοιων αλγορίθμων είναι τυχαίες μεταβλητές, επομένως η επίδοσή τους κρίνεται από τη μέση τιμή αυτών για το εκάστοτε στιγμιότυπο εισόδου.

3.1 Βασικοί ορισμοί και παραδείγματα

Στην περίπτωση των randomized online αλγορίθμων, η έννοια του αντιπάλου είναι πιο σύνθετη. Ειδικότερα, ακολουθώντας τις διατυπώσεις των Raghavan και Snir [RS89] και Ben-David, Borodin, Karp, Tardos και Wigderson [BenDBK^{+}90] διακρίνουμε αδρομερώς διάφορους τύπους αντιπάλων, αναλόγως της γνώσεως που έχουν για τις τυχαίες αποφάσεις του αλγορίθμου και του τρόπου με τον οποίο βρίσκουν τη δική τους λύση για τα στιγμιότυπα του προβλήματος που οι ίδιοι κατασκευάζουν.

- Ένας *αδαής ή επιλήσμων* (***oblivious***) αντίπαλος επιλέγει εξ αρχής την είσοδο που θα παρουσιάσει στον online αλγόριθμο, χωρίς φυσικά να γνωρίζει την έκβαση των υποθετικών ρίψεων νομισμάτων ούτε τις αποφάσεις που λαμβάνει ο τελευταίος βάσει αυτών. Τονίζεται όμως ότι ο αντίπαλος ξέρει πώς λειτουργεί ο αλγόριθμος και συγκεκριμένα με ποια πιθανότητα ενεργεί με τον έναν ή τον άλλο τρόπο σε κάθε περίπτωση.

- Ένας *προσαρμοστικός* (*adaptive*) αντίπαλος έχει ανά πάσα στιγμή πλήρη γνώση της κατάστασης στην οποία βρίσκεται ο αλγόριθμος, άρα και των τυχαίων αποφάσεων που ελήφθησαν. Με τούτα τα δεδομένα φτιάχνει κάθε φορά το επόμενο τμήμα της εισόδου που θα τροφοδοτήσει στον αλγόριθμο, δεν είναι όμως σε θέση να προβλέψει εκ των προτέρων το αποτέλεσμα των μελλοντικών ρίψεων νομισμάτων.

Συνεπώς, οι προσαρμοστικοί αντίπαλοι δεν είναι εκ των προτέρων βέβαιοι για την τελική μορφή της εισόδου την οποία δημιουργούν σταδιακά· εξαιτίας αυτού, κατηγοριοποιούνται περαιτέρω ως εξής:

– Ένας προσαρμοστικός online αντίπαλος επιλύει με online τρόπο το στιγμιότυπο του προβλήματος. Δηλαδή, σε κάθε βήμα εξετάζει το κομμάτι του στιγμιοτύπου που έχει οριστικοποιηθεί μέχρι την εκάστοτε στιγμή και αποφασίζει ποια (μερική) λύση θα δώσει για αυτό, χωρίς μάλιστα να ξέρει ακόμα την αντίστοιχη μερική λύση που έχει βρει ο αλγόριθμος.

– Αντιθέτως, ένας προσαρμοστικός offline αντίπαλος πρώτα ολοκληρώνει την κατασκευή του στιγμιοτύπου – οπότε γνωρίζει πλέον την τελική μορφή του – και ύστερα υπολογίζει τη βέλτιστη λύση του. Όπως λοιπόν προδίδει το όνομά του, η λειτουργία του είναι ουσιαστικά offline.

Τοιουτοτρόπως, κατ' απόλυτη αναλογία με τους Ορισμούς 1.1 και 1.2, έχουμε:

Ορισμός 3.1. *Ένας randomized online αλγόριθμος* ALG *που επιλύει το πρόβλημα* Π *θα λέγεται* c-*ανταγωνιστικός εναντίον αδαών (προσαρμοστικών online, προσαρμοστικών offline) αντιπάλων αν υπάρχει σταθερά* c_0 *τέτοια ώστε για κάθε αδαή (προσαρμοστικό online, προσαρμοστικό offline) αντίπαλο* ADV *να ισχύει*

$$E[\text{κόστος}_{\text{ALG}}(I) - c \cdot \text{κόστος}_{\text{ADV}}(I)] \leq c_0$$

$$\left[\text{ ή } \quad E[\text{κέρδος}_{\text{ADV}}(I) - c \cdot \text{κέρδος}_{\text{ALG}}(I)] \leq c_0 \right],$$

όπου I *το στιγμιότυπο που κατασκευάζει ο αντίπαλος καθώς αλληλεπιδρά με τον* ALG. *Η αναμενόμενη τιμή* $E[\cdot]$ *λαμβάνεται ως προς όλες τις τυχαίες αποφάσεις του αλγορίθμου οι οποίες είναι δυνατό να ληφθούν για το συγκεκριμένο στιγμιότυπο.*

Σημειωτέον ότι ειδικά για αδαείς αντιπάλους οι παραπάνω σχέσεις απλουστεύονται ως εξής:

$$E[\text{κόστος}_{\text{ALG}}(I)] \leq c \cdot \text{κόστος}_{\text{ADV}}(I) + c_0$$

$$\left[\text{ ή } \quad \text{κέρδος}_{\text{ADV}}(I) \leq c \cdot E[\text{κέρδος}_{\text{ALG}}(I)] + c_0 \right],$$

και τούτο διότι η συμπεριφορά ενός τέτοιου αντιπάλου δεν αποτελεί συνάρτηση κανενός τυχαίου γεγονότος. Κατά συνέπεια, τόσο η είσοδος I όσο και το κόστος$_{\text{ADV}}(I)$ ή το κέρδος$_{\text{ALG}}(I)$ δεν είναι τυχαίες μεταβλητές.

Ορισμός 3.2. *Ο λόγος ανταγωνιστικότητας ενός randomized online αλγορίθμου* ALG *εναντίον αντιπάλων τύπου* ADV *είναι*

$$c_{\text{ALG}}^{\mathsf{ADV}} \triangleq \inf \left\{ c \;\middle|\; \begin{array}{l} \text{ο ALG είναι } c\text{-ανταγωνιστικός} \\ \text{εναντίον κάθε αντιπάλου τύπου } \mathsf{ADV} \end{array} \right\} .$$

Είμαστε πλέον σε θέση να αναλύσουμε randomized αλγορίθμους για online προβλήματα. Κατ' αρχάς, επανερχόμαστε στο πρόβλημα της σελιδοποίησης μνήμης και μελετούμε τον απλούστερο δυνατό randomized αλγόριθμο, τον επονομαζόμενο RAND. Όταν ο τελευταίος πρέπει να αντικαταστήσει μια σελίδα που βρίσκεται στη γρήγορη μνήμη, επιλέγει τυχαία και με ομοιόμορφη πιθανοτική κατανομή κάποια από όλες τις k σελίδες. Η ανταγωνιστικότητά του τεκμηριώνεται από το παρακάτω θεώρημα, το οποίο οφείλεται στους Raghavan και Snir.

Θεώρημα 3.3. [RS89] *Ο αλγόριθμος* RAND *είναι k-ανταγωνιστικός εναντίον προσαρμοστικών online αντιπάλων.*

Απόδειξη. Ορίζουμε συνάρτηση δυναμικού Φ ως εξής:

$$\Phi_i \triangleq k(k - \phi_i) ,$$

όπου ϕ_i το πλήθος των σελίδων που βρίσκονται στη γρήγορη μνήμη και του RAND και του αντιπάλου ADON, μετά την ικανοποίηση της i-οστής αίτησης. Έτσι, αν t_i το κόστος του RAND στο i-οστό βήμα της λειτουργίας του, τότε το αντίστοιχο κόστος μετ' απόσβεσης είναι, κατά τα γνωστά, $a_i = t_i + \Phi_i - \Phi_{i-1}$. Υπενθυμίζεται ότι τα t_i, a_i και Φ_i είναι τυχαίες μεταβλητές. Όμως, εξ ορισμού η Φ φράσσεται κάτω από το μηδέν. Επομένως, αρκεί να δείξουμε ότι $E[a_i] \leq k \cdot E[o_i]$, για κάθε i, όπου o_i το κόστος ικανοποίησης της i-οστής αίτησης από τον ADON. Ισοδύναμα,

$$E[\Phi_i - \Phi_{i-1}] \leq k \cdot E[o_i] - E[t_i] . \tag{3.1}$$

Εξετάζουμε λοιπόν την κατάσταση του αλγορίθμου και του αντιπάλου αμέσως πριν τη λήψη της i-οστής αίτησης. Συμβολίζουμε με X και Y τα σύνολα σελίδων που βρίσκονται στη γρήγορη μνήμη των RAND και ADON αντίστοιχα. Θέτουμε $Z = X \cap Y$, οπότε $|Z| = \phi_{i-1}$. Χωρίς βλάβη της γενικότητας, θεωρούμε ότι ο αντίπαλος είναι απηνής και ζητά μια σελίδα $p \notin X$. Εξάλλου, αν $p \in X$, τότε $t_i = 0$ και $\Phi_i \leq \Phi_{i-1}$, άρα η (3.1) ισχύει τετριμμένα.

Απομένουν συνεπώς τα ακόλουθα ενδεχόμενα:

1. $p \in Y$: Αν ο RAND απομακρύνει από τη γρήγορη μνήμη μια σελίδα $p' \in Z$ για να βάλει την p στη θέση της, τότε $\Phi_i = \Phi_{i-1}$. Ειδάλλως, αν $p' \notin Z$, θα ισχύει $\phi_i = \phi_{i-1} + 1$ και $\Phi_i = \Phi_{i-1} - k$. Λόγω του τρόπου λειτουργίας του αλγορίθμου, η πιθανότητα να έχουμε $p' \in Z$ είναι $\frac{\phi_{i-1}}{k}$, επομένως η αναμενόμενη μεταβολή της συνάρτησης δυναμικού ισούται με

$$E[\Phi_i - \Phi_{i-1}] = \left(1 - \frac{\phi_{i-1}}{k}\right) \cdot (-k) = \phi_{i-1} - k \leq -1 \ ,$$

αφού $p \notin X \Rightarrow p \notin Z \Rightarrow \phi_{i-1} \leq k - 1$. Επειδή $t_i = 1$ και $o_i = 0$, η (3.1) αληθεύει.

2. $p \notin Y$: Ο αντίπαλος είναι υποχρεωμένος να αντικαταστήσει μια σελίδα $p'' \in Y$ με την p. Διακρίνουμε δύο περιπτώσεις:

   (α) $p'' \notin Z$: Αν με τη σειρά του ο RAND απομακρύνει από τη γρήγορη μνήμη μια σελίδα $p' \in Z$, τότε $\phi_i = \phi_{i-1}$, ειδάλλως $\phi_i = \phi_{i-1} + 1$. Σε κάθε περίπτωση, $\Phi_i \leq \Phi_{i-1}$ και επειδή $k \cdot o_i - t_i = k \cdot 1 - 1 \geq 0$, η (3.1) ικανοποιείται.

   (β) $p'' \in Z$: Αν με τη σειρά του ο RAND απομακρύνει από τη γρήγορη μνήμη μια σελίδα $p' \in (X - Z) \cup \{p''\}$, τότε $\phi_i = \phi_{i-1}$, οπότε $\Phi_i = \Phi_{i-1}$· ειδάλλως $\phi_i = \phi_{i-1} - 1$ και $\Phi_i = \Phi_{i-1} + k$. Η πιθανότητα να επιλεγεί μια $p' \in Z - \{p''\}$ είναι $\frac{\phi_{i-1}-1}{k}$, άρα

$$\begin{aligned} E[\Phi_i - \Phi_{i-1}] = k \cdot \frac{\phi_{i-1} - 1}{k} &= \phi_{i-1} - 1 \leq \\ &\leq k - 1 = k \cdot E[o_i] - E[t_i] \ , \end{aligned}$$

   επαληθεύοντας την (3.1).

Εφόσον με τα προαναφερθέντα εξαντλήσαμε κάθε δυνατό ενδεχόμενο, η απόδειξη έχει ολοκληρωθεί. □

Εκ πρώτης όψεως, το παραπάνω αποτέλεσμα δεν είναι ιδιαίτερα ενθαρρυντικό, αφού ο RAND δεν φαίνεται να επιτυγχάνει καλύτερη επίδοση απ' ότι ένας αιτιοκρατικός αλγόριθμος. Από την άλλη, όμως, πρέπει να τονίσουμε ότι ο RAND είναι *αμνήμων* (*memoryless*), δηλαδή δεν απαιτεί επιπλέον χώρο μνήμης για τη λειτουργία του. Αντιθέτως, όλοι οι αλγόριθμοι σήμανσης, λόγου χάρη, χρειάζονται $\Omega(k)$ bits μνήμης ώστε να αποθηκεύουν το ποιες ακριβώς σελίδες είναι σεσημασμένες.

Ας μελετήσουμε τώρα έναν randomized αλγόριθμο σήμανσης που αποκαλείται MARK και ανακαλύφθηκε από τους Fiat, Karp, Luby, McGeoch, Sleator και Young [FiKL+88]. Το χαρακτηριστικό το οποίο τον διακρίνει από τους υπόλοιπους αλγορίθμους της ίδιας κατηγορίας είναι πως, όταν υποχρεωθεί να απομακρύνει μια σελίδα από τη γρήγορη μνήμη, επιλέγει τυχαία και με ομοιόμορφη πιθανοτική κατανομή κάποια εκ των μη σεσημασμένων σελίδων.

Θεώρημα 3.4. [FiKL+88] *Ο αλγόριθμος* MARK *είναι* $2\mathcal{H}_k$*-ανταγωνιστικός εναντίον αδαών αντιπάλων, όπου* $\mathcal{H}_k \triangleq \sum_{i=1}^{k} i^{-1}$ *ο* k*-οστός αρμονικός αριθμός.*

Απόδειξη. Υποθέτουμε ότι ο MARK και ο αδαής αντίπαλος OBL έχουν αρχικά τις ίδιες σελίδες στη γρήγορη μνήμη. Κατά τα γνωστά, η λειτουργία του MARK με είσοδο την ακολουθία σ που κατασκευάζει ο OBL χωρίζεται σε φάσεις (βλέπε Υποενότητα 2.1.3). Έστω X_i το σύνολο σελίδων που βρίσκονται στη γρήγορη μνήμη του αλγορίθμου αμέσως πριν αρχίσει η i-οστή φάση και Y_i το σύνολο των σελίδων οι οποίες ζητούνται στη διάρκεια της τελευταίας. Κατά τα γνωστά, $|X_i| = |Y_i| = k$. Ακόμη, θέτουμε $m_i = |Y_i - X_i|$. Ισχύει $m_i \geq 1$, διότι οπωσδήποτε η πρώτη αίτηση της φάσης αφορά σελίδα που πρέπει να ανακτηθεί από την αργή μνήμη.

Χωρίς βλάβη της γενικότητας, μπορούμε να θεωρήσουμε ότι για καθεμία από τις σελίδες του Y_i διατυπώνεται ακριβώς μια αίτηση εντός της i-οστής φάσης. Πράγματι, τυχόν επόμενες αιτήσεις για την ίδια σελίδα θα ικανοποιούντο με μηδενικό κόστος από τον MARK, αφού μετά την πρώτη προσπέλαση η σελίδα βρίσκεται οπωσδήποτε στη γρήγορη μνήμη και παραμένει εκεί τουλάχιστον ως το τέλος της φάσης.

