Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Άμεσοι Αλγόριθμοι για Προβλήματα Χωροθέτησης Υποβοηθούμενοι από Τεχνικές Μάθησης

Διπλωματικη Εργασια

του

ΠΑΤΡΗ ΝΙΚΟΛΑΟΥ

Επιβλέπων: Φωτάκης Δημήτριος
Αναπληρωτής Καθηγητής

Αθήνα, Ιούλιος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Άμεσοι Αλγόριθμοι για Προβλήματα Χωροθέτησης Υποβοηθούμενοι από Τεχνικές Μάθησης

Διπλωματικη Εργασια

του

ΠΑΤΡΗ ΝΙΚΟΛΑΟΥ

Επιβλέπων: Φωτάκης Δημήτριος
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Ιουλίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Φωτάκης Δημήτριος Αναπληρωτής Καθηγητής	Παγουρτζής Αριστείδης Καθηγητής	Αγγελόπουλος Σπυρίδων Researcher CNRS

Αθήνα, Ιούλιος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Πατρής Νικόλαος

Ιούλιος 2021

Περίληψη

Στην παρούσα διπλωματική μελετάμε μια νέα εκδοχή του άμεσου προβλήματος χωροθέτησης. Το συγκεκριμένο πρόβλημα αποτελεί ένα κλασικό πρόβλημα συνδυαστικής βελτιστοποίησης και αποτελεί αντικείμενο έντονης ερευνητικής μελέτης τις τελευταίες δεκαετίες.

Ωστόσο, οι ραγδαίες εξελίξεις στον τομέα της μηχανικής μάθησης οδήγησαν στην δημιουργία ενός νέου πεδίου έρευνας όπου πέραν της άμεσης φύσης των δεδομένων, υποθέτουμε την ύπαρξη ενός μαντείου προβλέψεων. Οι προβλέψεις παρέχουν επιπρόσθετη πληροφορία, την οποία εκμεταλλεύονται οι αλγόριθμοι προκειμένου να ξεπεράσουν τα κλασικά φράγματα του λόγου ανταγωνιστικότητας.

Στην νέα εκδοχή - υποβοηθούμενη από μηχανική μάθηση - διατυπώνουμε έναν αλγόριθμο που συνδυάζει τις ενδεχομένως ατελείς προβλέψεις του μαντείου και πετυχαίνει σταθερό λόγο ανταγωνιστικότητας. Επιπλέον, διατυπώνουμε τα απαραίτητα κάτω φράγματα που αποδεικνύουν τόσο την καταλληλότητα του μαντείου όσο και την αδυναμία άλλων σφαλμάτων να οδηγήσουν σε σταθερό λόγο ανταγωνιστικότητας.

Λέξεις Κλειδιά

Άμεσοι Αλγόριθμοι, Λόγος Ανταγωνιστικότητας, Πρόβλημα Χωροθέτησης Εγκαταστάσεων, Μηχανική Μάθηση, Μαντείο, Πρόβλεψη

Abstract

In this dissertation we study a new version of the Online Facility Location problem, which is a classic combinatorial optimization problem extensively studied in the past decades.

However, the recent rapid developments in the field of Machine Learning have led to the birth of a new field called learning augmented algorithms. Apart from the online fashion of the input, an oracle provides additional information to the algorithm, concerning the sequence of input items. This additional information, termed advice, can boost the performance of the algorithm, which is reflected in better competitive ratios.

Based on this model of computation - learning augmented model- we present an algorithm for the online facility location problem which incorporates the possible imperfect advices provided by the oracle and achieves constant competitive ratio. We complement the analysis with the statements proving the appropriateness of the oracle and the inability of other errors to achieve constant competitive ratio.

Λέξεις Κλειδιά

Online Algorithms, Competitive Ratio, Facility Location Problem, Machine Learning, Oracle, Prediction

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Το Πρόβλημα Χωροθέτησης Εγκαταστάσεων

Το Πρόβλημα Χωροθέτησης Εγκαταστάσεων (Facility Location Problem) αποτελεί ένα από τα βασικότερα και πιο εκτενώς μελετημένα προβλήματα τόσο στο πεδιό της Επιστήμης των Υπολογιστών (Computer Science) όσο και της Επιχειρησιακής Έρευνας (Operations Research). Η παρούσα εργασία επικεντρώνεται στην βασικότερη και πιο διαδεδομένη εκδοχή του προβλήματος συγκεκριμένα στην *uniform metric uncapacitated facility location problem*. Πρακτικά ο χώρος του πρόβληματος είναι ένας μετρικός χώρος *(metric space)*, δηλαδή ένα σύνολο σημείων με αποστάσεις που ικανοποιούν την τριγωνική ανισότητα, τα κόστη κατασκευής νέων εγκαταστάσεων είναι ίδια *(uniform)* για κάθε σημείο του χώρου και τέλος κάθε εγκατάσταση εξυπηρετεί έναν αυθαίρετο αριθμό από αιτήματα *(uncapacitated)*.

Η είσοδος του προβλήματος αποτελείται από έναν μετρικό χώρο, το κόστος κατασκευής νεών εγκαταστάσεων και ένα πολυσύνολο [1] αιτημάτων σε διάφορα σημεία του μετρικού χώρου. Ο στόχος είναι η εύρεση ενός συνόλου τοποθεσιών για τις νέες εγκαταστάσεις που ελαχιστοποιούν το συνολικό κόστος κατασκευής τους, καθώς και το κόστος σύνδεσης των αιτημάτων, δηλαδή τις αποστάσεις τους ως προς τα κοντινότερα κέντρα [2].

Το πρόβλημα του facility location αποτελεί ένα απλό και φυσικό μοντέλο για μια μεγάλη κλάση εφαρμογών [2]: στη σχεδίαση κατανεμημένων συστημάτων (distribution systems) [3], σε ασύρματα δίκτυα αισθητήρων [4], στην όραση υπολογιστών (computer vision) [5],[6] και είναι αντικείμενο έντονης ερευνητικής δραστηριότητας. στην συνδυαστική βελτιστοποίηση, στην επιχειρησιακή έρευνα και στην επιστήμη των υπολογιστών από την σκοπιά των προσεγγιστικών αλγορίθμων Το πρόβλημα ανήκει στην κλάση πολυπλοκότητας MAX-SNP Hard και ο πρώτος αλγόριθμος με σταθερό

[1] Ένα πολυσύνολο είναι μία συλλογή στοιχείων, με περισσότερα από ένα πιθανά αντίγραφα για κάθε στοιχείο του συνόλου.

[2] Οι όροι *κέντρο* και *εγκατάσταση* είναι ισοδύναμοι και χρησιμοποιούνται εκ περιτροπής.

λόγο προσέγγισης ισό με 3.16, διατυπώθηκε από τους Shmoys, Tardos και Aardal [7]. Έπειτα, βελτιώθηκε σε 1.728 από τους Charikar και Guha [8] και στην συνέχεια σε 1.67 από τον Sviridenko [9]. Ο βέλτιστος, ως τώρα, πολυωνυμικός αλγόριθμος οφείλεται στους Byrka και Aardal [10] και πετυχαίνει λόγο προσέγγισης ίσο με 1.5, ενώ δεν υπάρχει αλγόριθμος με λόγο προσέγγισης μικρότερο από 1.463 εκτός και αν $NP \subseteq DTIME(n^{\log\log n})$ [11].

Πρακτικά, κάθε γνωστή τεχνική για τον σχεδιασμό και την ανάλυση προσεγγιστικών αλγορίθμων έχει εφαρμοστεί στο πρόβλημα χωροθέτησης εγκαταστάσεων (facility location). Υπάρχουν αλγόριθμοι σταθερού λόγου προσέγγισης βασισμένοι στην στρογγύλευση (rounding) του γραμμικού προβλήματος [7], σε μεθόδους τοπικής αναζήτησης (local search) [12], στην τεχνική του πρωτεύοντος-δυϊκού (primal-dual method) [13] ή σε άπληστους αλγορίθμους με διαφορετικά κριτήρια βελτιστότητας και τεχνικές ανάλυσης, βλπ [14],[15],[16].