Εύκολα διαπιστώνουμε ότι η χειρότερη περίπτωση για τον αλγόριθμο είναι να ζητηθούν πρώτα οι σελίδες του συνόλου $Y_i - X_i$ και κατόπιν οι υπόλοιπες, οι οποίες ανήκουν στο $Y_i \cap X_i$, αφού τότε θα είναι μεγιστοποιείται η πιθανότητα κάποιες από τις τελευταίες να έχουν ήδη απομακρυνθεί από τη γρήγορη μνήμη. Μάλιστα, έστω p_j η j-οστή κατά σειρά σελίδα του $Y_i \cap X_i$ που ζητείται. Συμβολίζουμε M_{ij} το σύνολο των σεσημασμένων σελίδων αμέσως πριν φθάσει η αίτηση για την p_j. Προφανώς, λοιπόν, $p_j \in X_i - M_{ij}$. Άρα, η πιθανότητα να βρίσκεται η p_j στη γρήγορη μνήμη εκείνη τη στιγμή ισούται με

$$\frac{\{(p \mid p \in X_i - M_{ij}) \wedge (p \text{ στη γρήγορη μνήμη})\}}{\{p \mid p \in X_i - M_{ij}\}} = \frac{k-(j-1)-m_i}{k-(j-1)} \;.$$

Συνεπώς, το αναμενόμενο κόστος του αλγορίθμου κατά την υπό εξέταση φάση είναι

$$\begin{aligned} m_i + \sum_{j=1}^{k-m_i} \left(1 - \frac{k-(j-1)-m_i}{k-(j-1)}\right) &= m_i + \sum_{j=1}^{k-m_i} \frac{m_i}{k-(j-1)} = \\ &= m_i + m_i(\mathcal{H}_k - \mathcal{H}_{m_i} + 1) \leq m_i \mathcal{H}_k \;, \end{aligned}$$

αφού $m_i \geq 1$. Επομένως:

$$E[\text{κόστος}_{\text{MARK}}(\sigma)] \leq \sum_i m_i \mathcal{H}_k \ .$$

Απομένει να βρούμε ένα κάτω φράγμα για το $\text{κόστος}_{\text{OBL}}(\sigma)$. Έτσι, αρκεί να παρατηρήσουμε ότι κατά τη διάρκεια της $(i-1)$-οστής και της i-οστής φάσης ζητούνται συνολικά $k+m_i$ ή περισσότερες διαφορετικές σελίδες. Δηλαδή, για κάθε $i > 1$, ο αντίπαλος υφίσταται κόστος τουλάχιστον m_i στην $(i-1)$-οστή και την i-οστή φάση μαζί. Επιπλέον, κατά την πρώτη φάση εκτελεί $\geq m_i$ προσπελάσεις της αργής μνήμης, οπότε αθροιστικά έχουμε

$$\text{κόστος}_{\text{OBL}}(\sigma) \geq \frac{1}{2} \sum_i m_i \ ,$$

που αποδεικνύει το επιθυμητό αποτέλεσμα. □

Οι Αχλιόπτας, Chrobak και Noga [AcCN96] προσδιόρισαν την ακριβή τιμή του λόγου ανταγωνιστικότητας του MARK, η οποία είναι $2\mathcal{H}_k - 1$, εφόσον όμως η αργή μνήμη περιέχει τουλάχιστον $k+2$ σελίδες. Ειδάλλως, στην ειδική περίπτωση που οι σελίδες της αργής μνήμης είναι μόνο $k+1$, εύκολα διαπιστώνουμε ότι ο MARK είναι $\mathcal{H}_k$-ανταγωνιστικός.

3.1.1 O randomized τελεστής MIN

Στην Υποενότητα 2.1.4 αναφερθήκαμε στον τελεστή MIN, ο οποίος μας έδινε τη δυνατότητα να συνδυάσουμε online αλγορίθμους με καλή επίδοση σε επιμέρους στιγμιότυπα ενός προβλήματος, ώστε να κατασκευάσουμε έναν ανταγωνιστικό αλγόριθμο για το γενικότερο πρόβλημα. Οι Fiat, Foster, Karloff, Rabani, Ravid και Vishwanathan [FiFK+91] ανακάλυψαν μια randomized παραλλαγή του MIN, την οποία βελτίωσαν λίγο αργότερα, οι Azar, Broder και Manasse. Η επίδοση της τελευταίας τεκμηριώνεται από το ακόλουθο θεώρημα:

Θεώρημα 3.5. [AzBM93] *Δίνονται οποιοιδήποτε m online αλγόριθμοι για ένα μετρικό σύστημα εργασιών. Έστω ότι ο i-οστός εξ αυτών έχει λόγο ανταγωνιστικότητας a_i εναντίον αδαών αντιπάλων σε ένα υποσύνολο $\mathcal{I}_i$ του συνόλου $\mathcal{I}$ των στιγμιοτύπων, $1 \leq i \leq m$. Αν ισχύει $\bigcup \mathcal{I}_i \subseteq \mathcal{I}$ και $\text{ALG}^* = RMIN\{\text{ALG}_1, \text{ALG}_2, \ldots, \text{ALG}_m\}$, τότε*

$$c^{\mathsf{OBL}}_{\text{ALG}^*} \leq 8f(\{a_1, a_2, \ldots, a_m\}) \ ,$$

όπου η f ορίζεται αναδρομικά για κάθε πολυσύνολο θετικών αριθμών A ως *εξής:*

$$\begin{cases} f(\emptyset) \triangleq 0 \\ f(A) \triangleq \dfrac{1+\displaystyle\sum_{x\in A}\frac{f(A-\{x\})}{x}}{\displaystyle\sum_{x\in A}\frac{1}{x}} \end{cases} .$$

Επιπλέον, οποιοσδήποτε άλλος συνδυασμός των m αυτών αλγορίθμων επιτυγχάνει λόγο ανταγωνιστικότητας όχι καλύτερο από $f(\{a_1, a_2, \ldots, a_m\})$.

Εφόσον η ανωτέρω ορισμένη f είναι άρρηκτα συνυφασμένη με την επίδοση του τελεστή $RMIN$, αξίζει να τη μελετήσουμε περαιτέρω.

Πρόταση 3.6. *Έστω m πραγματικοί αριθμοί $x_1, x_2, \ldots, x_m$, τέτοιοι ώστε $0 < x_1 \leq x_2 \leq \cdots \leq x_m$. Η συνάρτηση f έχει τις παρακάτω ιδιότητες:*

1. $\lim\limits_{x_1\to 0} f(\{x_1, x_2, \ldots, x_m\}) = f(\{x_2, \ldots, x_m\})$.
2. $f(\{x_1, x_2, \ldots, x_m\})$ *γνησίως αύξουσα ως προς κάθε* x_i.
3. $f(\{x_1, x_2, \ldots, x_m\}) > x_m$.
4. $\mathcal{H}_m \cdot x_1 \leq f(\{x_1, x_2, \ldots, x_m\}) \leq \mathcal{H}_m \cdot x_m$.
5. $f(\{x_1, x_2, \ldots, x_m\}) \geq \mathcal{H}_m \cdot \dfrac{x_1 + x_2 + \cdots + x_m}{m}$.
6. [AzBM93] $$\begin{aligned} \sum_{k=1}^{m} \frac{x_k}{k} \leq \sum_{k=1}^{m} \frac{x_k^2}{x_1 + \cdots + x_k} &\leq \\ \leq f(\{x_1, x_2, \ldots, x_m\}) &\leq \\ \leq \sum_{k=1}^{m} \frac{x_k^2}{x_k + \cdots + x_m} \leq \sum_{k=1}^{m} \frac{x_k}{m+1-k} & . \end{aligned}$$

Απόδειξη.

1. Με απλή επαγωγή ως προς m.
2. Εύκολα διαπιστώνουμε ότι η f είναι παραγωγίσιμη ως προς κάθε x_i. Επομένως, αρκεί να δειχθεί ότι για κάθε $i \in \{1, 2, \ldots, m\}$ ισχύει

$$\frac{\partial f(\{x_1, x_2, \ldots, x_m\})}{\partial x_i} > 0 . \tag{3.2}$$

Για το σκοπό αυτό, εφαρμόζουμε γενικευμένη επαγωγή ως προς m.

- Για $m = 1$, προφανώς $f(\{x_1\}) = x_1$ και $\frac{\partial f(\{x_1\})}{\partial x_1} = 1 > 0$.
- Υποθέτουμε ότι υπάρχει κάποιο $\ell \in \mathbb{N}^*$ τέτοιο ώστε για όλα τα $m \leq \ell$ να ισχύει η (3.2). Λόγω και της (1), θα είναι

  $$f(\{x_1, x_2, \ldots, x_m\}) > f(\{x_1, x_2, \ldots, x_m\} - \{x_i\}) \ ,$$

  για τις συγκεκριμένες τιμές του m και για κάθε $i \in \{1, 2, \ldots, m\}$. Τότε, θέτοντας $m = \ell + 1$ και $X = \{x_1, x_2, \ldots, x_{\ell+1}\}$ έχουμε:

  $$\frac{\partial f(X)}{\partial x_i} > 0 \iff \tag{3.3}$$

  $$\iff \left[-\frac{f(X - \{x_i\})}{x_i^2} + \sum_{\substack{j=1\\j\neq i}}^{n} \frac{\frac{\partial f(X - \{x_j\})}{\partial x_i}}{x_j} \right] \left(\sum_{j=1}^{n} \frac{1}{x_j} \right) - \\ - \left[1 + \sum_{j=1}^{n} \frac{f(X - \{x_j\})}{x_j} \right] \left(-\frac{1}{x_i^2} \right) > 0 \iff$$

  $$\iff \left[\sum_{\substack{j=1\\j\neq i}}^{n} \frac{\frac{\partial f(X - \{x_j\})}{\partial x_i}}{x_j} \right] \left(\sum_{j=1}^{n} \frac{1}{x_j} \right) + \\ + \frac{1}{x_i^2} \left[1 + \sum_{\substack{j=1\\j\neq i}}^{n} \frac{f(X - \{x_j\})}{x_j} - f(X - \{x_i\}) \cdot \sum_{\substack{j=1\\j\neq i}}^{n} \frac{1}{x_j} \right] > 0 \iff$$

  $$\iff \left[\sum_{\substack{j=1\\j\neq i}}^{n} \frac{\frac{\partial f(X - \{x_j\})}{\partial x_i}}{x_j} \right] \left(\sum_{j=1}^{n} \frac{1}{x_j} \right) + \\ + \frac{1}{x_i^2} \left[1 + \sum_{\substack{j=1\\j\neq i}}^{n} \frac{f(X - \{x_j\})}{x_j} - 1 - \sum_{\substack{j=1\\j\neq i}}^{n} \frac{f(X - \{x_i, x_j\})}{x_j} \right] > 0 \ .$$

  Η παραπάνω ανισότητα αληθεύει, αφού αμφότεροι οι προσθετέοι του αριστερού μέρους είναι γνησίως θετικοί· άρα, το ίδιο συμβαίνει και με την (3.3).

Κατά συνέπεια, η ζητούμενη ιδιότητα της f ισχύει για κάθε $m \in \mathbb{N}^*$.

3. Από τις (1) και (2), έχουμε άμεσα
$$f(\{x_1, x_2, \ldots, x_m\}) > f(\{x_2, \ldots, x_m\}) > \cdots > f(\{x_m\}) = x_m\ .$$

4. Η διπλή ανισότητα γράφεται ισοδύναμα:
$$f(\{x_1, x_1, \ldots, x_1\}) \leq f(\{x_1, x_2, \ldots, x_m\}) \leq f(\{x_m, x_m, \ldots, x_m\})\ ,$$
που αληθεύει λόγω της (2). Εναλλακτικά, μπορεί να προκύψει ως πόρισμα της (6).

5. Χρησιμοποιούμε και πάλι επαγωγή ως προς m.
   - Για $m = 1$, προφανώς ισχύει.
   - Υποθέτουμε ότι υπάρχει κάποιο $\ell \in \mathbb{N}^*$ τέτοιο ώστε για $m = \ell$ να είναι $f(\{x_1, x_2, \ldots, x_\ell\}) \geq \mathcal{H}_\ell \cdot \frac{x_1+x_2+\cdots+x_\ell}{\ell}$. Τότε, για $m = \ell + 1$ συμπεραίνουμε ότι

$$\begin{aligned}
f(\{x_1, x_2, \ldots, x_{\ell+1}\}) &= \frac{1 + \displaystyle\sum_{i=1}^{\ell+1} \frac{f(\{x_1, x_2, \ldots, x_{\ell+1}\} - \{x_i\})}{x_i}}{\displaystyle\sum_{i=1}^{\ell+1} \frac{1}{x_i}} \geq \\
&\geq \frac{1 + \displaystyle\sum_{i=1}^{\ell+1} \frac{\mathcal{H}_\ell \cdot \frac{(x_1+x_2+\cdots+x_{\ell+1})-x_i}{\ell}}{x_i}}{\displaystyle\sum_{i=1}^{\ell+1} \frac{1}{x_i}} = \\
&= \mathcal{H}_\ell \cdot \sum_{i=1}^{\ell+1} \frac{x_i}{\ell} + \frac{1 - \mathcal{H}_\ell \cdot \dfrac{\ell+1}{\ell}}{\displaystyle\sum_{i=1}^{\ell+1} \frac{1}{x_i}} \geq \\
&\overset{(*)}{\geq} \mathcal{H}_\ell \cdot \sum_{i=1}^{\ell+1} \frac{x_i}{\ell} + \frac{1 - \mathcal{H}_\ell \cdot \dfrac{\ell+1}{\ell}}{(\ell+1)^2} \cdot \sum_{i=1}^{\ell+1} x_i = \\
&= \left(\frac{\mathcal{H}_\ell}{\ell} + \frac{1}{(\ell+1)^2} - \frac{\mathcal{H}_\ell}{\ell(\ell+1)} \right) \cdot \sum_{i=1}^{\ell+1} x_i = \\
&= \frac{\mathcal{H}_{\ell+1}}{\ell+1} \cdot \sum_{i=1}^{\ell+1} x_i\ ,
\end{aligned}$$

όπου στο σημείο $(*)$ εφαρμόσαμε τη γνωστή ανισότητα Αριθμητικού Μέσου - Αρμονικού Μέσου:

$$\frac{\ell+1}{\dfrac{1}{x_1}+\dfrac{1}{x_2}+\cdots+\dfrac{1}{x_{\ell+1}}} \le \frac{x_1+x_2+\cdots+x_{\ell+1}}{\ell+1} .$$

6. Βλέπε [AzBM93]. □

Τονίζεται ότι οι ανισότητες (3, 4, 5, 6) είναι ιδιαίτερα χρήσιμες στην πράξη, διότι παρέχουν ευκόλως υπολογίσιμα άνω και κάτω φράγματα της f. Διαπιστώνουμε, λοιπόν, ότι ο randomized τελεστής MIN επιτυγχάνει πολύ ικανοποιητικότερη επίδοση, συγκρινόμενος με τον αιτιοκρατικό ομόλογό του.

3.2 Η ισχύς των αντιπάλων

Συμβολίζουμε $\mathcal{C}^{\Pi}_{\mathsf{OBL}}, \mathcal{C}^{\Pi}_{\mathsf{ADON}}, \mathcal{C}^{\Pi}_{\mathsf{ADOFF}}$ τους βελτίστους λόγους ανταγωνιστικότητας που μπορούν να επιτευχθούν με randomized online αλγόριθμο για το πρόβλημα ελαχιστοποίησης κόστους[1] Π εναντίον οποιουδήποτε αδαούς, προσαρμοστικού online και προσαρμοστικού offline αντιπάλου, αντίστοιχα. Δηλαδή, αν R_{Π} το σύνολο των randomized online αλγορίθμων που επιλύουν το Π, τότε

$$\begin{aligned}
\mathcal{C}^{\Pi}_{\mathsf{OBL}} &\triangleq \inf\left\{c^{\mathsf{OBL}}_{\text{ALG}}\right\} ,\\
\mathcal{C}^{\Pi}_{\mathsf{ADON}} &\triangleq \inf\left\{c^{\mathsf{ADON}}_{\text{ALG}}\right\} ,\\
\mathcal{C}^{\Pi}_{\mathsf{ADOFF}} &\triangleq \inf\left\{c^{\mathsf{ADOFF}}_{\text{ALG}}\right\} ,
\end{aligned}$$

όπου και στις τρεις περιπτώσεις $\text{ALG} \in R_{\Pi}$. Παρομοίως, έστω $\mathcal{C}^{\Pi}_{\mathsf{DET}}$ ο καλύτερος λόγος ανταγωνιστικότητας αιτιοκρατικού αλγορίθμου για το Π. Η παρακάτω πρόταση αποτελεί μια πρώτη ένδειξη των διαφορών δυναμικότητας μεταξύ των τριών τύπων αντιπάλων.

Πρόταση 3.7. $\mathcal{C}^{\Pi}_{\mathsf{OBL}} \le \mathcal{C}^{\Pi}_{\mathsf{ADON}} \le \mathcal{C}^{\Pi}_{\mathsf{ADOFF}} \le \mathcal{C}^{\Pi}_{\mathsf{DET}}$.

Απόδειξη. Κατ' αρχάς, ένας προσαρμοστικός αντίπαλος έχει τη δυνατότητα να μιμηθεί τη συμπεριφορά αδαούς αντιπάλου, αν αυτό τον συμφέρει. Δηλαδή, μπορεί να επιλέξει από την αρχή κάποιο στιγμιότυπο του Π και να βρει τη βέλτιστη λύση του, αγνοώντας τις τυχαίες αποφάσεις του randomized αλγορίθμου – οπότε εν προκειμένω δεν έχει σημασία αν πρόκειται για online ή offline αντίπαλο. Συνεπώς, το κόστος του τελευταίου δεν θα υπερβαίνει σε καμία περίπτωση το αντίστοιχο κόστος του αδαούς αντιπάλου, επαληθεύοντας την πρώτη ανισότητα.

[1] Φυσικά, η διατύπωση για προβλήματα μεγιστοποίησης κέρδους είναι εντελώς ανάλογη.

Εν συνεχεία, είναι προφανές ότι γενικά ο προσαρμοστικός online αντίπαλος δεν βρίσκει την καλύτερη δυνατή λύση για το στιγμιότυπο εισόδου, το οποίο κατασκευάζει επηρεαζόμενος και από τις τυχαίες αποφάσεις του εκάστοτε αλγορίθμου. Κάτι τέτοιο όμως είναι βέβαια εφικτό για τον προσαρμοστικό offline αντίπαλο, άρα $\mathcal{C}^{\Pi}_{\mathsf{ADON}} \leq \mathcal{C}^{\Pi}_{\mathsf{ADOFF}}$.

Τέλος, μπορούμε να θεωρήσουμε ότι κάθε online αλγόριθμος, randomized ή μη, φαίνεται ως αιτιοκρατικός σε έναν αντίπαλο που μπορεί να προβλέψει επακριβώς ποιες θα είναι οι τυχούσες μελλοντικές αποφάσεις του αλγορίθμου. Τοιουτοτρόπως, επειδή ο αντίπαλος αυτός έχει καταφανώς περισσότερες δυνατότητες από τον προσαρμοστικό offline, η τρίτη ανισότητα ισχύει. □

Οι Ben-David, Borodin, Karp, Tardos και Wigderson [BenDBK+90] ανέλυσαν περαιτέρω τη σχέση μεταξύ των αντιπάλων, όπως προκύπτει από τα δύο παρακάτω θεωρήματα.

Θεώρημα 3.8. [BenDBK+90] *Έστω* ALG *ένας online αλγόριθμος με λόγο ανταγωνιστικότητας α εναντίον προσαρμοστικών online αντιπάλων. Αν υπάρχει αλγόριθμος* ALG′ *για το ίδιο πρόβλημα με λόγο ανταγωνιστικότητας β εναντίον αδαών αντιπάλων, τότε ο* ALG *είναι $\alpha\beta$-ανταγωνιστικός εναντίον προσαρμοστικών offline αντιπάλων.*

Απόδειξη. Έστω ADON′ προσαρμοστικός online αντίπαλος, ο οποίος χρησιμοποιεί τον αλγόριθμο ALG′ για λύσει το στιγμιότυπο εισόδου.[2] Αν σ η είσοδος που κατασκευάζει ο βέλτιστος προσαρμοστικός offline αντίπαλος ADOFF, τότε

$$E_{\text{ALG}}[\text{κόστος}_{\text{ALG}}(\sigma) - \alpha \cdot \text{κόστος}_{\text{ADON}'}(\sigma)] \leq c_0 \; . \tag{3.4}$$

Όμως, γνωρίζουμε ότι

$$\text{κόστος}_{\text{ADON}'}(\sigma) = E_{\text{ALG}'}[\text{κόστος}_{\text{ALG}'}(\sigma)] \leq \beta \cdot \text{κόστος}_{\text{ADOFF}}(\sigma) + c_1 \; , \tag{3.5}$$

αφού ο ADOFF πάντοτε βρίσκει τη λύση της σ με το ελάχιστο κόστος. Άρα, από τις (3.4) και (3.5) έχουμε

$$E_{\text{ALG}}[\text{κόστος}_{\text{ALG}}(\sigma) - \alpha\beta \cdot \text{κόστος}_{\text{ADOFF}}(\sigma)] \leq c_0 + \alpha c_1 \; ,$$

όπερ τεκμηριώνει την $\alpha\beta$-ανταγωνιστικότητα του ALG εναντίον προσαρμοστικών offline αντιπάλων. □

Μάλιστα, το ανωτέρω θεώρημα είναι βέλτιστο, με την έννοια ότι δεν μπορεί να συναχθεί ισχυρότερο συμπέρασμα για την ανταγωνιστικότητα του ALG διατηρώντας τις ίδιες υποθέσεις.

[2] Ο ADON′ δεν είναι κατ' ανάγκη ο ισχυρότερος δυνατός προσαρμοστικός online αντίπαλος, όμως αυτό δεν επηρεάζει την απόδειξη.

Θεώρημα 3.9. [BenDBK+90] *Έστω* ALG *ένας online αλγόριθμος με λόγο ανταγωνιστικότητας α εναντίον προσαρμοστικών offline αντιπάλων. Υπάρχει αιτιοκρατικός α-ανταγωνιστικός αλγόριθμος για το ίδιο πρόβλημα.*

Απόδειξη. Παραλείπεται. □

Σημειωτέον ότι το Θεώρημα 3.9 γενικά δεν είναι κατασκευαστικό, μπορεί όμως να γίνει αν συντρέχουν ορισμένες προϋποθέσεις (βλέπε [BenDBK+90]). Από την άλλη, σε συνδυασμό με το Θεώρημα 3.8 λαμβάνουμε τα εξής:

Πόρισμα 3.10. *Έστω* ALG *ένας online αλγόριθμος με λόγο ανταγωνιστικότητας α εναντίον προσαρμοστικών online αντιπάλων. Αν υπάρχει αλγόριθμος* ALG$'$ *για το ίδιο πρόβλημα με λόγο ανταγωνιστικότητας β εναντίον αδαών αντιπάλων, τότε υπάρχει επίσης αιτιοκρατικός $\alpha\beta$-ανταγωνιστικός αλγόριθμος.*

Πόρισμα 3.11. *Έστω* ALG *ένας online αλγόριθμος με λόγο ανταγωνιστικότητας α εναντίον προσαρμοστικών online αντιπάλων. Υπάρχει αιτιοκρατικός α^2-ανταγωνιστικός αλγόριθμος για το ίδιο πρόβλημα.*

Χρησιμοποιώντας τους συμβολισμούς που θέσαμε στην αρχή της ενότητας, ισχύει

$$\mathcal{C}^{\Pi}_{\mathsf{OBL}} \cdot \mathcal{C}^{\Pi}_{\mathsf{ADON}} \geq \mathcal{C}^{\Pi}_{\mathsf{ADOFF}} = \mathcal{C}^{\Pi}_{\mathsf{DET}} \tag{3.6}$$

και

$$\left(\mathcal{C}^{\Pi}_{\mathsf{ADON}}\right)^2 \geq \mathcal{C}^{\Pi}_{\mathsf{ADOFF}} = \mathcal{C}^{\Pi}_{\mathsf{DET}} \; . \tag{3.7}$$

Επομένως, αν λόγου χάρη γνωρίζουμε ότι $\mathcal{C}^{\Pi}_{\mathsf{DET}} \geq B$, τότε συμπεραίνουμε αμέσως ότι κάθε randomized online αλγόριθμος ALG για το Π έχει λόγο ανταγωνιστικότητας $c^{\mathsf{ADON}}_{\text{ALG}} \geq \sqrt{B}$.

3.3 Κάτω φράγματα

Όσο και αν φαίνεται περίεργο, η συνηθέστερη μέθοδος εύρεσης κάτω φραγμάτων του λόγου ανταγωνιστικότητας randomized online αλγορίθμων εναντίον αδαών αντιπάλων εξετάζει την επίδοση αιτιοκρατικών αλγορίθμων σε πιθανοτικές κατανομές στιγμιοτύπων. Για να γίνουμε σαφέστεροι, κατ' αναλογία με τους Ορισμούς 1.1 και 1.2 δίνουμε τους εξής:

Ορισμός 3.12. *Έστω $\mathcal{P}$ πιθανοτική κατανομή επί των στιγμιοτύπων ενός προβλήματος βελτιστοποίησης* Π. *Ο (αιτιοκρατικός) online αλγόριθμος* ALG *θα θεωρείται c-ανταγωνιστικός ως προς την $\mathcal{P}$ αν ισχύει:*

$$E_{\mathcal{P}}[\text{κόστος}_{\text{ALG}}(I)] \leq c \cdot E_{\mathcal{P}}[\text{κόστος}_{\text{OPT}}(I)] + c_0 \; ,$$

$$\left[\ \acute{\eta} \quad E_{\mathcal{P}}[\text{κέρδος}_{\text{OPT}}(I)] \leq c \cdot E_{\mathcal{P}}[\text{κέρδος}_{\text{ALG}}(I)] + c_0\ \right],$$

όπου c_0 *σταθερά και η είσοδος* I *είναι τυχαία μεταβλητή που ακολουθεί την κατανομή* $\mathcal{P}$.

Ορισμός 3.13. *Ο λόγος ανταγωνιστικότητας* $c^{\mathcal{P}}_{\text{ALG}}$ *του online αλγορίθμου* ALG *ως προς την πιθανοτική κατανομή* $\mathcal{P}$ *προκύπτει ως:*

$$c^{\mathcal{P}}_{\text{ALG}} \triangleq \inf\{c \mid \text{ο ALG είναι } c\text{-ανταγωνιστικός ως προς την } \mathcal{P}\}\ .$$

Ακολούθως διατυπώνουμε ένα βασικό θεώρημα, το οποίο φέρει την ονομασία *«αρχή του Yao»* (*Yao's principle*) και αποτελεί θεμέλιο της αποδεικτικής τεχνικής που θα παρουσιάσουμε.

Θεώρημα 3.14. *Αν* $D(\Pi)$ *το σύνολο των αιτιοκρατικών αλγορίθμων που λύνουν το πρόβλημα* Π, *τότε*

$$\mathcal{C}^{\Pi}_{\mathsf{OBL}} = \sup_{\mathcal{P}} \left\{ \inf_{\text{ALG} \in D(\Pi)} \left\{ c^{\mathcal{P}}_{\text{ALG}} \right\} \right\}\ .$$

Το προαναφερθέν έπεται από το διάσημο θεώρημα minimax της Θεωρίας Παιγνίων. Ο Yao [Yao77] το εφήρμοσε πρώτος στα πλαίσια της κλασικής Θεωρίας Πολυπλοκότητας, ενω οι Borodin, Linial και Saks [BorLS87] το αξιοποίησαν στη μελέτη των online αλγορίθμων. Παρακάτω θα δούμε πώς το Θεώρημα 3.14 βοηθά στην απόδειξη κάτω φραγμάτων του λόγου ανταγωνιστικότητας.

Θεώρημα 3.15. [FiKL$^+$88] *Έστω* ALG$'$ *οποιοσδήποτε randomized online αλγόριθμος για το πρόβλημα σελιδοποίησης μνήμης. Ο λόγος ανταγωνιστικότητας του* ALG$'$ *εναντίον αδαούς αντιπάλου αποκλείεται να είναι μικρότερος από* $\mathcal{H}_k$.

Απόδειξη. Αρκεί να βρούμε κατάλληλη πιθανοτική κατανομή $\mathcal{P}$ ώστε για κάθε αιτιοκρατικό αλγόριθμο σελιδοποίησης ALG να έχουμε $c^{\mathcal{P}}_{\text{ALG}} \geq \mathcal{H}_k$. Πράγματι, έστω A το σύνολο σελίδων που βρίσκεται αρχικά στη γρήγορη μνήμη του ALG και του βέλτιστου offline αλγορίθμου OPT, με $|A| = k$. Σχηματίζουμε το σύνολο $S = A \cap \{p\}$, όπου $p \notin A$ μια αυθαίρετη σελίδα, και θεωρούμε την $\mathcal{P}$ ως την ομοιόμορφη κατανομή επί των $k+1$ σελίδων του S. Έτσι, μια ακολουθία αιτήσεων σ κατασκευάζεται διαλέγοντας ανεξάρτητα και με τυχαίο τρόπο καθεμία αίτηση από το σύνολο S. Προφανώς, το αναμενόμενο κόστος οποιουδήποτε αιτιοκρατικού αλγορίθμου σε στιγμιότυπα που παράγονται από την κατανομή αυτή είναι

$$E_{\mathcal{P}}[\text{κόστος}_{\text{ALG}}(\sigma)] = \frac{|\sigma|}{k+1}\ ,$$

αφού για κάθε αίτηση η πιθανότητα να μην περιέχεται η ζητούμενη σελίδα στη γρήγορη μνήμη ισούται με $\frac{1}{k+1}$.

Επιδιώκουμε τώρα να φράξουμε άνω την αναμενόμενη επίδοση του OPT. Διαμερίζουμε λοιπόν την ακολουθία αιτήσεων σε φάσεις, όπως ακριβώς όταν εξετάζουμε αλγορίθμους σήμανσης. Υπενθυμίζουμε ότι σε κάθε φάση (πλην ίσως της τελευταίας) διατυπώνονται αιτήσεις για k διαφορετικές σελίδες. Επομένως, αν το πλήθος των φάσεων είναι r, ο OPT μπορεί να εξυπηρετήσει την ακολουθία με κόστος το πολύ r. Ειδικότερα, στην αρχή κάθε νέας φάσης απομακρύνει από τη γρήγορη μνήμη τη σελίδα η οποία δεν θα ζητηθεί στο υπόλοιπο της φάσης. Τοιουτοτρόπως, το αναμενόμενο βέλτιστο κόστος προκύπτει:

$$E_{\mathcal{P}}[\text{κόστος}_{\text{OPT}}(\sigma)] \leq E_{\mathcal{P}}[\mathcal{N}(|\sigma|)] \ ,$$

όπου $\mathcal{N}(|\sigma|)$ η τυχαία μεταβλητή που εκφράζει το πλήθος των φάσεων σε ακολουθία που παράγεται από την κατανομή $\mathcal{P}$.