Ωστόσο σε πολλές πραγματικές εφαρμογές η υπόθεση ότι τα δεδομένα εισόδου (αιτήματα) είναι εκ των προτέρων γνωστά, δεν είναι ιδιαίτερα ρεαλιστική. Παραδείγματος χάριν, ας υποθέσουμε ότι μας ανατίθεται η κατασκευή ενός νέου δικτύου. Σε πρώτη φάση θα πρέπει να αγοράσουμε ορισμένους servers ώστε να τους συνδέσουμε με τους υπάρχοντες clients. Το κόστος σύνδεσης σε αυτήν την περίπτωση ισούται με το κόστος του καλωδίου που τους συνδέει. Μετά την δημιουργία του δικτύου, ενδέχεται κάποιοι νέοι clients να επιθυμούν να συνδεθούν επίσης. Συνεπώς, θα πρέπει να αγοράσουμε επιπλέον καλώδιο ή ακόμα και νέους servers ώστε να προσαρμοστούμε στην αυξανόμενη ζήτηση. Πάλι, ο στόχος μας είναι να ελαχιστοποιήσουμε το συνολικό κόστος.

Ως εναλλακτικό παράδειγμα, ας θεωρήσουμε το πρόβλημα ομαδοποίησης (clustering) του διαδικτύου. Με την χρήση διαφόρων χαρακτηριστικών, μπορούμε να απεικονίσουμε τις σελίδες σε έναν νέο χώρο περιεχομένου (content space), ώστε ο διαχωρισμός τους σε ομάδες να γίνει αποτελεσματικά. Φυσικά, οι σελίδες κάθε ομάδας πρέπει να βρίσκονται *σχετικά* κοντά στον νέο χώρο, δηλαδή να εμφανίζουν κοινά στοιχεία ως προς περιεχόμενο τους. Αντιστοίχως, επιθυμούμε να αποφύγουμε την δημιουργία πολλών ομάδων, διότι τότε η ομαδοποίηση δεν αναδεικνύει τα κοινά τους χαρακτηριστικά. Παρ΄ όλα αυτά με την πάροδο του χρόνου και όσο περισσότερες σελίδες κατασκευάζονται είτε συνδέονται σε υπάρχουσες ομάδες είτε δημιουργούν νέες. Πάλι ο στόχος μας παραμένει κοινός: να διατηρήσουμε μια καλή ομαδοποίηση χωρίς να χωρίζουμε ανεξέλεκτα τις υπάρχουσες ομάδες κάθε φορά που κάποια νέα σελίδα προκύπτει.

Ο Meyerson [17] εισήγαγε την άμεση εκδοχή του προβλήματος χωροθέτησης εγκαταστάσεων (online facility location - ofl) και διατύπωσε έναν απλό πιθανοτικό αλγόριθμο με σταθέρο competitive ratio σε τυχαίες μεταθέσεις της ακολουθίας των αιτημάτων. Παρ΄όλα αυτά, ενάντια σε adversarial είσοδο, ο αλγόριθμος έχει λογαριθμικό $O(\log n)$ competitive ratio [3]. Λίγο μετά, ο Fotakis [19] απέδειξε το πρώτο κάτω φράγ-

[3]Ωστόσο, με πιο προσεκτική ανάλυση ο Fotakis[18] έδειξε ότι ο αλγόριθμος του Meyerson πετυχαίνει πράγματι το βέλτιστο $O(\frac{\log n}{\log\log n})$ competitive ratio

μα $\Omega(\frac{\log n}{\log \log n})$ και διατύπωσε έναν βέλτιστο, αλλά με υψηλή χρονική πολυπλοκότητα, ντετερμινιστικό αλγόριθμο. Ύστερα οι Anagnostopoulos et al. [20] διατύπωσαν έναν ντετερμινιστικό αλγόριθμο με $\Theta(2^d \log n)-$competitive ratio σε ευκλείδειους χώρους d διαστάσεων. Ο αλγόριθμος τους εντοπίζει τον υπερκύβο d διαστάσεων και πλευράς α με συσσωρευμένο κόστος σύνδεσης μεγαλύτερο από f στον οποίο θα κατασκευάστει το νέο κέντρο. Έπειτα, χωρίζει τον υπερκύβο σε 2^d νεόυς υπερκύβους (quadrants) με πλευρές $\alpha/2$. Έτσι, εντοπίζοντας τις περιοχές που εμφανίζουν έντονη συσσωρεύση και κατασκευάζοντας νέα κέντρα, εξυπηρετεί τα μελλοντικά αιτήματα με μικρότερο κόστος σύνδεσης. Σε Ευκλείδειους χώρους με μικρές διαστάσεις, ο αλγόριθμος είναι και εύκολος στην υλοποίηση αλλά και ταχύτατος στην εκτέλεση. Παρ΄ όλα αυτά, δεν μπορεί να εξασφαλίσει το βέλτιστο competitive ratio για την περίπτωση των γενικών μετρικών χώρων.

Το τυπικό μοντέλο του online computation [21] υποθέτει ότι ο αλγόριθμος δεν έχει καμία γνώση σχετικά με τα δεδομένα εισόδου. Παρόλα αυτά, η υπόθεση αυτή δεν είναι ιδιαίτερα ρεαλιστική σε πραγματικά προβλήματα. Ο αλγόριθμος συχνά γνωρίζει το μέγεθος της είσοδου είτε ακόμα και ορισμένα από τα μελλοντικά στοιχεία. Για παράδειγμα, ας θεωρήσουμε ένα στιγμιότυπο του προβλήματος χωροθέτησης εγκαταστάσεων. Έστω, ότι βρισκόμαστε στην επιτροπή για την κατασκευή νέων ιατρικών κέντρων (εγκαταστάσεις) σε ένα *σχετικά* απομακρυσμένο σημείο της χώρας. Οι πολίτες των γύρω περιοχών μπορούν να ψηφίσουν για τις νέες τοποθεσίες κατασκευής. Ανεξάρτητα όμως από τον αριθμό των ψήφων και τις προτεινόμενες νέες θέσεις των κέντρων, γνωρίζουμε *προσεγγιστικά* τον πληθυσμό των γύρω περιοχών (δημογραφικά στοιχεία) και έτσι μπορούμε να ξεκινήσουμε την διαδικάσια, κατασκευάζοντας πρώτα ιατρικά κέντρα στα σημεία όπου ο πληθυσμός είναι πιο αυξημένος. Τυπικά, οι πολίτες φαντάζουν σαν μια κατανομή στον ευκλείδειο χώρο τόσο με κέντρα έντονης συσσώρευσης (πληθυσμού) όσο και με αραιώματα.

Τα δύο γνωστά μοντέλα που λαμβάνουν υπόψιν τους το παραπάνω σενάριο είναι το *advice complexity* model και το *machine learned predictions* model. Το πρώτο μοντελό, αν και όχι τόσο χρήσιμο λόγω των περιορισμών που επιβάλλει, προσφέρει μια information-theoretic οπτική μέλετης της πληροφορίας. Στην παρούσα εργασία χρησιμοποιείται στην απόδειξη ενός κάτω φράγματος για την ελάχιστη πληροφορία που οδηγεί σε σταθερό competitive ratio. Το δεύτερο μοντέλο αποτελεί μια πιο πρόσφατη προσεγγίση μελέτης και σχεδίασης online αλγορίθμων που οφείλεται στις ραγδαίες εξελίξεις στους τομείς της μηχανική μάθηση (machine learning), βλπ [22],[23]. Εδώ, ο αλγόριθμος υποθέτει την ύπαρξη ενός προβλέπτη (oracle/predictor) που προσφέρει επιπρόσθετη πληροφορία σχετικά με τα δεδομένα εισόδου. Ωστόσο το oracle θεωρείται ότι (1) περιέχει κάποια σφάλματα (noisy) και (2) η πληροφορία που παρέχει είναι εύλογη/ρεαλιστική, π.χ. δεν αφορά άμεσα την βέλτιστη λύση.