Οι διάρκειες όλων των φάσεων είναι ανεξάρτητες μεταξύ τους και ισόνομες τυχαίες μεταβλητές με πεπερασμένη μέση τιμή, ο υπολογισμός της οποίας ανάγεται στη λύση του παρακάτω προβλήματος.

Πρόβλημα του Συλλέκτη Κουπονιών. Έστω N διαφορετικά είδη αντικειμένων, σε απεριόριστη ποσότητα το καθένα. Εκτελούμε το εξής πείραμα τύχης: διαλέγουμε ένα αντικείμενο, το οποίο ανήκει με ίση πιθανότητα σε κάποιο από τα N είδη. Πόσες κατά μέσον όρο επαναλήψεις του πειράματος πρέπει να γίνουν ώστε να λάβουμε τουλάχιστον ένα αντικείμενο από κάθε είδος;

Η απάντηση είναι:

$$\begin{aligned} E[\text{επαναλήψεις πειράματος τύχης}] &= 1 + \frac{N}{N-1} + \frac{N}{N-2} + \cdots + N = \\ &= N \cdot \mathcal{H}_N \ . \end{aligned}$$

Το αναμενόμενο μήκος κάθε φάσης ισούται με τη λύση του προβλήματος του συλλέκτη κουπονιών μείον ένα, θέτοντας $N = k+1$. Κατά συνέπεια:

$$E[\text{μήκος φάσης}] = (k+1) \cdot \mathcal{H}_{k+1} - 1 = (k+1) \cdot \mathcal{H}_k \ .$$

Από το Θεώρημα Kolmogorov (γνωστό και ως 2$^{\text{ος}}$ Ισχυρός Νόμος Μεγάλων Αριθμών) έχουμε

$$\lim_{|\sigma| \to \infty} \frac{|\sigma|}{E[\mathcal{N}(|\sigma|)]} = E[\text{μήκος φάσης}] \ ,$$

οπότε τελικά

$$\begin{aligned}
\lim_{|\sigma|\to\infty} \frac{E_{\mathcal{P}}[\text{κόστος}_{\mathrm{ALG}}(\sigma)]}{E_{\mathcal{P}}[\text{κόστος}_{\mathrm{OPT}}(\sigma)]} &\geq \lim_{|\sigma|\to\infty} \frac{\dfrac{|\sigma|}{k+1}}{E_{\mathcal{P}}[\text{κόστος}_{\mathrm{OPT}}(\sigma)]} \geq \\
&\geq \lim_{|\sigma|\to\infty} \frac{\dfrac{|\sigma|}{k+1}}{E_{\mathcal{P}}[\mathcal{N}(|\sigma|)]} = \\
&= \frac{E[\text{μήκος φάσης}]}{k+1} = \\
&= \frac{(k+1)\cdot \mathcal{H}_k}{k+1} = \mathcal{H}_k \, .
\end{aligned}$$

Επομένως, $c_{\mathrm{ALG}}^{\mathcal{P}} \geq \mathcal{H}_k$ για κάθε αιτιοκρατικό online αλγόριθμο. Εφαρμόζοντας το Θεώρημα 3.14 εξάγουμε το επιθυμητό κάτω φράγμα για τον λόγο ανταγωνιστικότητας randomized αλγορίθμων εναντίον αδαών αντιπάλων. □

Κεφάλαιο 4

Το Πρόβλημα των k-Εξυπηρετητών

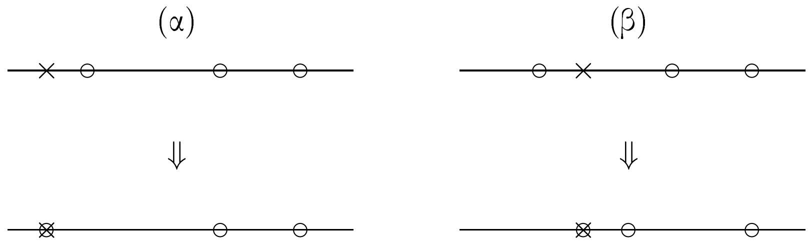
Το κεφάλαιο αυτό αφιερώνεται σε ένα πρόβλημα το οποίο, όπως έχουμε επανειλημμένως τονίσει, κατέχει εξέχουσα θέση στην περιοχή των online υπολογισμών. Ακολούθως, θα περιγράψουμε και θα μελετήσουμε διαφόρων ειδών αλγορίθμους, τόσο εξειδικευμένους – που προορίζονται για συγκεκριμένους μετρικούς χώρους ή/και πλήθος εξυπηρετητών – όσο και γενικότερους. Η ποικιλία των αποτελεσμάτων καταδεικνύει την ερευνητική προσπάθεια η οποία έχει καταβληθεί τα τελευταία χρόνια και εκείνη με τη σειρά της επιβεβαιώνει τη σπουδαιότητα του όλου προβλήματος.

4.1 Εξυπηρετητές εντός απλών μετρικών χώρων

Δυστυχώς, λίγοι αιτιοκρατικοί αλγόριθμοι που να είναι ανταγωνιστικοί και συνάμα όχι πολύπλοκοι έχουν βρεθεί μέχρι σήμερα. Στη συνέχεια θα αναλύσουμε έναν τέτοιο αλγόριθμο, ο οποίος λειτουργεί σε ευθεία γραμμή ή δένδρο. Αντιθέτως, δεν έχουμε κάποιο αντίστοιχα κομψό αποτέλεσμα ακόμη και για σχετικά απλές γεωμετρικές δομές, όπως είναι οι Ευκλείδιοι χώροι και ο κύκλος, εκτός αν καταφύγουμε σε randomized αλγορίθμους ή περιορίσουμε σημαντικά τον αριθμό των εξυπηρετητών.

4.1.1 Ευθεία γραμμή

Ο αλγόριθμος *διπλής κάλυψης* (DOUBLE-COVERAGE ή εν συντομία DC) λειτουργεί ως εξής.



Σχήμα 4.1: Τρόπος λειτουργίας του DC. Με $\times$ απεικονίζεται η θέση της αίτησης και με $\circ$ εκείνες των εξυπηρετητών

- Αν η επόμενη αίτηση βρίσκεται εκτός του διαστήματος που ορίζουν οι ακραίοι εξυπηρετητές, τότε θα ικανοποιηθεί μετακινώντας τον πλησιέστερο εξυπηρετητή (βλέπε Σχήμα 4.1α).
- Ειδάλλως, η αίτηση εντοπίζεται μεταξύ δύο γειτονικών εξυπηρετητών. Τότε, αμφότεροι κινούνται προς το σημείο της αίτησης, με ταχύτητες ίσου μέτρου αλλά αντίθετης φοράς, έως ότου κάποιος από τους δύο φθάσει εκεί (βλέπε Σχήμα 4.1β). Στην περίπτωση που πολλοί εξυπηρετητές βρίσκονται στο ίδιο σημείο, επιλέγεται αυθαίρετα ένας ο οποίος θα μετακινηθεί.

Ο ελκυστικά απλός αυτός αλγόριθμος, αν και αμνήμων, είναι ισχυρά ανταγωνιστικός.

Θεώρημα 4.1. [CKPV90] *Ο* DC *είναι k-ανταγωνιστικός.*

Απόδειξη. Έστω $M_{\min}$ το ελάχιστο βάρος ταιριάσματος (matching) μεταξύ των εξυπηρετητών του DC και του βέλτιστου αλγορίθμου OPT. Αν συμβολίσουμε $s_1, s_2, \ldots, s_k$ τις θέσεις των εξυπηρετητών του DC, θέτουμε

$$\Sigma_{\text{DC}} \triangleq \sum_{i<j} d(s_i, s_j) \ .$$

Κατασκευάζουμε λοιπόν την παρακάτω συνάρτηση δυναμικού:

$$\Phi \triangleq k \cdot M_{\min} + \Sigma_{\text{DC}} \ ,$$

που είναι προφανώς μη αρνητική. Αρκεί να δείξουμε ότι (α) αν μια κίνηση του OPT έχει κόστος d, η συνάρτηση δυναμικού αυξάνεται κατά kd το πολύ, και (β) αν μια κίνηση του DC έχει κόστος d, η συνάρτηση δυναμικού μειώνεται κατά τουλάχιστον d. Υποθέτουμε επίσης ότι κάθε αίτηση ικανοποιείται πρώτα από τον offline και ύστερα από τον online αλγόριθμο.

Όσον αφορά το (α), εύκολα διαπιστώνουμε ότι ο όρος Σ_{DC} δεν επηρεάζεται από τις ενέργειες του OPT και το βάρος του ταιριάσματος δεν μεταβάλλεται περισσότερο από d. Συνεπώς, $\Delta\Phi = \Delta\Sigma_{\text{DC}} + k \cdot \Delta M_{\min} \leq kd$.

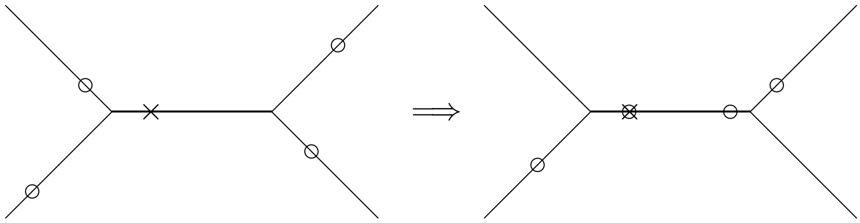
Για το (β) τώρα, έστω ότι ο online αλγόριθμος μετακινεί μόνο έναν εξυπηρετητή για να ικανοποιήσει την αίτηση, βάσει του τρόπου λειτουργίας που περιγράψαμε προηγουμένως. Εύλογα, ο εξυπηρετητής αυτός θα είναι κάποιος από τους δύο ακραίους και επιπλέον απομακρύνεται από τους υπολοίπους κατά μία απόσταση d, άρα $\Delta\Sigma_{\text{DC}} = (k-1)d$. Από την άλλη μεριά, στο ταίριασμα ελαχίστου βάρους, ο εν λόγω εξυπηρετητής θα αντιστοιχίζεται στο σημείο της αίτησης – όπου φυσικά ήδη βρίσκεται ένας εξυπηρετητής του OPT. Επομένως, εξαιτίας της κίνησης αυτής έχουμε μείωση του $M_{\min}$ ίση με d τουλάχιστον. Συνολικά, $\Delta\Phi \leq (k-1)d - kd = -d$.

Τέλος, εξετάζουμε το ενδεχόμενο ο DC να μετακινήσει δύο εξυπηρετητές κατά απόσταση $\frac{d}{2}$ έκαστον, υφιστάμενος κόστος d. Υπάρχει και πάλι ταίριασμα ελαχίστου βάρους σύμφωνα με το οποίο ένας εκ των δύο αντιστοιχίζεται στο σημείο της αίτησης. Όμως, ο άλλος εξυπηρετητής ίσως να απομακρύνεται από το «ταίρι» του καθώς μετακινείται. Δηλαδή, ο όρος $M_{\min}$ είτε μειώνεται είτε παραμένει αμετάβλητος, στη χειρότερη περίπτωση. Απομένει να υπολογίσουμε τη μεταβολή του Σ_{DC}. Το άθροισμα των αποστάσεων ενός οποιουδήποτε τρίτου εξυπηρετητή του DC από τους δύο προαναφερθέντες δεν αλλάζει, αφού οι τελευταίοι κινούνται προς αντίθετες διευθύνσεις. Εντούτοις, η μεταξύ τους απόσταση μειώνεται κατά d, οπότε $\Delta\Phi \leq 0 - d = -d$. □

Παρατήρηση. Όπως είδαμε, συχνά ο αλγόριθμος DC μετακινεί δύο εξυπηρετητές, τη στιγμή που ένας μόνο θα αρκούσε για να ικανοποιήσει την αίτηση. Για το λόγο αυτό υπάρχει και η *οκνηρή* (*lazy*) εκδοχή του αλγορίθμου, DC^{ℓ}, σύμφωνα με την οποιά δεν κινούνται περισσότεροι του ενός εξυπηρετητές σε κάθε βήμα. Έστω δηλαδή ότι για κάποια αίτηση ο DC απαιτεί να μετατοπιστούν οι εξυπηρετητές s και s', με τον πρώτο να την ικανοποιεί. Σε μια τέτοια περίπτωση, ο DC^{ℓ} κινεί μόνο τον s, αλλά στους εφεξής υπολογισμούς του θεωρεί ότι ο s' βρίσκεται στη θέση όπου θα τον είχε μεταφέρει ο DC.

Εύκολα διαπιστώνεται ότι όλοι οι αλγόριθμοι της παρούσας ενότητας επιδέχονται οκνηρές παραλλαγές. Μάλιστα, εφόσον σε κάθε μετρικό χώρο ισχύει η τριγωνική ανισότητα των αποστάσεων, η «οκνηρία» ποτέ δεν αυξάνει το συνολικό online κόστος, αντίθετα μπορεί να το μειώσει. Παρ' όλ' αυτά, η μελέτη οκνηρών αλγορίθμων είναι συνήθως εξαιρετικά δύσκολη, συνεπώς περιοριζόμαστε στην ανάλυση των μη οκνηρών ομολόγων τους.

Σημειώνουμε επίσης ότι οι Bartal, Chrobak και Larmore [BarCL98] εφηύραν έναν αμνήμονα randomized αλγόριθμο για δύο μόνον εξυπηρετητές επί ευθείας γραμμής, ο οποίος αποδεικνύεται $\frac{155}{78}$-ανταγωνιστικός εναντίον αδα-



Σχήμα 4.2: Τρόπος λειτουργίας του DC-TREE. Με $\times$ απεικονίζεται η θέση της αίτησης και με $\circ$ εκείνες των εξυπηρετητών

ούς αντιπάλου. Επειδή $\frac{155}{78} \approx 1,987 < 2$, η επίδοσή του είναι καλύτερη από εκείνη οποιουδήποτε αιτιοκρατικού αλγορίθμου. Μολαταύτα, ο λόγος ανταγωνιστικότητάς του φράσσεται κάτω από το $\frac{107}{54} \approx 1,981$.

4.1.2 Δένδρο

Κατά τα γνωστά, οποιοδήποτε δένδρο με βάρη στις ακμές μπορεί να εμβαπτιστεί (με άλλα λόγια, να «σχεδιαστεί») στο Ευκλείδιο επίπεδο ούτως ώστε το μήκος κάθε ακμής να ισούται με το βάρος της. Θεωρούμε το μετρικό χώρο με σημεία όλα τα σημεία των ακμών ενός τέτοιου δένδρου, στον οποίο η απόσταση $d(x, y)$ ορίζεται ως το μήκος του συντομότερου μονοπατιού μεταξύ των x και y – που είναι μοναδικό, αφού πρόκειται για δένδρο.

Ένας εξυπηρετητής s_i λέγεται ότι *γειτονεύει* με την αίτηση r αν δεν υπάρχει άλλος εξυπηρετητής στο συντομότερο μονοπάτι που συνδέει τους s_i και r. Βεβαίως, αν πολλοί εξυπηρετητές βρίσκονται στο ίδιο σημείο και γειτονεύουν με την r, αυθαίρετα επιλέγουμε έναν μόνο εξ αυτών, ο οποίος πράγματι θα λογίζεται ως γείτονας της r, ενώ οι υπόλοιποι όχι.

Στηριζόμενοι στα παραπάνω μπορούμε εύκολα να περιγράψουμε τη λειτουργία του αλγορίθμου με την ονομασία DC-TREE. Συγκεκριμένα, όταν λαμβάνεται μια νέα αίτηση, όλοι οι εξυπηρετητές που γειτονεύουν με αυτή μετακινούνται με σταθερή ταχύτητα προς την κατεύθυνση της αίτησης, έως ότου κάποιος φθάσει στο ζητούμενο σημείο, όπως στο Σχήμα 4.2. Παρατηρούμε ότι στην εκφυλισμένη περίπτωση που το δέντρο είναι απλά μια ευθεία γραμμή, ο DC-TREE ταυτίζεται με τον DC.

Αξίζει να σημειωθεί ότι, γενικά, για την ικανοποίηση μιας μεμονωμένης αίτησης ενδέχεται δύο ή περισσότεροι συνολικά εξυπηρετητές του DC-TREE να αλλάξουν θέση. Επιπλέον, καθώς ένας εξυπηρετητής μετακινείται ίσως πάψει να γειτονεύει με την αίτηση, πράγμα το οποίο συμβαίνει αν παρεμβληθεί μεταξύ τους κάποιος άλλος κινούμενος εξυπηρετητής. Εύλογα, ο πρώτος τότε σταματά στο σημείο που βρίσκεται εκείνη τη στιγμή.

Θεώρημα 4.2. [CL91b] *Ο* DC-TREE *είναι k-ανταγωνιστικός.*

Απόδειξη. Θεωρούμε την ίδια συνάρτηση δυναμικού όπως στο Θεώρημα 4.1. Προφανώς, μια κίνηση του αντιπάλου που έχει κόστος d αυξάνει τη Φ κατά kd το πολύ. Από την άλλη μεριά, οι ενέργειες του αλγορίθμου για την ικανοποίηση μιας αίτησης είναι συνήθως αρκετά πολύπλοκες για να μελετηθούν άμεσα. Επομένως, θα διαμερίσουμε τη διαδικασία αυτή σε φάσεις, έτσι ώστε στη διάρκεια κάθε φάσης το πλήθος των εξυπηρετητών που κινούνται να είναι σταθερό.

Έστω λοιπόν ότι σε μία φάση έχουμε m εξυπηρετητές που γειτονεύουν με την αίτηση και μετατοπίζονται κατά απόσταση d έκαστος, επισύροντας συνολικό κόστος md. Κάποιος εξ αυτών θα αντιστοιχίζεται στο σημείο της αίτησης σε ένα ταίριασμα ελαχίστου βάρους και εξαιτίας του ο όρος $M_{\min}$ μειώνεται κατά d. Όμως, οι υπόλοιποι $m-1$ εξυπηρετητές πιθανόν να απομακρύνονται από τα «ταίρια» τους ενώ κινούνται, οπότε $\Delta M_{\min} \leq (m-1)d - d = (m-2)d$. Όσον αφορά τη μεταβολή του Σ_{DC}, έστω s ένας εξυπηρετητής ο οποίος δεν γειτονεύει με την αίτηση. Κάποιος εκ των m προαναφερθέντων εξυπηρετητών απομακρύνεται από τον s, αλλά οι υπόλοιποι πλησιάζουν σε αυτόν, επιφέροντας ελάττωση του Σ_{DC} ίση με $(k-m)(m-2)d$. Επιπρόσθετα, η απόσταση μεταξύ κάθε ζεύγους κινούμενων εξυπηρετητών μειώνεται κατά $2d$, και αφού υπάρχουν $\frac{m(m-1)}{2}$ τέτοια ζεύγη, έχουμε

$$\Delta\Sigma_{\text{DC}} = -(k-m)(m-2)d - \frac{m(m-1)}{2} \cdot 2d = -kmd + 2kd - dm \ .$$

Τελικά,

$$\Delta\Phi = k \cdot \Delta M_{\min} + \Delta\Sigma_{\text{DC}} \leq k(m-2)d - kmd + 2kd - dm = -dm \ ,$$

άρα η απόδειξη έχει ολοκληρωθεί. □

4.1.3 Κύκλος

Έστω ότι διαθέτουμε k εξυπηρετητές, οι οποίοι μπορούν να κινούνται επί της περιφέρειας ενός κύκλου για την ικανοποίηση αιτήσεων. Ακολούθως διατυπώνουμε τον randomized αλγόριθμο CIRC, ο οποίος βρίσκει εφαρμογή σε μια τέτοια περίπτωση. Κατ' αρχάς, ο CIRC επιλέγει τυχαία και με ομοιόμορφη πιθανοτική κατανομή ένα σημείο P στην περιφέρεια του κύκλου, το οποίο θεωρείται «οδόφραγμα» που κανένας εξυπηρετητής δεν επιτρέπεται να περάσει είτε προς τη μία είτε προς την άλλη κατεύθυνση. Έτσι, εφόσον ο μετρικός χώρος όπου κινούνται οι εξυπηρετητές ισοδυναμεί πλέον με μια ευθεία γραμμή, ο CIRC μιμείται τον DC για να ικανοποιήσει τις εισερχόμενες αιτήσεις.

Θεώρημα 4.3. [Karp89] *Ο* CIRC *είναι* $2k$*-ανταγωνιστικός εναντίον αδαούς αντιπάλου.*

Απόδειξη. Ως συνήθως, συμβολίζουμε OPT τον βέλτιστο offline αλγόριθμο για τον κύκλο, και επιπλέον OPT-LINE τον βέλτιστο offline αλγόριθμο για ευθεία γραμμή – ή εν προκειμένω το ευθύγραμμο τμήμα που προκύπτει από την περιφέρεια του κύκλου η οποία «κόβεται» στο σημείο P και «τεντώνεται». Από τον ορισμό του αλγορίθμου CIRC ισχύει

$$\text{κόστος}_{\text{CIRC}}(\sigma) \leq k \cdot \text{κόστος}_{\text{OPT-LINE}}(\sigma) \ , \tag{4.1}$$

για κάθε ακολουθία αιτήσεων σ.

Αρκεί λοιπόν να φράξουμε άνω το $\text{κόστος}_{\text{OPT-LINE}}(\sigma)$ σε σύγκριση με το $\text{κόστος}_{\text{OPT}}(\sigma)$. Για το σκοπό αυτό, θεωρούμε έναν ακόμη αλγόριθμο OPT$'$, που συμπεριφέρεται όπως ακριβώς ο OPT με μια μόνο διαφορά: όταν κάποιος εξυπηρετητής του τελευταίου μετακινείται από ένα σημείο A στο B, όπου $P \in \overset{\frown}{AB}$, ο OPT$'$ υποχρεώνει τον αντίστοιχο δικό του εξυπηρετητή να διανύσει το συμπληρωματικό τόξο $\overset{\smile}{AB}$, ώστε να παρακάμψει το P. Τούτο βέβαια συμβαίνει με πιθανότητα ανάλογη του μήκους του τόξου $\overset{\frown}{AB}$, λόγω του τρόπου επιλογής του P. Παρατηρούμε ότι ο OPT$'$ αποτελεί ουσιαστικά αλγόριθμο για ευθύγραμμο τμήμα και όχι για κύκλο, συνεπώς

$$\text{κόστος}_{\text{OPT-LINE}}(\sigma) \leq \text{κόστος}_{\text{OPT}'}(\sigma) \ . \tag{4.2}$$

Έστω $d_1, d_2, \ldots, d_n$ και $d'_1, d'_2, \ldots, d'_n$ τα μήκη των μεμονωμένων διαδρομών που καλύπτουν οι εξυπηρετητές του OPT και του OPT$'$, αντίστοιχα. Υποθέτοντας χωρίς βλάβη της γενικότητας ότι το μήκος της περιφέρειας του κύκλου είναι 1, έχουμε

$$E[d'_i] = \underbrace{d_i(1-d_i)}_{\text{παράκαμψη}} + \underbrace{(1-d_i)d_i}_{\substack{\text{κανονική} \\ \text{κίνηση}}} = 2\left(d_i - d_i^2\right) \ , \tag{4.3}$$

όπου η αναμενόμενη τιμή λαμβάνεται ως προς όλες τις δυνατές επιλογές του P. Άρα,

$$\begin{aligned} E[\text{κόστος}_{\text{OPT}'}(\sigma)] &= \sum_{i=1}^{n} E[d'_i] = 2\sum_{i=1}^{n}\left(d_i - d_i^2\right) \leq \\ &\leq 2\sum_{i=1}^{n} d_i = 2 \cdot \text{κόστος}_{\text{OPT}}(\sigma) \ . \end{aligned} \tag{4.4}$$

Από τις (4.1), (4.2) και (4.4) συνάγεται το επιθυμητό συμπέρασμα. □

Ο αλγόριθμος CIRC παρουσιάζει ιδιαίτερο ενδιαφέρον, αφ' ενός επειδή είναι απλός και αμνήμων, αφ' ετέρου διότι εγκαινιάζει μια νέα μέθοδο κατασκευής randomized αλγορίθμων, την οποία θα εξετάσουμε αναλυτικότερα στην Ενότητα 4.4.

Τέλος, αναφέρουμε ότι οι Fiat, Rabani, Ravid και Schieber [FiRRS94] πρότειναν μια αιτιοκρατική παραλλαγή του CIRC, που όμως απαιτεί $\Omega(k)$ bits μνήμης και είναι $O(k^3)$-ανταγωνιστική. Από τα μέχρι τώρα αποτελέσματα φαίνεται μάλλον αδύνατο να βρεθεί αμνήμων αιτιοκρατικός αλγόριθμος για τον κύκλο με ικανοποιητικό λόγο ανταγωνιστικότητας, αν και κάτι τέτοιο μένει να αποδειχθεί.

4.1.4 Ευκλείδιοι χώροι

Μολονότι γνωρίζουμε ότι για την ευθεία γραμμή υπάρχει ισχυρά ανταγωνιστικός αλγόριθμος με σύντομη διατύπωση και μικρή υπολογιστική πολυπλοκότητα, δεν μπορούμε να ισχυριστούμε το ίδιο για Ευκλείδιους χώρους υψηλότερης διάστασης. Αντ' αυτού, στη συνέχεια παρουσιάζουμε έναν απλό αιτιοκρατικό αλγόριθμο για δύο μόνον εξυπηρετητές, με επίδοση σχετικά κοντά στην καλύτερη δυνατή.

Έστω $\mathcal{M}$ Ευκλείδιος χώρος. Για οποιαδήποτε τρία σημεία $x, y, z \in \mathcal{M}$ ορίζουμε την ποσότητα $\texttt{slack}(x, y, z)$ ως

$$\texttt{slack}(x, y, z) \triangleq d(x, y) + d(x, z) - d(y, z) \ .$$

Λόγω της τριγωνικής ανισότητας, ισχύει πάντοτε ότι $\texttt{slack}(x, y, z) \geq 0$. Θεωρούμε τώρα έναν πραγματικό αριθμό $\gamma \in [0, 1]$ και διατυπώνουμε τον αλγόριθμο SLACK-COVERAGE$_\gamma$ (ή εν συντομία SC$_\gamma$). Ο τελευταίος διαθέτει δύο εξυπηρετητές και λειτουργεί ως εξής. Έστω r το σημείο που ζητείται από την τρέχουσα αίτηση και x ο πλησιέστερος σε αυτήν εξυπηρετητής. Αν το r ισαπέχει από τους δύο εξυπηρετητές, επιλέγεται αυθαίρετα ο ένας από τους δύο. Κατόπιν, ο έτερος εξυπηρετητής y μετακινείται προς την κατεύθυνση του x καλύπτοντας απόσταση $\gamma \cdot \texttt{slack}(x, y, r)$ και εν τέλει ο x πηγαίνει στο r για να ικανοποιήσει την αίτηση.

Τονίζεται ότι στις περιπτώσεις που αμφότεροι οι εξυπηρετητές κινούνται, η νέα θέση του πιο απομακρυσμένου εξυπηρετητή y απέχει λιγότερο από το σημείο r σε σχέση με την αρχική. Παρατηρούμε επίσης ότι αν ο χώρος $\mathcal{M}$ είναι ευθεία γραμμή, ο SC$_{1/2}$ ταυτίζεται με τον DC.

Για να δείξουμε την ανταγωνιστικότητα του SC$_\gamma$ ορίζουμε την ακόλουθη συνάρτηση δυναμικού, με παραμέτρους $\alpha, \beta \in \mathbb{R}_+$:

$$\Phi \triangleq \alpha \cdot M_{\min} + \beta \cdot d(x, y) \ ,$$

όπου, κατά τα γνωστά, $M_{\min}$ είναι το βάρος του ελαχίστου ταιριάσματος μεταξύ των εξυπηρετητών του SC_γ και του βέλτιστου offline αλγορίθμου OPT για το ίδιο πρόβλημα. Η Φ προφανώς λαμβάνει μη αρνητικές τιμές και επιπλέον αποτελεί γενίκευση της συνάρτησης δυναμικού που χρησιμοποιήσαμε στην ανάλυση των DC και DC-TREE – για την περίπτωση που $k = 2$, βεβαίως.

Χρησιμοποιώντας την εν λόγω Φ, μπορούμε να δείξουμε ότι ο SC_γ είναι α-ανταγωνιστικός, αλλά όχι κάτι ισχυρότερο. Πράγματι, μια κίνηση του OPT με κόστος d αυξάνει τον όρο $M_{\min}$ κατά d το πολύ, άρα και την Φ κατά αd το πολύ. Αρκεί λοιπόν να τεκμηριωθεί ότι όταν ο SC_γ ικανοποιεί μια αίτηση (μετά από τον OPT), επισύρει μείωση της Φ τουλάχιστον ίση με το κόστος που υφίσταται στο βήμα αυτό της λειτουργίας του.