1.2 Συνεισφορά

Στην παρούσα εργασία μελετήσαμε για πρώτη φορά μια νέα εκδοχή του online facility location (υπό το machine learned predictions μοντέλο), συγκεκριμένα την learning augmented εκδοχή (υποβοηθούμενη από μηχανική μάθηση). Πέραν λοιπόν της online φύσης των δεδομένων, υποθέτουμε την ύπαρξη ένος μοντέλου προβλέψεων, oracle, που παρέχει στον αλγόριθμο μια επιπρόσθετη πληροφορία αναφορικά με την ακολουθία των αιτημάτων. Ο χώρος του προβλήματος, δηλαδή εκεί που κατασκευάζονται νέα κέντρα και βρίσκονται τα αιτήματα, δεν είναι ένας γενικός μετρικός χώρος, αλλά μια ειδική κατηγορία του, ονόματι Hierarchical Well-Separated Tree (HST) [24]. Πρόκειται για ένα πλήρες δυαδικό δέντρο με γεωμετρικά μειούμενες αποστάσεις μεταξύ των διαδοχικών επιπέδων. Τα αιτήματα επιτρέπεται να βρίσκονται σε οποιοδήποτε κόμβο του δέντρου, ωστόσο τα κέντρα κατασκευάζονται αποκλειστικά στα φύλλα.

Ο τρόπος λειτουργίας του αλγορίθμου είναι αρκετά απλός. Κατά την άφιξη ενός νέου αιτήματος καλείται να αποφασίσει αν θα συνδέσει το αίτημα με ένα υπάρχον κέντρο ή αν πρώτα θα κατασκευάσει ένα νέο στην γύρω περιοχή. Η πρώτη περίπτωση είναι προφανώς τετριμμένη, αφού το αίτημα συνδέεται απλά με το κοντινότερο κέντρο. Αντιθέτως, η δεύτερη περίπτωση είναι σαφώς πιο δύσκολη και ουσιαστική. Η επιλογή της τοποθεσίας του νέου κέντρου θα καθορίσει το μελλοντικό κόστος σύνδεσης των αιτημάτων. Για την ακρίβεια, είναι η μόνη απόφαση που καλείται να πάρει ο αλγόριθμος και άρα η μόνη που επηρεάζει το competitive ratio. Συνεπώς, ο στόχος του είναι να εντοπίσει, όσο το δυνατό καλύτερα, την θέση του βέλτιστου κέντρου.

Προκειμένου να το καταφέρει κινείται σε δύο επίπεδα. Πρώτα, λειτουργεί ως κλασσικός online αλγόριθμος. Τυπικά, κάθε αλγόριθμος [18] για το online facility location διαθέτει ως μοναδική πηγή πληροφορίας την θέση των ίδιων των αιτημάτων. Εν μέρει, τα αιτήματα είναι εκείνα που καθορίζουν την επίδοση του. Ένα απλό παράδειγμα είναι οι περιοχές έντονης συσσώρευσης. Οι συγκεκριμένες περιοχές μαρτυρούν αυτομάτως τις θέσεις των βέλτιστων κέντρων και ο λόγος είναι προφανής. Κατασκευάζοντας κέντρα σε εκείνες τις περιοχές, η βέλτιστη λύση κρατάει χαμηλά το συνολικό κόστος σύνδεσης. Αντιστοίχως, ο αλγόριθμος μας αναζητά τα υποδέντρα έντονης συσσώρευσης, δηλαδή εκείνα που συσσωρεύουν ένα αρκούντως μεγάλο κόστος σύνδεσης. Ειδικότερα, ενδιαφέρεται για το χαμηλότερο επιβαρυμένο υποδέντρο, έστω T_j. Έκει βρίσκεται βέβαια και το βέλτιστο κέντρο.

Το επόμενο βήμα αφορά την επιλογή του φύλλου στο υποδέντρο T_j, που θα κατασκευαστεί το νέο κέντρο. Αν ο αλγόριθμος επιλέξει τυχαία, τότε το κάτω φράγμα $O(\frac{\log n}{\log \log n})$ [19] των online αλγορίθμων παραμένει. Ο λόγος είναι ότι και ο αλγόριθμος μας μέχρι στιγμής συμπεριφέρεται ως κλασικός online. Για το λόγο αυτό, χρειάζεται επιπλέον πληροφορία από το oracle προκειμένου να ξεπεράσει το κάτω φράγμα. Μόνο η συμβολή του oracle ενδέχεται να τον οδηγήσει σε καλύτερο competitive ratio.

Το advice που δέχεται, αφορά την πλειονότητα (majority) των μελλοντικών αι-

τημάτων σε συγκεκριμένες περιοχές του χώρου. Οι περιοχές αυτές έχουν την μορφή εμφωλευμένων σφαιρών, με γεωμετρικά μειούμενες ακτίνες. Με αυτό τον τρόπο «ακολουθώντας» τις εσωτερικές σφαίρες ο αλγόριθμος προσεγγίζει όλο και μικρότερες περιοχές συσσωρεύσης που κατά συνέπεια συγκλίνουν στο βέλτιστο κέντρο. Στην περιπτωσή του Hierarchical Well-Separated Tree (HST) βέβαια, οι σφαίρες αποκτούν μία απλή και διασθητική ερμηνεία. Εν γένει, κάθε σφαίρα αντιστοιχεί σε ένα από τα δύο υποδέντρα ενός εσωτερικού κόμβου. Ειδικότερα, το κέντρο της πρώτης σφαίρας είναι ένας κόμβος-παιδί της ρίζας του T_j, έστω ο v_{j+1}. Το κέντρο της δεύτερης σφαίρας, ένας κόμβος παιδί του v_{j+1} κ.ο.κ. Έτσι, η τελευταία πρόβλεψη οδηγεί σε ένα συγκεκριμένο φύλλο, v_h, του HST. Εκεί κατασκευάζεται και το νέο κέντρο.

Επίσης, το advice ικανοποιεί δύο πολύ σημαντικές ιδιότητες που έχουμε αναφέρει επανειλλημένα. Αφενός, παρέχει στον αλγόριθμο μια ρεαλιστική πληροφορία. Μια πληροφορία δηλαδή, που ενδέχεται να προκύψει σε μια πραγματική εφαρμογή. Για παράδειγμα, αν υποθέσουμε ότι τα νέα αιτήματα ακολουθούν μια συγκεκριμένη κατανομή στον χώρο, τότε το oracle φαντάζει ως ένα μοντέλο μάθησης των κέντρων συσσώρευσης της κατανομής. Αφετέρου, η πληροφορία θεωρείται εύλογη και λογική καθώς δεν σχετίζεται άμεσα με την βέλτιστη λύση, αλλά με την φύση των ίδιων των αιτημάτων.

Συνοψίζοντας, διατυπώσαμε έναν απλό και φυσικό αλγόριθμο (LaOFL) που χρησιμοποιεί την επιπρόσθετη πληροφορία του oracle και πετυχαίνει σταθερό competitive ratio, που είναι φυσικά το βέλτιστο δυνατό. Ωστόσο, αυτό δεν συνάγει άμεσα και την βελτιστότητα του αλγορίθμου μας. Πλεόν, στο νέο μοντέλο υπολογισμού, ο αλγόριθμος δεν λειτουργεί αυτόνομα. Το μοντέλο προβλέψεων, oracle, παρέχει επιπρόσθετη πληροφορία, η οποία πρέπει επίσης να αποδειχθεί ότι είναι η ελάχιστη δυνατή για την επίτευξη σταθερού competitive ratio. Ακόμη, σε περίπτωση λανθασμένης πρόβλεψης, το σφάλμα που χρεώνεται το oracle και αφορά έμμεσα το κόστος του αλγορίθμου, πρέπει επίσης να είναι το ελάχιστο δυνατό. Εν ολίγοις, όλοι οι προβληματισμοί που αναφέραμε, αποδεικνύονται στα μετέπειτα κεφάλαια. Ακολούθως, κάνουμε μια σύντομη αναφορά στα βασικά αποτελέσματα της εργασίας.