Κατ' αρχάς, έστω y' η νέα θέση του εξυπηρετητή y. Όπως αναφέραμε, $d(y', r) \le d(y, r)$, συνεπώς

$$\Delta d(x, y) = d(y', r) - d(x, y) \le d(y, r) - d(x, y) \ .$$

Αν συμβολίσουμε s_1 και s_2 τους εξυπηρετητές του OPT και υποθέσουμε ότι ο s_1 μόλις ικανοποίησε την αίτηση, δηλαδή $s_1 \equiv r$, τότε διακρίνουμε δύο περιπτώσεις:

- Στο ελάχιστο ταίριασμα, ο x αντιστοιχίζεται με τον s_1. Επομένως, η μετακίνηση του x μειώνει το $M_{\min}$ κατά $d(x, r)$, αλλά εκείνη του y ενδεχομένως να το αυξήσει κατά $\gamma \cdot \texttt{slack}(x, y, r)$. Συνολικά δηλαδή

$$\Delta\Phi \le \alpha\big[\gamma \cdot \texttt{slack}(x, y, r) - d(x, r)\big] + \beta\big[d(y, r) - d(x, y)\big] \ . \tag{4.5}$$

- Στο ελάχιστο ταίριασμα, ο y αντιστοιχίζεται με τον s_1. Προφανώς, όμως, μετά την ικανοποίηση της αίτησης ο y θα αντιστοιχίζεται με τον s_2. Έτσι,

$$\begin{aligned} \Delta M_{\min} &= d(y', s_2) - \big[d(x, s_2) + d(y, r)\big] \le \\ &\le d(y', x) - d(y, r) = \\ &= d(x, y) - \gamma \cdot \texttt{slack}(x, y, r) - d(y, r) \ . \end{aligned}$$

  Συμπεραίνουμε λοιπόν ότι

$$\begin{aligned} \Delta\Phi \le \alpha\big[d(x, y) - \gamma \cdot \texttt{slack}(x, y, r) - d(y, r)\big] + \\ + \beta\big[d(y, r) - d(x, y)\big] \ . \end{aligned} \tag{4.6}$$

Υπενθυμίζουμε ότι πρέπει να αποδείξουμε πως

$$\Delta\Phi \le -\big[d(x, r) + \gamma \cdot \texttt{slack}(x, y, r)\big] \ . \tag{4.7}$$

Αντικαθιστώντας τις (4.5) και (4.6) στην (4.7) και ομαδοποιώντας τους όρους λαμβάνουμε τις δύο παρακάτω συνθήκες:

$$\begin{aligned} d(x,y)\left[\gamma(\alpha+1)-\beta\right]+d(x,r)\left[\gamma(\alpha+1)+1-\alpha\right]+ \\ +\,d(y,r)\left[b-\gamma(\alpha+1)\right]\le 0\ , \end{aligned} \tag{4.8}$$

$$\begin{aligned} d(x,y)\left[\gamma(1-\alpha)+\alpha-\beta\right]+d(x,r)\left[\gamma(1-\alpha)\right]+ \\ +\,d(y,r)\left[\gamma(1-\alpha)+\beta-\alpha\right]\le 0\ . \end{aligned} \tag{4.9}$$

Εύκολα διαπιστώνουμε ότι οι ανωτέρω ανισότητες ικανοποιούνται όταν $\alpha = 3$, $\beta = 2$ και $\gamma = \frac{1}{2}$, ενώ κάτι τέτοιο δεν είναι δυνατόν για οποιαδήποτε τιμή του α μικρότερη από 3. Κατά συνέπεια, προκύπτει το ακόλουθο θεώρημα:

Θεώρημα 4.4. [Bar94] *Ο αλγόριθμος* $\mathrm{SC}_{1/2}$ *είναι 3-ανταγωνιστικός.*

4.2 Αλγόριθμος εξισορρόπησης κόστους

Στην Υποενότητα 2.1.2 αποδείξαμε το Θεώρημα 2.1, σύμφωνα με το οποίο ο αλγόριθμος εξισορρόπησης κόστους BALANCE είναι k-ανταγωνιστικός σε μετρικούς χώρους με $k+1$ σημεία. Ακολούθως θα δούμε πώς προκύπτει το ίδιο αποτέλεσμα με τη μέθοδο της συνάρτησης δυναμικού. Όπως και στην πρώτη απόδειξη του θεωρήματος, χρησιμοποιούμε πάλι τους συμβολισμούς D_i^t και h^t.

Λήμμα 4.5. *Για κάθε* $i, j \neq h^t$ *ισχύει*

$$\left|D_i^t - D_j^t\right| \le d(i,j)\ .$$

Απόδειξη. Εφαρμόζουμε επαγωγή ως προς t.

- Για $t = 0$ προφανώς αληθεύει.
- Έστω ότι για κάποιο $u \ge 0$ έχουμε $i, j \neq h^u \Rightarrow \left|D_i^u - D_j^u\right| \le d(i,j)$. Αν $min \neq h^u$ το σημείο που ελαχιστοποιεί την ποσότητα $D_i^u + d(i, h^u)$, ο BALANCE θα στείλει στην οπή h^u τον εξυπηρετητή που βρίσκεται στο min, ώστε να ικανοποιήσει την $(u+1)$-οστή αίτηση. Τότε $D_{h^u}^{u+1} \leftarrow D_{min}^u + d(min, h^u)$, ενώ για όλες τις τιμές του j πλην των min και h^u ισχύει $D_j^{u+1} = D_j^u$. Αρκεί λοιπόν να δείξουμε ότι

$$i \neq min \implies \left|D_i^{u+1} - D_{h^u}^{u+1}\right| \le d(i, h^u)\ . \tag{4.10}$$

Τω όντι, αφ' ενός

$$\begin{aligned} D_i^{u+1} - D_{h^u}^{u+1} &= D_i^u - (D_{min}^u + d(min, h^u)) \le \\ &\le |D_i^u - D_{min}^u| - d(min, h^u) \le \\ &\le d(i, min) - d(min, h^u) \le \qquad (4.11) \\ &\le d(i, h^u)\ , \qquad (4.12) \end{aligned}$$

και αφ' ετέρου

$$D_{h^u}^{u+1} - D_i^{u+1} = D_{min}^u + d(min, h^u) - D_i^u \leq d(i, h^u) \ , \tag{4.13}$$

λόγω του τρόπου επιλογής του min. Από τις (4.12) και (4.13) συνάγεται η (4.10). □

Πέραν αυτών, από την (4.11) συμπεραίνουμε ότι

$$D_j^t - D_{h^{t-1}}^t \geq d\big(j, h^t\big) - d\big(h^t, h^{t-1}\big) \ , \tag{4.14}$$

για κάθε $t \geq 1$ και $j \neq h^t$. Με τις παραπάνω σχέσεις είμαστε πλέον σε θέση να αποδείξουμε την k-ανταγωνιστικότητα του BALANCE.

Θεωρούμε τον βέλτιστο offline αλγόριθμο OPT για το πρόβλημα των k-εξυπηρετητών και O_t το σύνολο σημείων που καλύπτουν οι εξυπηρετητές του μετά την ικανοποίηση της t-οστής αίτησης. Για $t \geq 1$ ορίζουμε την παρακάτω συνάρτηση δυναμικού:

$$\Phi_t \triangleq \begin{cases} k \cdot \big(D_{h^{t-1}}^t - d(h^t, h^{t-1})\big) - S, & \text{αν } h^t \notin O_t \\ k \cdot D_i^t - S, & \text{αν } (i \notin O_t) \wedge (i \neq h^t) \end{cases} ,$$

όπου $S = \sum_{i \neq h^t} D_i^t$. Λόγω του Λήμματος 4.5 διαπιστώνουμε ότι εφόσον $t \geq 1$ υπάρχει σταθερά b ανεξάρτητη της ακολουθίας αιτήσεων τέτοια ώστε $\Phi_t \geq b$.

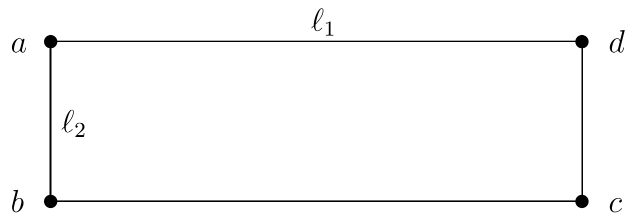
Πρέπει να δείξουμε ότι (α) αν μια κίνηση του OPT έχει κόστος d, η συνάρτηση δυναμικού αυξάνεται κατά kd το πολύ, και (β) αν μια κίνηση του BALANCE έχει κόστος d, η συνάρτηση δυναμικού μειώνεται κατά τουλάχιστον d. Ως συνήθως, υποθέτουμε ότι κάθε αίτηση ικανοποιείται πρώτα από τον offline και ύστερα από τον online αλγόριθμο. Φυσικά, το σημείο που ζητείται κατά την $(t+1)$-οστή αίτηση δεν είναι άλλο από το h^t.

Όσον αφορά το (α), αν $h^t \in O_t$ ο OPT δεν χρειάζεται να εκτελέσει καμία ενέργεια. Έστω λοιπόν ότι $h^t \notin O_t$ και ότι η αίτηση ικανοποιείται μετακινώντας τον εξυπηρετητή που βρίσκεται στο σημείο j με κόστος $d(j, h^t)$. Έχουμε:

$$\begin{aligned} \Delta\Phi &= k \cdot D_j^t - S - \big(k \cdot \big(D_{h^{t-1}}^t - d\big(h^t, h^{t-1}\big)\big) - S\big) = \\ &= k \cdot \big(D_j^t - D_{h^{t-1}}^t + d\big(h^t, h^{t-1}\big)\big) \leq \\ &\overset{(*)}{\leq} k \cdot \big(d\big(j, h^t\big) - d\big(h^t, h^{t-1}\big) + d\big(h^t, h^{t-1}\big)\big) = \\ &= k \cdot d\big(j, h^t\big) \ , \end{aligned}$$

όπου στο σημείο $(*)$ χρησιμοποιήθηκε η (4.14).

Για το (β) τώρα, μετά την κίνηση του OPT υπάρχει κάποιο $j \neq h^t$ τέτοιο ώστε $j \notin O_{t+1}$. Συμβολίζουμε S' τη νέα τιμή του S αφ' ότου ο BALANCE ικανοποιήσει την αίτηση μετακινώντας τον εξυπηρετητή που βρίσκεται στο



Σχήμα 4.3: Αντιπαράδειγμα της ανταγωνιστικότητας του BALANCE

σημείο ℓ· επομένως, $S' = S + d(\ell, h^t)$. Προκύπτει λοιπόν ότι:

$$\Delta\Phi = \left\{ \begin{array}{ll} k \cdot \big(D^t_\ell + d\big(\ell, h^t\big) - d\big(\ell, h^t\big)\big) - S' - \big(k \cdot D^t_j - S\big)\,, & \text{αν } \ell = j \\ k \cdot D^t_j - S' - \big(k \cdot D^t_j - S\big)\,, & \text{αν } \ell \neq j \end{array} \right\} =$$

$$= \left\{ \begin{array}{ll} k \cdot \big(D^t_\ell - D^t_j\big) - S' + S, & \text{αν } \ell = j \\ -S' + S, & \text{αν } \ell \neq j \end{array} \right\} = -d\big(\ell, h^t\big)\ ,$$

όπως ακριβώς απαιτείται. Έτσι, καταλήγουμε για άλλη μια φορά στο Θεώρημα 2.1.

Δυστυχώς, ο αλγόριθμος BALANCE δεν είναι ανταγωνιστικός για μετρικούς χώρους με $N > k + 1$ σημεία, γεγονός που έχουμε ήδη τονίσει. Ακολούθως θα δώσουμε ένα σχετικό αντιπαράδειγμα, στο οποίο $k = 2$ και $N = 4$. Θεωρούμε το μετρικό χώρο 4 σημείων που αναπαρίσταται στο Σχήμα 4.3, για τον οποίο ισχύει $\ell_1 \gg \ell_2$. Υποθέτουμε ότι οι εξυπηρετητές του BALANCE βρίσκονται αρχικά στα σημεία c και d. Έστω τώρα η ακολουθία αιτήσεων $\sigma = a, b, c, d, a, b, c, d, \ldots, a, b, c, d$, μήκους $4n$. Διαπιστώνουμε ότι $\textit{κόστος}_{\text{BAL}}(\sigma) = 4n\ell_1$, ενώ προφανώς $\textit{κόστος}_{\text{OPT}}(\sigma) = \ell_1 + (4n - 1)\ell_2$. Κατά συνέπεια, ο λόγος ανταγωνιστικότητας αποκλείεται να είναι καλύτερος από $\frac{\ell_1}{\ell_2}$, κλάσμα που μπορεί να γίνει αυθαίρετα μεγάλο. Άρα, στην περίπτωση αυτή ο BALANCE αποδεικνύεται μη ανταγωνιστικός.

4.3 Αλγόριθμος συνάρτησης έργου

Δίνεται οποιοσδήποτε μετρικός χώρος $\mathcal{M}$. Στην παρούσα ενότητα, ως *διάταξη* (*configuration*) των εξυπηρετητών ενός αλγορίθμου θα νοείται το πολυσύνολο των σημείων στα οποία βρίσκονται οι τελευταίοι. Για δύο διατάξεις C_1 και C_2, η παράσταση $C_1 + C_2$ συμβολίζει την πράξη της ένωσης πολυσυνόλων και η $C_1 - C_2$ την αφαίρεση πολυσυνόλων. Αν p σημείο και C διάταξη, γράφουμε για συντομία $C + p$ και $C - p$ αντί των $C + \{p\}$ και $C - \{p\}$ αντιστοίχως. Με $D(C_1, C_2)$ συμβολίζουμε το βάρος του ελάχιστου ταιριάσματος μεταξύ των (ισοπληθικών) διατάξεων C_1 και C_2.

Στην πραγματικότητα, βέβαια, ο βέλτιστος online αλγόριθμος ποτέ δεν θα μεταφέρει περισσότερους του ενός εξυπηρετητές στην ίδια θέση. Το αυτό συμβαίνει και με τον αλγόριθμο συνάρτησης έργου που θα παρουσιάσουμε προσεχώς. Υπάρχει όμως το ενδεχόμενο πολλοί εξυπηρετητές να ήταν τοποθετημένοι σε κάποιο σημείο κατά την αρχική διάταξη και να μην έχουν κινηθεί έκτοτε· συνεπώς, υποχρεωτικά ορίζουμε τις διατάξεις ως πολυσύνολα και όχι ως απλά σύνολα.

Έστω ακολουθία αιτήσεων $\sigma = r_1, r_2, \ldots, r_n$. Θέτουμε $\sigma_i = r_1, r_2, \ldots, r_i$ και $\sigma_0 = \emptyset$. Για κάθε διάταξη C, αρχική διάταξη C_0 και $i = 0, 1, \ldots, n$, η *συνάρτηση έργου* (*work function*) για το πρόβλημα των k-εξυπηρετητών $w_{\sigma_i}(C) = w_{\sigma_i}(C_0, C)$ ορίζεται ως το βέλτιστο offline κόστος ικανοποίησης των αιτήσεων της σ_i, με τη σειρά που εμφανίζονται, ξεκινώντας από την διάταξη C_0 και καταλήγοντας στην C. Σημειωτέον ότι η C_0 θεωρείται συνήθως δεδομένη και γι' αυτό παραλείπεται από το συμβολισμό της w_{σ_i}.

Αν r η $(i{+}1)$-οστή αίτηση, ισχύει $r \in C \Rightarrow w_{\sigma_i r}(C) = w_{\sigma_i}(C)$. Γενικότερα λοιπόν έχουμε:

$$\begin{aligned} w_{\sigma_i r}(C) &= \min_{x \in C} \{ w_{\sigma_i r}(C - x + r) + d(r, x) \} = \\ &= \min_{x \in C} \{ w_{\sigma_i}(C - x + r) + d(r, x) \} \ , \end{aligned}$$

αφού $r \in C - x + r$. Άρα, οι συναρτήσεις έργου ικανοποιούν την ακόλουθη αναδρομική σχέση:

$$\begin{aligned} w_{\emptyset}(C) &= D(C_0, C) \\ w_{\sigma_i r}(C) &= \min_{x \in C} \{ w_{\sigma_i}(C - x + r) + d(r, x) \} \ . \end{aligned} \tag{4.15}$$

Είναι φανερό λοιπόν ότι $\text{κόστος}_{\text{OPT}}(\sigma) = \min_C w_\sigma(C)$. Με άλλα λόγια, το βέλτιστο offline κόστος μπορεί να υπολογιστεί χρησιμοποιώντας δυναμικό προγραμματισμό, χάρη στη σχέση (4.15).

Στη συνέχεια θα περιγράψουμε τον αλγόριθμο συνάρτησης έργου WFA. Έστω σ_i το αρχικό τμήμα της ακολουθίας αιτήσεων που έχει ήδη ικανοποιηθεί και C η τρέχουσα διάταξη εξυπηρετητών του WFA. Όταν ληφθεί η επόμενη αίτηση $r \equiv r_{i+1}$, ο αλγόριθμος θα μετακινήσει τον εξυπηρετητή s που δίνεται από τη σχέση

$$s = \arg\min_{x \in C} \{ w_{\sigma_i}(C - x + r) + d(r, x) \}$$

στο σημείο r. Σε περίπτωση ισοπαλίας, επιλέγεται αυθαίρετα ένας από τους ισόπαλους εξυπηρετητές.