Συμπέρασμα 6.2: *Ο αλγόριθμος 5.5 είναι* $O(1)$ *consistent.*

Θεώρημα 6.1: *Ο αλγόριθμος 5.5 έχει* $O(1)$ *competitive ratio.*

Θεώρημα 6.2: *Κάθε βέλτιστος αλγόριθμος χρειάζεται τουλάχιστον* h *bits πληροφορίας, όπου* h *το ύψος του* HST*.*

Πρόταση 6.7: *Κάθε βέλτιστος αλγόριθμος χρεώνει σφάλμα ίσο με* $d(c, w)$ *για κάθε unsatisfied demand.*

1.3 Οργάνωση

Στο Κεφάλαιο 2, παραθέτουμε το απαραίτητο θεωρητικό υπόβαθρο για την πλήρη κατανόηση των αποτελεσμάτων που παρουσιάζονται στα μετέπειτα κεφάλαια. Το Κεφάλαιο 3 επικεντρώνεται εξ ολοκλήρου στο online facility location πρόβλημα. Αρχικά, διατυπώνουμε τους δομικούς λίθους των online αλγορίθμων, και έπειτα το *κύριο* κάτω φράγμα που τους διέπει. Στην συνέχεια, κάνουμε μια σύντομη αναδρομή στις βασικότερες λύσεις με στόχο την ανάδειξη τεσάρων διαφορετικών τεχνικών αντιμετώπισης του προβλήματος. Το Κεφάλαιο 4 αφορά το νέο μοντέλο υπολογισμού: machine learned predictions. Πρώτα, κάνουμε μια σύντομη ιστορική αναδρομή που οδήγησε στη γέννηση του μοντέλου και στη συνέχεια παρουσιάζουμε τα κυριότερα αποτελέσματα. Έπειτα, συνεχίζουμε με τα Κεφάλαια 5,6 όπου παρουσιάζουμε την δουλειά αυτής της εργασίας και αποδεικνύουμε τα κυριότερα αποτελέσματα που αναφέραμε νωρίτερα. Τέλος, στο Κεφάλαιο 7 συνοψίζουμε τα αποτελέσματα μας και συζητάμε πιθανές μελλοντικές κατευθύνσεις έρευνας της learning augmented εκδόχης του online facility location προβλήματος.

Στα ακόλουθα κεφάλαια χρησιμοποιούμε συχνά την αγγλική ορολογία για την αποφυγή συγχύσεων με την ξένη βιβλιογραφία. Παρ΄όλα αυτά, στο Παράρτημα Απόδοσης ξενόγλωσσων όρων έχουμε συγκεντρώσει τους κυριότερους ξενόγλωσσους όρους για λόγους διευκόλυνσης του αναγνώστη.

Κεφάλαιο 2

Τεχνικό Υπόβαθρο

Στο Κεφάλαιο 2 αναφέρουμε το απαραίτητο θεωρητικό υπόβαθρο για την ανάλυση και των αλγορίθμων που παρουσιάζονται στα επόμενα κεφάλαια. Ξεκινάμε με μια σύντομη εισαγωγή στις βασικές έννοιες του Γραμμικού Προγραμματισμού [2.1], οι οποίες είναι απαραίτητες για την βαθύτερη κατανόηση της λειτουργίας των άμεσων (online) αλγορίθμων για το Πρόβλημα Χωροθέτησης (Facility Location). Συνεχίζουμε με μια αναφορά στους μετρικούς χώρους και στην ειδική περίπτωση του Hierarchical Well-Separated Tree. Στην Ενότητα 2.3 εισάγουμε τον αναγνώστη στους άμεσους αλγορίθμους και τελειώνουμε με μία σύντομη αναδρομή στα σημαντικότερα προβλήματα του πεδίου.

2.1 Γραμμικός Προγραμματισμός

Πολλά από τα προβλήματα που θέλουμε να λύσουμε είναι προβλήματα βελτιστοποίησης: η συντομότερη διαδρομή, η μέγιστη ροή, το φθηνότερο επικαλυπτικό δέντρο κτλπ. Σε αυτές τι περιπτώσεις αναζητάμε λύσεις οι οποίες ικανοποιούν ορισμένους περιορισμούς (constraints) και συγχρόνως ελαχιστοποιούν ή μεγιστοποιούν κάποια αντικειμενική συνάρτηση (objective function) κόστούς ή κέρδους αντιστοίχως.

Ο γραμμικός προγραμματισμός περιγράφει μια μεγάλη κατηγορία τέτοιων προβλημάτων όπου η αντικειμενική συνάρτηση, διαφορετικά *το κριτήριο βελτιστοποιήσης*, είναι μια γραμμική συναρτήση. Στην τυπική του μορφή (standard form) ένα πρόβλημα Γραμμικού Προγραμματισμού εκφράζεται ως εξής:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_{ij} x_i = b_j, \quad j = 1, \ldots, m \\ & x_i \geq 0, \qquad\quad j = 1, \ldots, n \end{aligned}$$

Πίνακας 2.1: *Τυπική Μορφή του Πρωτεύοντος Προβλήματος (P)*

Το γραμμικό πρόβλημα μπορεί επίσης να εκφραστεί σε μία πιο βολική μορφή με την χρήση πινάκων.

$$\min\ c^T x \text{ subject to } \begin{cases} Ax = b \\ x \geq 0 \end{cases} \quad \text{where } x \in \mathbb{R}^{n\times 1}, b \in \mathbb{R}^{m\times 1}, c \in \mathbb{R}^{n\times 1}, A \in \mathbb{R}^{m\times n}$$

Αν υπάρχουν τιμές της μεταβλητής $x \in \mathbb{R}^{n\times 1}$ οι οποίες να ικανοποιούν τους περιορισμούς ονομάζονται *εφικτές λύσεις* και το σύνολο που ορίζουν αναφέρεται ώς εφικτό σύνολο. Έτσι κάθε γραμμικό πρόβλημα με μη κενό εφικτό σύνολο ονομάζεται *εφικτό πρόβλημα* (feasible). Επίσης ως *βέλτιστες λύσεις* ονομάζουμε τις τιμές x^* για τις οποίες ικανοποιείται η ακόλουθη σχέση: $c^T x^* = \min\{c^T x :\ Ax = b,\ x \leq 0\}$

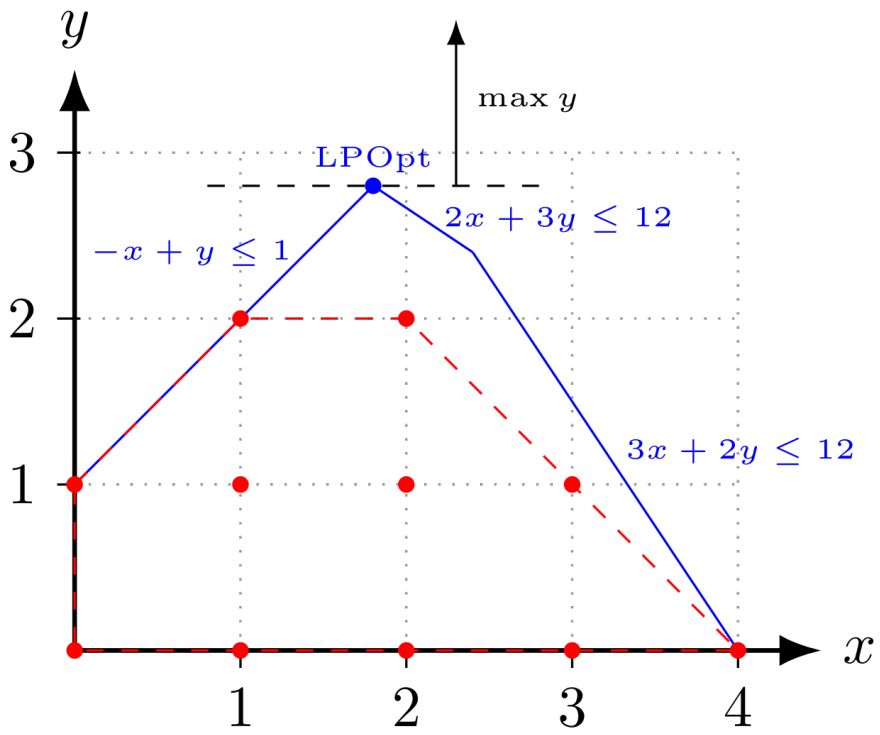
Υπάρχει ένα πλήθος τεχνικών με τις οποίες μπορούμε να τροποποιήσουμε ένα γραμμικό πρόβλημα σε κάποια ισοδύναμη μορφή της αρεσκίας μας. Για παράδειγμα αντιστρέφοντας το ένα πρόβλημα ελαχιστοποίησης σε πρόβλημα βελτιστοποίησης ή ΄σπάζοντας΄ τις ισότητες σε ανισότητες ή ακόμη αντικαθιστώντας μη-θετικές μεταβλητές σε μη-αρνητικές. Ωστόσο μια σημαντική ισοδύναμη μορφή που εμφανίζεται συχνά στην βιβλιογραφία και αξίζει να αναφέρουμε είναι η κανονική μορφή (canonical form): $\min\{c^T x\ s.t.\ Ax \geq b\}$.

Υπάρχουν αρκετοί αλγόριθμοι επίλυσης ενός Γραμμικού Προβλήματος. Ήδη από την δεκαετία του 40 είχε προταθεί ο πρώτος αλγόριθμος, ο simplex. Ωστόσο ενώ λειτουργούσε αρκετά καλά στην πράξη, ήταν γνωστό ότι στην χειρότερη περίπτωση χρειάζεται εκθετικό χρόνο για να εκτελεστεί. Έτσι μέχρι την δεκαετία του 70, όπου ορίστηκαν για πρώτη φορά οι κλάσεις πολυπλοκότητας P, NP, παρατηρήθηκε ότι ο Γραμμικός Προγραμματισμός άνηκε στην κλάση $NP \cap co - NP$. Παρόλα αυτά δεν υπήρχε μέχρι τότε κάποιος αλγόριθμος πολυωνυμικού χρόνου.

Το 1980, ο Khachiyan [25] απέδειξε ότι ο γραμμικός προγραμματισμός ανήκει τελικά στην κλάση P και πρότεινε τον πρώτο πολυωνυμικό αλγόριθμο, την ελλιψοειδή μέθοδο. Λίγα χρόνια μετά, 1984, ο Karmakar [26] εισήγαγε έναν νέο αλγόριθμο, των εσωτερικών σημείων interior-points method, ο οποίος αποδείχθηκε ότι λειτούργει καλύτερα στην πράξη.

2.1.1 Γεωμετρία Γραμμικού Προβλήματος

Το σύστημα $Ax = b$ με m εξισώσεις και n αγνώστους, περιγράφει την τομή m επιπέδων - ένα για κάθε εξίσωση. Κατ΄ αναλογία ένα σύστημα m ανισώσεων $Ax \geq b$ περιγράφει την τομή m ημιχώρων. Εαν απαιτήσουμε επιπλέον κάθε συνιστώσα του x να είναι μη αρνητική, αυτό προσθέτει άλλους n το πλήθος ημιχώρους. Φαίνεται λοιπόν λογικό ότι όσο περισσότερους περιορισμούς επιβάλλουμε, τόσο μικρότερο γίνεται το εφικτό σύνολο, βλπ Σχήμα 2.1.



Σχήμα 2.1: *Εφικτό σύνολο ενός Γραμμικού Προβλήματος (Πηγή: Wikipedia)*

2.1.2 Δυϊκό Πρόβλημα

Ίσως η πιο σημαντική έννοια που συναντάμε στον Γραμμικό Προγραμματισμό είναι η Δυϊκότητα (duality). Σε κάθε γραμμικό πρόβλημα, το οποίο καλείται **πρωτεύον** (primal), αντιστοιχεί ένα άλλο γραμμικό πρόβλημα που ονομάζεται **δυϊκό** (dual).

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} b_j y_j \\ \text{subject to} \quad & \sum_{j=1}^{m} \alpha_{ij} x_j = c_i, \quad i = 1, \ldots, n \end{aligned}$$

Πίνακας 2.2: *Τυπική Μορφή του Δυϊκού Προβλήματος (D)*

Το δυϊκό πρόβλημα λειτουργεί ως πιστοποιητικό για την βελιστότητα της λύσης του πρωτεύοντος. Πλην τούτου οδηγεί σε κομψές συνδυαστικές προτάσεις όπως η ακόλουθη:

> Σε ένα γράφημα, ο μικρότερος αριθμός ακμών του μονοπατιού μεταξύ δύο κορυφών s, t ισούται με τον μέγιστο αριθμό των $s-t$ cuts.

Η παραπάνω πρόταση είναι άμεση συνέπεια της δυϊκότητας στον ΓΠ (Γραμμικό Προγραμματισμό). Τα δύο κύρια θεωρήματα που συναντάμε αφορούν την Ασθενή (Weak) και Ισχυρή (Strong) Δυϊκότητα.

Θεώρημα 2.1 (Ασθενής Δυϊκότητα). *Έστω x, y δύο εφικτές λύσεις του πρωτεύοντος και του δυϊκού προβλήματος αντιστοίχως. Τότε ισχύει ότι $\sum_{i=1}^{n} c_i x_i \geq \sum_{j=1}^{m} b_j y_j$.*

Πρακτικά το Θεώρημα 2.1 μας λέει ότι η λύση του δυικού λειτουργεί ως κάτω φράγμα για την λύση του πρωτεύοντος. Τι συμβαίνει όταν οι αντίστοιχες λύσεις είναι και οι βέλτιστες; Την απάντηση την δίνει το Θεώρημα της Ισχυρής Δυϊκότητας 2.2.

Θεώρημα 2.2 (Ισχυρή Δυϊκότητα). *Το πρωτεύον πρόβλημα είναι εφικτό ανν το δυϊκό πρόβλημα έχει βέλτιστη λύση. Στην περίπτωση αυτή ισχύει $\sum_{i=1}^{n} c_i x_i^* = \sum_{j=1}^{m} b_j y_j^*$.*

Το Θεώρημα 2.1 αποδεικνύεται άμεσα από τις σχέσεις των αντίστοιχων γραμμικών προβλημάτων, ενώ για το Θεώρημα 2.2 απαιτείται η χρήση του Λήμματος του (Gyula) Farkas.

Για το πρόβλημα που εξετάζουμε στην παρούσα εργασία αρκεί μονάχα να γνωρίζουμε την έννοια του Δυϊκού Προβλήματος, καθώς θα αποτελέσει την βάση πάνω στην οποία χτίζονται όλοι οι άμεσοι (online) αλγόριθμοι. Για επιπλέον μελέτη στον γραμμικό προγραμματισμό προτείνουμε τον αναγνώστη να ανατρέξει στο [27].

2.2 Μετρικοί Χώροι και HST

Οι μετρικοί χώροι είναι μια ειδική κατηγορία τοπολογικών χώρων εφοδιασμένων με μία συνάρτηση που ορίζει την έννοια της απόστασης. Πρακτικά πρόκειται για ένα σύνολο σημείων $\mathcal{M}$, μαζί με μία μη-αρνητική συνάρτηση $d: \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ η οποία ονομάζεται μετρική. Η σημαντικότερη ιδιότητα που χαρακτηρίζει αυτούς τους χώρους είναι η τριγωνική ανισότητα, η οποία διατηρείται και σε πρακτικές εφαρμογές του Προβλήματος Χωροθέτησης.