Ο WFA έχει προταθεί ανεξάρτητα από διάφορους ερευνητές, και μάλιστα όχι κατ' αποκλειστικότητα για το πρόβλημα των k-εξυπηρετητών. Μια πρώτη ουσιώδης μελέτη του – όσον αφορά το υπό εξέταση πρόβλημα – δημοσιεύτηκε από τους Chrobak και Larmore [CL92]. Λίγο αργότερα, οι Κουτσουπιάς και Παπαδημητρίου έδειξαν το ακόλουθο διάσημο θεώρημα:

Θεώρημα 4.6. [KoutP94] *Για κάθε μετρικό χώρο και πλήθος εξυπηρετητών k, ο αλγόριθμος* WFA *είναι $(2k-1)$-ανταγωνιστικός.*

Απόδειξη. Σκιαγραφούμε μόνο τις κύριες ιδέες από τις οποίες προκύπτει το επιθυμητό αποτέλεσμα, χωρίς να επεκταθούμε σε λεπτομέρειες. Συγκεκριμένα, οι αποδείξεις των λημμάτων που θα αναφέρουμε παρακάτω παραλείπονται.

Θεωρούμε ότι ένας απηνής αντίπαλος κατασκευάζει ακολουθία αιτήσεων $\sigma = r_1, r_2, \ldots, r_n$. Χωρίς βλάβη της γενικότητας, υποθέτουμε ότι ο WFA και ο βέλτιστος offline αλγόριθμος OPT ξεκινούν από την ίδια αρχική διάταξη εξυπηρετητών C_0 και επίσης καταλήγουν στην ίδια τελική διάταξη C_n. Πράγματι, έστω ότι συνέβαινε το αντίθετο: δηλαδή, ο OPT κατέληγε στην C_n αλλά ο WFA σε κάποια C'_n. Τότε ο αντίπαλος θα μπορούσε να διατυπώσει επιπλέον αιτήσεις για σημεία της C_n, χωρίς να αυξήσει το συνολικό κόστος του OPT, έως ότου οι διατάξεις των δύο αλγορίθμων ταυτιστούν.

Έστω ότι μετά και το i-οστό βήμα της λειτουργίας του ο WFA βρίσκεται στη διάταξη C. Κατόπιν, μετακινεί τον εξυπηρετητή που βρίσκεται στο σημείο $s \in C$ για να ικανοποιήσει την $(i+1)$-οστή αίτηση $r \equiv r_{i+1}$, σχηματίζοντας μια νέα διάταξη $C' = C - s + r$. Ορίζουμε το τρέχον *offline ψευδοκόστος* ως εξής:

$$\text{offline ψευδοκόστος} \triangleq w_{\sigma_i r}(C') - w_{\sigma_i}(C) \ .$$

Το τηλεσκοπικό άθροισμα από τα offline ψευδοκόστη για όλη την ακολουθία σ ισούται με $w_\sigma(C_n) - w_\emptyset(C_0) = \text{κόστος}_{\text{OPT}}(\sigma)$.

Αφού $r \in C'$ και $w_{\sigma_i r}(C) = w_{\sigma_i}(C') + d(s,r)$, ισχύει

$$w_{\sigma_i r}(C') = w_{\sigma_i}(C') = w_{\sigma_i r}(C) - d(s,r) \ .$$

Άρα, αν αθροίσουμε το τρέχον offline ψευδοκόστος και το κόστος που υφίσταται ο WFA για την ικανοποίηση της αίτησης, λαμβάνουμε

$$\begin{aligned} w_{\sigma_i r}(C') - w_{\sigma_i}(C) + d(s,r) &= w_{\sigma_i r}(C) - d(s,r) - w_{\sigma_i}(C) + d(s,r) = \\ &= w_{\sigma_i r}(C) - w_{\sigma_i}(C) \leq \\ &\leq \max_X \{w_{\sigma_i r}(X) - w_{\sigma_i}(X)\} \ . \end{aligned} \tag{4.16}$$

Η ποσότητα (4.16) ονομάζεται *επαυξημένο κόστος* (*extended cost*). Έχουμε λοιπόν:

Λήμμα 4.7. *Έστω α σταθερά, ανεξάρτητη της ακολουθίας αιτήσεων σ. Αν το άθροισμα από τα επαυξημένα κόστη για όλες τις κινήσεις του αλγορίθμου* WFA *φράσσεται άνω από το $(c+1) \cdot \text{κόστος}_{\text{OPT}}(\sigma) + \alpha$, τότε ο τελευταίος είναι c-ανταγωνιστικός.*

Ειδικότερα, εν προκειμένω αρκεί να δείξουμε ότι

$$\text{συνολικό επαυξημένο κόστος} \le 2k \cdot \text{κόστος}_{\text{OPT}}(\sigma) + \alpha \; . \tag{4.17}$$

Τοιουτοτρόπως, δεν απαιτείται να ασχοληθούμε καθόλου με τις διατάξεις στις οποίες βρίσκεται μετά από κάθε βήμα ο WFA, απλουστεύοντας την αποδεικτική διαδικασία.

Μια συνάρτηση έργου w αποκαλείται *οιονεί κυρτή* (*quasi-convex*) αν για οποιεσδήποτε διατάξεις εξυπηρετητών X και Y και κάθε σημείο $x \in X$ ισχύει

$$\min_{y \in Y} \{w(X - x + y) + w(Y - y + x)\} \le w(X) + w(Y) \; .$$

Λήμμα 4.8 (Οιονεί Κυρτότητα). *Όλες οι συναρτήσεις έργου είναι οιονεί κυρτές.*

Έστω y κάποιο σημείο και w_{σ_i} η τρέχουσα συνάρτηση έργου. Μια διάταξη A λέγεται *ελαχιστοποιήτρια* (*minimizer*) του y ως προς την w_{σ_i} αν

$$A = \arg\min_X \left\{ w_{\sigma_i}(X) - \sum_{x \in X} d(x, y) \right\} \; .$$

Ακολούθως θεωρούμε ότι η επόμενη αίτηση αφορά το σημείο r. Με τη βοήθεια του Λήμματος 4.8 αποδεικνύονται δύο ακόμη λήμματα.

Λήμμα 4.9. *Αν w_{σ_i} η τρέχουσα συνάρτηση έργου και r το σημείο της επόμενης αίτησης, τότε*

$$(A \text{ είναι ελαχιστοποιήτρια του } r \text{ ως προς την } w_{\sigma_i}) \implies$$
$$\implies (A \text{ είναι ελαχιστοποιήτρια του } r \text{ ως προς την } w_{\sigma_i r}) \; .$$

Μια διάταξη A τέτοια ώστε

$$A = \arg\max_X \{w_{\sigma_i r}(X) - w_{\sigma_i}(X)\}$$

λέγεται *μεγιστοποιήτρια* (*maximizer*) ως προς την w_{σ_i}. Υπενθυμίζουμε εδώ ότι η ποσότητα $\max_X \{w_{\sigma_i r}(X) - w_{\sigma_i}(X)\}$ ισούται εξ ορισμού με το επαυξημένο κόστος.

Λήμμα 4.10 (Δυϊκότητα). *Κάθε ελαχιστοποιήτρια του r ως προς την w_{σ_i} είναι επίσης μεγιστοποιήτρια ως προς την w_{σ_i}.*

Έστω τώρα $U = \{u_1, u_2, \ldots, u_k\}$ η τρέχουσα διάταξη των εξυπηρετητών του βέλτιστου offline αλγορίθμου OPT. Ορίζουμε την παρακάτω συνάρτηση δυναμικού:

$$\Phi(U, w_{\sigma_i}) \triangleq \sum_{u \in U} \texttt{MIN}_{w_{\sigma_i}}(u) \; ,$$

όπου

$$\mathtt{MIN}_{w_{\sigma_i}}(u) \triangleq \min_X \left\{ w_{\sigma_i}(X) - \sum_{x \in X} d(x,u) \right\} .$$

Ας υποθέσουμε ότι ο OPT ικανοποιεί την επόμενη αίτηση r μετακινώντας τον εξυπηρετητή που βρίσκεται στο σημείο u_j. Τοιουτοτρόπως, η νέα διάταξη των εξυπηρετητών είναι $U' = U - u_j + r$. Έχουμε λοιπόν:

Λήμμα 4.11. $\Phi(U', w_{\sigma_i}) - \Phi(U, w_{\sigma_i}) \geq -k \cdot d(u_j, r)$.

Επιπροσθέτως, εκ των Λημμάτων 4.9 και 4.10 συνάγεται ότι:

Λήμμα 4.12. $\Phi(U', w_{\sigma_i r}) - \Phi(U', w_{\sigma_i}) \geq \max\limits_X \{w_{\sigma_i r}(X) - w_{\sigma_i}(X)\}$.

Χρησιμοποιώντας όλα τα παραπάνω επαληθεύεται η ανισότητα (4.17), αποδεικνύοντας το Θεώρημα 4.6. □

Σε αντίθεση με τους αλγορίθμους που παρουσιάσαμε στις προηγούμενες ενότητες, ο WFA είναι ανταγωνιστικός για κάθε μετρικό χώρο, πλησιάζοντας μάλιστα τον ελάχιστο δυνατό λόγο ανταγωνιστικότητας, ο οποίος ισούται με k. Εύλογα γεννάται λοιπόν η απορία εάν και υπό ποιες προϋποθέσεις ο αλγόριθμος συνάρτησης έργου μπορεί να επιτύχει την ιδανική αυτή επίδοση. Παραθέτουμε ορισμένα σχετικά αποτελέσματα:

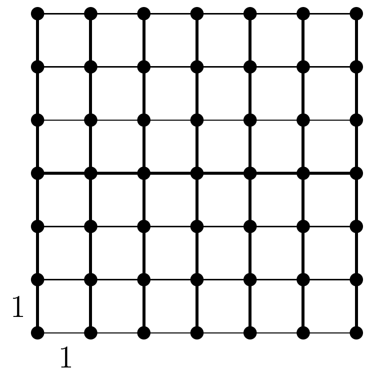
Θεώρημα 4.13. [CL92] *Στην ειδική περίπτωση που $k = 2$, ο αλγόριθμος* WFA *είναι 2-ανταγωνιστικός.*

Θεώρημα 4.14. [KoutP96] *Ο αλγόριθμος* WFA *είναι k-ανταγωνιστικός για κάθε μετρικό χώρο ο οποίος περιέχει $k+2$ σημεία το πολύ.*

Θεώρημα 4.15. [BeiCL99] *Στην ειδική περίπτωση που $k = 3$, ο αλγόριθμος* WFA *είναι 3-ανταγωνιστικός για το διδιάστατο επίπεδο Manhattan (τούτ' έστιν υπό την μετρική ℓ_1), άρα και $(3\sqrt{2})$-ανταγωνιστικός για το Ευκλείδιο διδιάστατο επίπεδο.*

Θεώρημα 4.16. [BarK00] *Ο αλγόριθμος* WFA *είναι k-ανταγωνιστικός για την ευθεία γραμμή.*

Τονίζεται ότι όλα τα προαναφερθέντα θεωρήματα στηρίζονται στις ιδιότητες της οιονεί κυρτότητας και της δυϊκότητας (Λήμματα 4.8 και 4.10). Γενικότερα, εικάζεται ότι ο WFA είναι ισχυρά ανταγωνιστικός για κάθε μετρικό χώρο. Υπάρχει επιπλέον έντονη υποψία ότι ο λόγος που δεν αποδεικνύεται κάτι τέτοιο από το Θεώρημα 4.6 ίσως πρέπει να αναζητηθεί στο ότι το επαυξημένο κόστος υπερεκτιμά το πραγματικό κόστος του αλγορίθμου.



Σχήμα 4.4: Εμβάπτιση $N \times N$ πλέγματος σε δένδρο (έντονες γραμμές)

4.4 Εμβαπτίσεις μετρικών χώρων

Ένα κλασικό μαθηματικό πρόβλημα συνίσταται στην εμβάπτιση ενός μετρικού χώρου σε κάποιον άλλο με μικρή παραμόρφωση, δηλαδή χωρίς να αλλοιώνεται δραματικά η τοπολογική μορφή του πρώτου. Εν προκειμένω, θα ασχοληθούμε με εμβαπτίσεις διότι μας επιτρέπουν να κατασκευάζουμε νέους ανταγωνιστικούς online αλγορίθμους για διάφορα προβλήματα, μεταξύ των οποίων και αυτό των k-εξυπηρετητών.

Ορισμός 4.17. *Έστω $\mathcal{M} = (M, d)$ και $\tilde{\mathcal{M}} = \left(M, \tilde{d}\right)$ δύο μετρικοί χώροι που αποτελούνται από το ίδιο σύνολο σημείων M. Ο $\tilde{\mathcal{M}}$ α-προσεγγίζει τον $\mathcal{M}$ αν για κάθε $u, v \in M$ ισχύει $d(u, v) \leq \tilde{d}(u, v) \leq \alpha \cdot d(u, v)$.*

Σημειωτέον ότι αναγκαία συνθήκη για να ικανοποιείται ο παραπάνω ορισμός είναι οι μετρικές d και $\tilde{d}$ να είναι *ισοδύναμες*. Άμεσα λοιπόν προκύπτει το ακόλουθο θεώρημα.

Θεώρημα 4.18. *Έστω $\mathcal{M} = (M, d)$ και $\tilde{\mathcal{M}} = \left(M, \tilde{d}\right)$ δύο μετρικοί χώροι που αποτελούνται από το ίδιο σύνολο σημείων M, ώστε ο $\tilde{\mathcal{M}}$ να α-προσεγγίζει τον $\mathcal{M}$. Αν υπάρχει c-ανταγωνιστικός αιτιοκρατικός (αντιστοίχως, randomized) αλγόριθμος $\text{ALG}_{\tilde{\mathcal{M}}}$ για το πρόβλημα των k-εξυπηρετητών στον $\tilde{\mathcal{M}}$, τότε υπάρχει αιτιοκρατικός (αντιστοίχως, randomized) αλγόριθμος $\text{ALG}_{\mathcal{M}}$ για το ίδιο πρόβλημα στον $\mathcal{M}$ ο οποίος είναι $c\alpha$-ανταγωνιστικός.*

Απόδειξη. Προφανής. $\square$

Παράδειγμα. Έστω το $N \times N$ πλέγμα του Σχήματος 4.4. Οι k εξυπηρετητές μπορούν να κινούνται μόνο κατά μήκος των ακμών του πλέγματος, ενώ οι αιτήσεις αφορούν αποκλειστικά κόμβους του τελευταίου. Με άλλα λόγια, ο μετρικός χώρος στον οποίο εξετάζουμε το πρόβλημα των k-εξυπηρετητών έχει πληθικότητα N^2 και χρησιμοποιεί την μετρική ℓ_1, γνωστή και ως μετρική Manhattan.

Θεωρούμε ένα παράγον δένδρο (spanning tree) των κόμβων του πλέγματος, όπως φαίνεται από τις έντονες γραμμές στο σχήμα, και εφαρμόζουμε τον αλγόριθμο DC-TREE$^\ell$, δηλαδή την οκνηρή εκδοχή του DC-TREE, για την ικανοποίηση των αιτήσεων που διατυπώνονται. Φυσικά, εδώ οι εξυπηρετητές έχουν τη δυνατότητα να διασχίζουν όλες τις ακμές του πλέγματος, όχι μόνον εκείνες οι οποίες ανήκουν στο παράγον δένδρο, εφόσον έτσι «κόβουν δρόμο» και μειώνουν το online κόστος.