Θα δώσουμε τον γενικότερο ορισμό των μετρικών χώρων και στην συνέχεια θα μιλήσουμε για μια ειδική κατηγορία που ονομάζεται Hierarchical Well-Separated Tree (HST) [24].

Ορισμός 2.1 (Μετρικός Χώρος). *Ένας μετρικός χώρος είναι ένα ζεύγος $(\mathcal{M}, d)$, όπου το $\mathcal{M}$ είναι ένα σύνολο σημείων και d μια μη-αρνητικής συνάρτησης (μετρική) $d: \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ τέτοια ώστε για κάθε $x, y, z \in \mathcal{M}$ να ισχύει:*

1. $d(x, y) = 0 \Leftrightarrow x = y$
2. $d(x, y) = d(y, x)$
3. $d(x, y) \leq d(x, z) + d(z, y)$

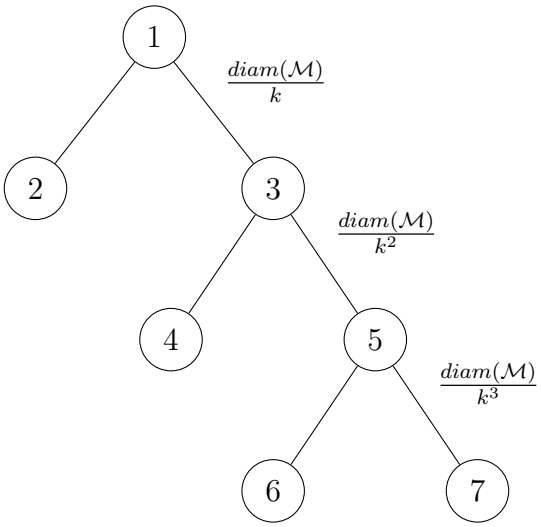
Ο επόμενος ορισμός αφορά μια ειδική κατηγορία μετρικού χώρου, το Hierarchical Well-Separated Tree (HST). Συχνά στην βιβλιογραφία το συναντάμε και ως $k-HST$, όπου το k αντιστοιχεί στον ρυθμό μείωσης (decreasing factor) της απόστασης (ή βάρους) μεταξύ των επιπέδων του δέντρου.

Ορισμός 2.2. *[24] Ως $k-HST$ ορίζεται ένα δέντρο με ρίζα και βάρη, ή μήκοι, (rooted weighted tree) με τις ακόλουθες ιδιότητες:*

(1) Το βάρος της ακμής από έναν κόμβο σε κάθε παιδί του είναι ίδιο.

(2) Τα βάρη των ακμών του μονοπάτιου από την ρίζα σε κάποιο από τα φύλλα μειώνονται κατά ένα παράγοντα τουλάχιστον k.

Η μορφή του Hierarchical Well-Separated Tree είναι φυσικά δέντρο, επομένως όλες οι ιδιότητες των δέντρων ισχύουν κάλλιστα και στο HST. Για παράδειγμα, σε ένα πλήρες (complete) δυαδικό (binary) HST, όλα τα επίπεδα του δέντρου είναι συμπληρωμένα, και κάθε εσωτερικός κόμβος έχει ακριβώς δύο κόμβους-παιδιά.



Σχήμα 2.2: *Δυαδικό k-HST*

Οι κόμβοι του δέντρου συμβολίζονται με μικρά λατινικά γράμματα ενώ συχνά χρησιμοποιούμε έναν δείκτη για να δηλώσουμε το επίπεδο στο οποίο βρίσκονται. Για πάραδειγμα ο u_i είναι ένας κόμβος του $i-$επιπέδου. Αντίστοιχα, το υποδέντρο με ρίζα τον κόμβο u_i συμβολίζεται ως T_{u_i}. Συνεχίζουμε με ένα ακόμη ορισμό σχετικά με την απόσταση των εσωτερικών κόμβων προς κάθε φύλλο.

Ορισμός 2.3. *Η απόσταση του κόμβου u_i από κάθε φύλλο του υποδέντρου T_{u_i} είναι ίδια και ίση $L_i = \sum_{j=i}^{h} \frac{\delta}{k^j} = \frac{\delta}{k^i}\frac{k}{k-1}\frac{k^{h-i+1}-1}{k^{h-i+1}}$. Επομένως μεταξύ δύο διαδοχικών επιπέδων ισχύει οι σχέσεις:* (1) $k\,L_{i+1} \leq L_i \leq L_{i-1}/k$ (2) $k\,L_{i+1} \leq L_i \leq 2\,k\,L_{i+1}$.

2.3 Άμεσοι Αλγόριθμοι

Παρόλο που το παραδοσιακό μοντέλο υπολογισμού υποθέτει ότι τα δεδομένα εισόδου είναι εκ των προτέρων γνωστά [1] , σε πολλά πραγματικά προβλήματα η υπόθεση αυτή δεν ισχύει στην πράξη. Ας φανταστούμε για παράδειγμα μια εταιρία εξυπηρέτησης πελατών με έδρα την Αθήνα, στην οποία απευθύνονται οι πελάτες όταν έχουν πρόβλημα με κάποια από τις ηλεκτρικές συσκευές τους. Οι τεχνικοί του κέντρου εξυπηρέτησης, 2 στο σύνολο, θα πρέπει να μεταβούν στην τοποθεσία του πελάτη για να επιδιορθώσουν το πρόβλημα. Η μετακίνηση όμως αυτή επιφέρει μία μορφή κόστους στην εταιρία, π.χ. τον χρόνο που χάνει ο τεχνικός για να μεταβεί στην τοποθεσία του πελάτη.

Έστω λοιπόν ότι την πρώτη μέρα λειτουργίας της εταιρίας, δύο πελάτες στην Πάτρα και στην Θεσσαλονίκη, ενημερώνουν ότι έχουν πρόβλημα με κάποια συσκευή που αγόρασαν. Η εταιρία ενημερώνει τους τεχνικούς ότι πρέπει να αναχωρήσουν άμεσα για τις δύο πόλεις και να επιστρέψουν αφότου επιδιορθώσουν το πρόβλημα. Μερικές ώρες αργότερα, και ενώ οι δύο τεχνικοί έχουν επιστρέψει, ένας νέος πελάτης από την Πάτρα ενημερώνει την εταιρία για ένα πρόβλημα που αντιμετωπίζει. Ένας από τους τεχνικούς θα πρέπει εκ νέου να αναχωρήσει για την Πάτρα, επιφέροντας επιπλέον κόστος, παρόλο που βρισκόταν ήδη εκεί πριν από λίγες ώρες. Το πρόβλημα αυτό είναι γνωστό στην βιβλιογραφία ως Το Πρόβλημα των κ-Εξυπηρετητών (k-server problem).

Ένα άλλο κλασικό πρόβλημα, με εφαρμογή στη σχεδίαση λειτουργικών συστημάτων, είναι το paging. Η κρυφή μνήμη (cache) είναι ένα μικρό τμήμα μνήμης με υψηλή ταχύτητα που έχει απευθείας σύνδεση με τον επεξεργαστή. Οι συχνά χρησιμοποιούμενες τιμές δεδομένων αποθηκεύονται στην κρυφή μνήμη, απ' όπου μπορούν να προσπελαστούν πιο γρήγορα απ' ότι οι τιμές δεδομένων που αποθηκεύονται στην μεγαλύτερη αλλά πιο αργή, κύρια μνήμη. Ωστόσο λόγο του περιορισμένου μεγέθους της, χωράει έναν προκαθορισμένο αριθμό δεδομένων, γνωστών ως σελίδες (pages). Προκειμένου μια νέα σελίδα να εισαχθεί στην κρυφή μνήμη όταν εκείνη είναι πλήρης, θα πρέπει να αντικαταστήσει μια υπάρχουσα. Το πρόβλημα έγκειται στο γεγονός ότι ο αλγόριθμος δεν γνωρίζει ποιες σελίδες θα χρησιμοποιηθούν ξανά στο μέλλον ώστε να τις κρατήσει στην κρυφή μνήμη και να αποφύγει ένα μελλοντικό σφάλμα.