Εύκολα διαπιστώνουμε ότι το συγκεκριμένο δένδρο, ιδωμένο ως μετρικός χώρος, $\left(2\left\lfloor\frac{N}{2}\right\rfloor+1\right)$-προσεγγίζει το πλέγμα διαστάσεων $N \times N$. Συνεπώς, ο προαναφερθείς νέος αλγόριθμος που προκύπτει ως επέκταση του DC-TREE στο πλέγμα αποδεικνύεται γενικά $k\left(2\left\lfloor\frac{N}{2}\right\rfloor+1\right)$-ανταγωνιστικός.

Δυστυχώς, η παραπάνω μέθοδος δεν παρέχει ιδιαίτερη ευελιξία στην κατασκευή νέων αλγορίθμων και τα αποτελέσματα, ως προς την ανταγωνιστικότητα, κάθε άλλο παρά θεαματικά είναι. Για το λόγο αυτό, προχωρούμε ένα βήμα παραπέρα στη γενικότερη έννοια των *πιθανοτικών εμβαπτίσεων* (*probabilistic embeddings*).

Ορισμός 4.19. *Έστω $\mathcal{M}=(M,d)$ μετρικός χώρος και $\mathcal{N}=\left(M,d^{(x)}\right)$ κλάση μετρικών χώρων οι οποίοι ορίζονται όλοι επί του συνόλου σημείων M. Η $\mathcal{N}$ λ-προσεγγίζει πιθανοτικά τον $\mathcal{M}$ αν υπάρχει κατανομή πιθανότητας $\mathcal{D}(x)$ επί της $\mathcal{N}$ τέτοια ώστε για κάθε $u,v \in M$:*

1. $d(u,v) \leq d^{(x)}(u,v)$.
2. $E_{\mathcal{D}(x)}\left[d^{(x)}(u,v)\right] \leq \lambda \cdot d(u,v)$.

Κατ' αναλογία με το Θεώρημα 4.18 έχουμε εύλογα:

Θεώρημα 4.20. *Έστω $\mathcal{M}=(M,d)$ μετρικός χώρος και $\mathcal{N}=\left(M,d^{(x)}\right)$ κλάση μετρικών χώρων οι οποίοι ορίζονται όλοι επί του συνόλου σημείων M, ώστε η $\mathcal{N}$ να λ-προσεγγίζει πιθανοτικά τον $\mathcal{M}$. Αν για το πρόβλημα των k-εξυπηρετητών σε κάθε έναν χώρο $\tilde{\mathcal{M}} \in \mathcal{N}$ υπάρχει c-ανταγωνιστικός αιτιοκρατικός ή randomized αλγόριθμος $\mathrm{ALG}_{\tilde{\mathcal{M}}}$, τότε υπάρχει randomized αλγόριθμος $\mathrm{ALG}_{\mathcal{M}}$ για το ίδιο πρόβλημα στον $\mathcal{M}$ που είναι $c\lambda$-ανταγωνιστικός εναντίον αδαών αντιπάλων.*

Ο καινούριος αλγόριθμος $\mathrm{ALG}_{\mathcal{M}}$ λειτουργεί ως εξής. Κατ' αρχάς, επιλέγει έναν μετρικό χώρο $\tilde{\mathcal{M}} \in \mathcal{N}$ με τυχαίο τρόπο, σύμφωνα με την κατανομή $\mathcal{D}(x)$ για την οποία η $\mathcal{N}$ λ-προσεγγίζει πιθανοτικά τον $\mathcal{M}$. Κατόπιν, εμβαπτίζει τον $\mathcal{M}$ στον $\tilde{\mathcal{M}}$ και απλά εφαρμόζει τον αλγόριθμο $\mathrm{ALG}_{\tilde{\mathcal{M}}}$.

Παρατήρηση. Το γεγονός ότι ο αντίπαλος είναι αδαής αποτελεί θεμελιώδη απαίτηση του προηγούμενου θεωρήματος, διότι η αναμενόμενη επίδοση του $\text{ALG}_{\mathcal{M}}$ είναι εγγυημένη αν και μόνο αν ο αντίπαλος δεν γνωρίζει ποιο χώρο $\tilde{\mathcal{M}} \in \mathcal{N}$ επιλέγει ο αλγόριθμος. Καταφανώς, αν ο αντίπαλος είναι προσαρμοστικός, το Θεώρημα 4.20 καταρρέει.

Παράδειγμα. Συναντήσαμε ήδη τον randomized αλγόριθμο CIRC στην Υποενότητα 4.1.3, ο οποίος κατασκευάζεται εμβαπτίζοντας πιθανοτικά την περιφέρεια κύκλου $\mathcal{C}$ σε μια συγκεκριμένη κλάση ευθυγράμμων τμημάτων $\mathcal{L}$, υπεραριθμήσιμων το πλήθος. Μάλιστα, έχοντας κατά νου τον Ορισμό 4.19, διαπιστώνουμε ότι η σχέση (4.3) αποδεικνύει πως η $\mathcal{L}$ 2-προσεγγίζει πιθανοτικά την $\mathcal{C}$.

Όπως προαναφέραμε, η πρωτοποριακή (και αδημοσίευτη!) εργασία του Karp στην οποία παρουσιάζεται και αναλύεται ο αλγόριθμος CIRC [Karp89] απετέλεσε το έναυσμα για διεξοδικότερη μελέτη των πιθανοτικών εμβαπτίσεων και των εφαρμογών τους σε αλγοριθμικά προβλήματα. Ειδικότερα, όσον αφορά την περιοχή των άμεσων αλγορίθμων, γρήγορα το ενδιαφέρον των ερευνητών στράφηκε στην αναζήτηση κλάσεων από δενδρικούς μετρικούς χώρους (tree metrics) οι οποίες να προσεγγίζουν πιθανοτικά έναν οποιονδήποτε μετρικό χώρο με μικρή παραμόρφωση. Η προτίμηση αυτή είναι δικαιολογημένη, διότι πολλά προβλήματα σε δένδρα επιδέχονται κομψές και αποτελεσματικές λύσεις – όπως για παράδειγμα εκείνο των k-εξυπηρετητών, που επιλύεται από τον DC-TREE.

Έτσι λοιπόν, οι Alon, Karp, Peleg και West [AlKPW95] εξέτασαν πιθανοτικές εμβαπτίσεις γραφημάτων σε παράγοντα δένδρα. Εκ των αποτελεσμάτων τους συνάγεται ότι κάθε μετρικός χώρος $\mathcal{M}$ $\left(2^{O\left(\sqrt{\log n \log\log n}\right)}\right)$-προσεγγίζεται πιθανοτικά από δενδρικούς μετρικούς χώρους, όπου n το πλήθος σημείων του $\mathcal{M}$. Αργότερα, ο Bartal βελτίωσε τον παράγοντα προσέγγισης, πρώτα σε $O(\log^2 n)$ [Bar96] και εν συνεχεία σε $O(\log n \log\log n)$ [Bar98]. Για να το κατορθώσει αυτό, όρισε την λεγόμενη κλάση των *ιεραρχικά καλώς διαχωρισμένων δένδρων* (*hierarchically well-separated trees, HSTs*). Επιπλέον, διαπίστωσε ότι υπάρχει μετρικός χώρος ο οποίος αν λ-προσεγγιστεί πιθανοτικά από δενδρικούς μετρικούς χώρους, τότε $\lambda = \Omega(\log n)$ [Bar96].

Εν κατακλείδι, σημειώνουμε ότι προσφάτως οι Fakcharoenphol, Rao και Talwar [FaRT03] απέδειξαν ότι κάθε μετρικός χώρος $O(\log n)$-προσεγγίζεται από HSTs. Δηλαδή, λαμβάνοντας υπ' όψη το ανωτέρω κάτω φράγμα του Bartal, επέτυχαν τον βέλτιστο παράγοντα προσέγγισης modulo μια πολλαπλασιαστική σταθερά.

Βιβλιογραφικές Αναφορές

[AcCN96] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. In *Proceedings of the 4^{th} European Symposium on Algorithms*, volume 1136 of *Lecture Notes in Computer Science*, pages 419–430. Springer-Verlag, 1996. (Journal version [AcCN00]).

[AcCN00] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1-2):203–218, 2000.

[AlKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West:. A graph-theoretic game and its application to the k-server problem. *SIAM Journal of Computing*, 24(1):78–100, 1995.

[AzBM93] Yossi Azar, Andrei Z. Broder, and Mark S. Manasse. Online choice of online algorithms. In *Proceedings of the 4^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 432–440, 1993.

[BaeYCR88] Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins. Searching in the plane. In *Proceedings of the 1^{st} Scandinavian Workshop on Algorithm Theory*, volume 318 of *Lecture Notes in Computer Science*, pages 176–189, 1988. (Journal version [BaeYCR93]).

[BaeYCR93] Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.

[Bar94] Yair Bartal. A fast memoryless 2-server algorithm in Euclidean spaces. Unpublished manuscript, 1994.

[Bar96] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37^{th}*

Annual IEEE Symposium on Foundations of Computer Science, pages 184–193, 1996.

[Bar98] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[BarCL98] Yair Bartal, Marek Chrobak, and Lawrence L. Larmore. A randomized algorithm for two servers on the line. In *Proceedings of the 6th European Symposium on Algorithms*, pages 247–258, 1998. (Journal version [BarCL00]).

[BarCL00] Yair Bartal, Marek Chrobak, and Lawrence L. Larmore. A randomized algorithm for two servers on the line. *Information and Computation*, 158(1):53–69, 2000.

[BarK00] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. In *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 605–613, 2000.

[BeiCL99] Wolfgang W. Bein, Marek Chrobak, and Lawrence L. Larmore. The 3-server problem in the plane. In *Proceedings of the 7th European Symposium on Algorithms*, pages 301–312, 1999. (Journal version [BeiCL02]).

[BeiCL02] Wolfgang W. Bein, Marek Chrobak, and Lawrence L. Larmore. The 3-server problem in the plane. *Theoretical Computer Science*, 289(1):335–354, 2002.

[BenDBK+90] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 379–386, 1990. (Journal version [BenDBK+94]).

[BenDBK+94] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[BerGR93] Marshall W. Bern, Daniel H. Greene, and Arvind Ragunathan. Online algorithms for cache sharing. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 422–430, 1993.

[BorEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[BorLS87] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal online algorithm for metrical task systems. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 373–382, 1987. (Journal version [BorLS92]).

[BorLS92] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal online algorithm for metrical task systems. *Journal of the ACM*, 39:745–763, 1992.

[Bur93] William R. Burley. Traversing layered graphs using the work function algorithm. Technical Report CS93-319, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, 1993. (Journal version [Bur96]).

[Bur96] William R. Burley. Traversing layered graphs using the work function algorithm. *Journal of Algorithms*, 20(3):479–511, 1996.

[CKPV90] Marek Chrobak, Howard J. Karloff, Thomas H. Payne, and Sundar Vishwanathan. New results on server problems. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 291–300, 1990. (Journal version [CKPV91]).

[CKPV91] Marek Chrobak, Howard J. Karloff, Thomas H. Payne, and Sundar Vishwanathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991.

[CL91a] Marek Chrobak and Lawrence L. Larmore. On fast algorithms for two servers. *Journal of Algorithms*, 12:607–614, 1991.

[CL91b] Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.

[CL92] Marek Chrobak and Lawrence L. Larmore. The server problem and online games. In Lyle A. McGeoch and Daniel D. Sleator, editors, *On-Line Algorithms*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 11–64, 1992.

[FaRT03] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 448–455, 2003.

[FiFK+91] Amos Fiat, Dean P. Foster, Howard J. Karloff, Yuval Rabani, Yiftach Ravid, and Sundar Vishwanathan. Competitive algorithms for layered graph traversal. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 288–297, 1991. (Journal version [FiFK+98]).

[FiFK+98] Amos Fiat, Dean P. Foster, Howard J. Karloff, Yuval Rabani, Yiftach Ravid, and Sundar Vishwanathan. Competitive algorithms for layered graph traversal. *SIAM Journal of Computing*, 28(2):447–462, 1998.

[FiKL+88] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel D. Sleator, and Neal E. Young. Competitive paging algorithms. Technical Report CMU-CS-88-196, Carnegie Mellon University, Pittsburgh, 1988. (Journal version [FiKL+91]).

[FiKL+91] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel D. Sleator, and Neal E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[FiRR90] Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 454–463, 1990.

[FiRRS94] Amos Fiat, Yuval Rabani, Yiftach Ravid, and Baruch Schieber. A deterministic $O(k^3)$-competitive k-server algorithm for the circle. *Algorithmica*, 11(6):572–578, 1994.

[FiW98] Amos Fiat and Gerhard J. Woeginger, editors. *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, 1979.

[Gra66] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[IK97] Sandy Irani and Anna R. Karlin. Online computation. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 13, pages 521–564. PWS Publishing Company, 1997.

[IR91] Sandy Irani and Ronitt Rubinfeld. A competitive 2-server algorithm. *Information Processing Letters*, 39(2):85–91, 1991.

[KalyP91] Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. In *Proceedings of the 2^{nd} Annual ACM-SIAM Symposium on Discrete Algorithms*, 1991. (Journal version [KalyP93]).

[KalyP93] Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14:478–488, 1993.

[KarlMMO90] Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan S. Owicki. Randomized competitive algorithms for non-uniform problems. In *Proceedings of the 1^{st} Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 301–309, 1990. (Journal version [KarlMMO94]).

[KarlMMO94] Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan S. Owicki. Randomized competitive algorithms for non-uniform problems. *Algorithmica*, 11(6):542–571, 1994.

[KarlMRS88] Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[Karp89] Richard M. Karp. A $2k$-competitive algorithm for the circle. Unpublished manuscript, 1989.

[KhulMV91] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. In *Proceedings of the 18^{th} International Colloquium on Automata, Languages, and Programming*, volume 510 of *Lecture Notes in Computer Science*, pages 728–738. Springer-Verlag, 1991. (Journal version [KhulMV94]).

[KhulMV94] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. Online algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.

[Klei94] Jon M. Kleinberg. A lower bound for two-server balancing algorithms. *Information Processing Letters*, 1994.

[KoutP94] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. In *Proceedings of the 26^{th} Annual ACM Symposium on Theory of Computing*, pages 507–511, 1994. (Journal version [KoutP95]).

[KoutP95] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42:971–983, 1995.

[KoutP96] Elias Koutsoupias and Christos H. Papadimitriou. The 2-evader problem. *Information Processing Letters*, 57:249–252, 1996.

[LR94] Carsten Lund and Nick Reingold. Linear programs for randomized online algorithms. In *Proceedings of the 5^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 382–391, 1994.

[MMS88] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for online problems. In *Proceedings of the 20^{th} Annual ACM Symposium on Theory of Computing*, pages 322–333, 1988. (Journal version [MMS90]).

[MMS90] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for online problems. *Journal of Algorithms*, 11:208–230, 1990.

[PY89] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. In *Proceedings of the 16^{th} International Colloquium on Automata, Languages, and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 610–620. Springer-Verlag, 1989. (Journal version [PY91]).

[PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.

[RS89] Prabhakar Raghavan and Marc Snir. Memory versus randomization in on-line algorithms. In *Proceedings of the 16^{th} International Colloquium on Automata, Languages, and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 687–703. Springer-Verlag, 1989.

[ST85a] Daniel D. Sleator and Richard E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[ST85b] Daniel D. Sleator and Richard E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32:652–686, 1985.

[Tar83] Richard E. Tarjan. *Data Structures and Network Algorithms*. SIAM, Philadelphia, 1983.

[Tar85] Richard E. Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods*, 6(2):306–318, 1985.

[Yao77] Andrew C. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the 18^{th} Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.