Στα άμεσα (online) προβλήματα, η εισόδος I χωρίζεται σε μία ακολουθία αντικειμένων $x_1, x_2, \cdots, x_n$, όπου τα x_i συμβολίζουν τα στοιχεία του εκάστοτε προβλήματος, π.χ. στο paging αντιστοιχούν στις ζητούμενες σελίδες. Δεδομένου της εισόδου I, ένας αλγόριθμος ALG για το πρόβλημα βελτιστοποίησης $\mathcal{P}$, υπολογίζει μια εφικτή λύση $\mathrm{ALG}[I] \in F(I)$. Το αντίστοιχο κόστος συμβολίζεται με $\mathrm{ALG}(I) = C(I, \mathrm{ALG}[I])$.

Ο **βέλτιστος αλγόριθμος** (optimal algorithm) OPT είναι εκείνος όπου από όλες τις πιθανές λύσεις, βρίσκει εκείνη με το μικρότερο κόστος.

[1] Τα προβλήματα αυτά είναι γνωστά ως offline.

$$\mathrm{OPT}(I) = \min_{O \in F(I)} C(I, O)$$

Οι Sleator, Tarjan [21] πρότειναν ως μοντέλο αξιολόγησης της επίδοσης των online αλγορίθμων την ανάλυση ανταγωνιστικότητας competitive analysis. Στο μοντέλο αυτό, ο online αλγόριθμος ALG συγκρίνεται ως προς την βέλτιστη offline λύση. Ένας optimal offline αλγόριθμος γνωρίζει εκ των προτέρων όλη την ακολουθία αιτημάτων και την εξυπηρετεί με το μικρότερο δυνατό κόστος.

Ορισμός 2.4. *[21] Ένας αλγόριθμος* ALG *είναι c-ανταγωνιστικός (c-competitive)* [2] *για το πρόβλημα ελαχιστοποίησης $\mathcal{P}$ αν υπάρχει σταθερά α τέτοια ώστε για κάθε είσοδο I να ισχύει:*

$$ALG(I) \leq c \cdot OPT(I) + \alpha$$

Όταν η προσθετική σταθερά α είναι μικρότερη ή ίση με μηδέν ($\mathrm{ALG}(I) \leq c \cdot \mathrm{OPT}(I)$), θα λέμε ότι ο $\mathrm{ALG}(I)$ είναι **αυστηρά $c-$ανταγωνιστικός** (strictly c-competitive).

2.3.1 Παιγνιοθεωρητική Προσέγγιση Άμεσων Προβλημάτων

Ένας εναλλακτικός τρόπος ανάλυσης του online προβλήματος γεννάται από την παιγνιοθεωρητική (game-theoretic model) προσέγγιση του. [28]

Ας φανταστούμε ένα παιχνίδι *αιτημάτων-απαντήσεων* μεταξύ δύο παικτών. Σε αυτό το παιχνίδι ένας κακόβουλος αντίπαλος (adversary) υποβάλλει μια ακολουθία απο αιτήματα στον αλγόριθμο, τα οποία καλείται να εξυπηρετήσει ένα προς ένα. Ο αντίπαλος διαμορφώνει τα αιτήματα του με τέτοιο τρόπο, ώστε να αναγκάσει τον αλγόριθμο να οδηγηθεί σε υποβέλτιστες επιλογές. Με άλλα λόγια, να αυξήσει το competitive ratio. Για να το καταφέρει εκμεταλλεύεται την πληροφορία που διαθέτει για αυτόν, π.χ. την μέθοδο με την οποία λαμβάνει αποφάσεις.

Η περίπτωση των ντετερμινιστικών αλγόριθμων είναι τετριμμένη, καθώς αν ο αντίπαλος γνωρίζει την περιγραφή του αλγορίθμου μπορεί εύκολα να βρει μία ακολουθία για να αυξήσει το competitive ratio. Στην περίπτωση των πιθανοτικών αλγορίθμων ωστόσο, η κατάσταση είναι διαφορετική. Έχει φανεί στην πράξη, [29] [30] [28], ότι η τυχαιότητα προσφέρει στον online παίκτη πολύ μεγαλύτερη δύναμη/ευελιξία στην αντιμετώπιση ενός δύσκολου προβλήματος. Ως εκ τούτου έχουν προταθεί τρία διαφορετικά μοντέλα αντιπάλων.

[2] Εναλλακτικά λέμε ότι αλγόριθμος πετυχαίνει λόγο ανταγωνιστικότητας (competitive ratio) ίσο με $O(c)$.

- **Oblivious Αντίπαλος** *(Ασθενής Αντίπαλος)*: Ο αντίπαλος διαμορφώνει την ακολουθία των αιτημάτων εκ των προτέρων (γνωρίζοντας την περιγραφή του online αλγορίθμου) και πληρώνει το βέλτιστο κόστος.
- **Adaptive Online Αντίπαλος** *(Ισχυρός Αντίπαλος)*: Ο αντίπαλος διαμορφώνει το επόμενο αίτημα με βάση τις πρότερες αποφάσεις του αλγορίθμου, αλλά το εξυπηρετεί άμεσα (σαν online αλγόριθμος).
- **Adaptive Offline Αντίπαλος** *(Μέτριος Αντίπαλος)*: Ο αντίπαλος διαμορφώνει το επόμενο αίτημα με βάση τις πρότερες αποφάσεις του αλγορίθμου, αλλά το εξυπηρετεί βέλτιστα (σαν offline αλγόριθμος).

Η κατάταξη των μοντέλων γίνεται με βάση της *δύναμης* του αντιπάλου ως προς τον online αλγόριθμο: με τον oblivious να είναι ο πιο αδύναμος, έπειτα τον adaptive online και τέλος τον adaptive offline. Εύκολα παρατηρούμε ότι τα δύο μοντέλα: oblivious, adaptive online είναι πρακτικά ισοδύναμα στους ντετερμινιστικούς αλγορίθμους καθώς δεν εμπλέκεται κάποια τυχαιότητα στις απαντήσεις τους. Οι Ben-David et al.[28] μελέτησαν την σχετική δύναμη των τριών μοντέλων και διατύπωσαν τις ακόλουθες προτάσεις.

Θεώρημα 2.3. *Αν υπάρχει πιθανοτικός αλγόριθμος, ο οποίος είναι $c-$competitive ενάντια σε adaptive-offline αντίπαλο, τότε υπάρχει επίσης $c-$competitive ντετερμινιστικός online αλγόριθμος.*

Θεώρημα 2.4. *Αν ο πιθανοτικός αλγόριθμος $\mathcal{A}$, ο οποίος είναι $c-$competitive ενάντια σε adaptive-online αντίπαλο και υπάρχει επίσης $d-$competitive ενάντια σε oblivious αντίπαλο, τότε ο $\mathcal{A}$ είναι $(c \cdot d)-$competitive ενάντια σε adaptive-offline.*

Διασθητικά, το Θεώρημα 2.3 υπονοεί ότι η τυχαιότητα (randomization) δεν βοηθάει ενάντια σε adaptive-offline αντιπάλους.

2.3.2 Αρχή του Yao

Ο Yao [31] μελέτησε τον αναμενόμενο χρόνο εκτέλεσης των αλγορίθμων από δύο διαφορετικές οπτικές. Στην πρώτη, την οποία ονόμασε distributional, υποθέτουμε την ύπαρξη μιας κατανομής δεδομένων εισόδου (inputs) και αναζητάμε τον γρηγορότερο αλγόριθμο υπό την συγκεκριμένη κατανομή. Όπως για παράδειγμα σε ένα πρόβλημα ταξινόμησης όπου ψάχνουμε τον ταχύτερο αλγόριθμο, παράλληλα έχουμε υποθέσει ότι κάθε στιγμιότυπο από τις $n!$ δυνατές διατάξεις εμφανίζεται ισοπίθανα με τα υπόλοιπα. Στην δεύτερη, την οποία ονόμασε randomized, δεν κάνουμε κάμια υπόθεση για τα δεδομένα εισόδου αλλά επιτρέπουμε στον αλγόριθμο να κάνει τυχαίες επιλογές, να λειτουργεί δηλαδή στοχαστικά. Ωστόσο η ιδέα του Yao για την μορφή των πιθανοτικών αλγορίθμων είναι κατά τι διαφορετική από αυτό που γνωρίζουμε στις μέρες μας.

Κατά τον Yao ένας πιθανοτικός αλγόριθμος ορίζεται από μια κατανομή σε ένα σύνολο «αγνών» (pure) αλγορίθμων, δηλαδή ντετερμινιστικών. Έτσι, το αναμενόμενο κόστος ενός πιθανοτικού αλγορίθμου ισοδυναμεί με το σταθμισμένο μέσο όρο των ντετερμινιστικών αλγορίθμων με τα βάρη να αντιστοιχούν στην πιθανότητα επιλογής τους. Τυπικά, στην μία περίπτωση μιλάμε για κατανομή ως προς τα δεδομένα εισόδου και στην άλλη για κατανομή ως προς τους «αγνούς» αλγόριθμους.

Οι δύο αυτές οπτικές οδήγησαν τον Yao σε δύο διαφορετικούς ορισμούς πολυπλοκότητας για το πρόβλημα βελτιστοποίησης $\mathcal{P}$: distributional complexity, randomized complexity. Στους ορισμούς που ακολουθούν συμβολίζουμε με $\mathcal{A}$ το σύνολο των pure αλγορίθμων και με $\mathcal{I}$ το σύνολο των στιγμιοτύπων εισόδου. Τα αναμενόμενα κόστη για τους ντετερμινιστικούς και πιθανοτικούς αλγορίθμους είναι τα ακόλουθα:

- Το $C(A, \mathcal{D}) = \sum_{I \in \mathcal{I}} d(I)\, r(A, I)$ ορίζει το μέσο κόστος ενός συγκεκριμένου ντετερμινιστικού αλγορίθμου $A \in \mathcal{A}$ υπό μια κατανομή $\mathcal{D}$ στα δεδομένα εισόδου $\mathcal{I}$.
- Το $\mathbb{E}[c(R, I)]] = \sum_{A \in \mathcal{A}} q(A)\, r(A, I)$ ορίζει το αναμενόμενο κόστος του πιθανοτικού αλγορίθμου R σε ένα συγκεκριμένο στιγμιότυπο $I \in \mathcal{I}$. Οι πιθανότητες ορίζονται από την κατανομή $\mathcal{Q}$ στο σύνολο των ντετερμινιστικών αλγορίθμων $\mathcal{A}$.

Ορισμός 2.5. *Distributional Complexity:* $F_1(\mathcal{P}) = \sup_{\mathcal{D}} \min_{A \in \mathcal{A}} \mathrm{C}(A, \mathcal{D})$

Ορισμός 2.6. *Randomized Complexity:* $F_2(\mathcal{P}) = \inf_{R} \max_{I \in \mathcal{I}} \mathbb{E}[c(R, I)]]$

Θεώρημα 2.5. *Η Αρχή του Yao:* $F_1(\mathcal{P}) = F_2(\mathcal{P})$

Το Θεώρημα 2.5 αποδεικνύεται ένα πολύ ισχυρό εργαλείο για τον υπολογισμό κάτω-φραγμάτων στους πιθανοτικούς αλγορίθμους.

$$\max_{I \in \mathcal{I}} \mathbb{E}[cost(R, I)]] \geq \inf_{R} \max_{I \in \mathcal{I}} \mathbb{E}[cost(R, I)]] = F_2 = F_1 = \sup_{\mathcal{D}} \min_{A \in \mathcal{A}} C(A, \mathcal{D}) \qquad (2.1)$$

Η Σχέση 2.1 είναι ισοδύναμη με την ακόλουθη πρόταση: *η χειρότερη επίδοση του καλύτερου πιθανοτικού αλγορίθμου είναι ίση με τον μέσο όρο της επίδοσης του καλύτερου ντετερμινιστικού αλγορίθμου στην χειρότερη κατανομή των δεδομένων εισόδου.*

Αρκεί λοιπόν να κατασκευάσουμε μία **δύσκολη** κατανομή στιγμιοτύπων εισόδου, και να υπολογίσουμε την επίδοση των ντετερμινιστικών αλγορίθμων σε αυτήν. Όπως θα δούμε και στην συνέχεια, η συγκεκριμένη τεχνική χρησιμοποιείται στο [1] για την απόδειξη του κάτω φράγματος των άμεσων αλγορίθμων για το Πρόβλημα Χωροθέτησης Εγκαταστάσεων.

2.4 Σύντομη Αναδρομή σε Άμεσα Προβλήματα

Ήδη από την δεκαετία του 90, όπου το πεδίο των online αλγορίθμων άρχισε να αποκτά έντονο ερευνητικό ενδιαφέρον, τέσσερα βασικά πρόβλημα μελετήθηκαν εκτενώς: της κρυφής μνήμης (paging), της ενημέρωσης λίστας (list update), των κ-εξυπηρετητές (k-servers), των συστήματων μετρικών εργασιών (metrical task systems).

2.4.1 Πρόβλημα της Κρυφής Μνήμης

Υποθέτουμε ένα σύστημα 2 επιπέδων μνήμης, μιας γρήγορης (cache) και μίας αργής. Η γρήγορη μνήμη έχει χωρητικότητα k σελίδων δεδομένων, ενώ η αργή μπορεί να αποθηκεύσει έναν αυθαίρετα μεγάλο αριθμό σελίδων. Τώρα, κάθε νέο αίτημα που φτάνει, ζητάει από την γρήγορη μνήμη μια συγκεκριμένη σελίδα. Αν η σελίδα αυτή υπάρχει μέσα στην γρήγορη μνήμη τότε το αίτημα εξυπηρετείται άμεσα χωρίς κάποιο επιπλέον κόστος. Αν όμως η σελίδα δεν υπάρχει, τότε προκαλείται σφάλμα μνήμης έτσι ώστε η σελίδα να μεταφερθεί από την αργή μνήμη στην γρήγορη. Σε αυτό το σημείο ο αλγόριθμος αποφασίζει ποια σελίδα θα αφαιρέσει από την γρήγορη μνήμη, ώστε να την αντικαταστήσει με την ζητούμενη. Η απόφαση αυτή λαμβάνεται δίχως γνώση για τις σελίδες που θα ζητηθούν ξανά στα μέλλον. Έτσι, υπάρχει περίπτωση να αφαιρεθεί μια σελίδα που θα ζητηθεί ξανά στο μέλλον και να προκληθεί εκ νέου σφάλμα μνήμης. Στόχος λοιπόν του αλγορίθμου ειναι να βρει την κατάλληλη πολιτική αντικατάστασης σελίδων ώστε να μεγιστοποιήσει τον συνολικό αριθμό σφαλμάτων μνήμης.

Οι Sleator, Tarjan [21] μελέτησαν το πρόβλημα από την σκοπιά των ντετερμινιστικών αλγορίθμων και διατύπωσαν δύο πολιτικές αντικατάστασης: Least Recently Used (LRU), First-In-First-Out (FIFO). Στην πρώτη αφαιρείται η σελίδα που χρησιμοποιήθηκε πιο παλιά στο παρελθόν, και στην δεύτερη η σελίδα που καταλαμβάνει χώρο στην κρυφή μνήμη για περισσότερη ώρα. Τα θεωρήματα που διατύπωσαν είναι τα ακόλουθα:

Θεώρημα 2.6. *Οι πολιτικές αντικατάστασης LRU,FIFO είναι $k-$competitive.*

Θεώρημα 2.7. *Κανένας ντετερμινιστικός αλγόριθμος δεν μπορεί να πετύχει competitive ratio μικρότερο από k.*

Οι Fiat et al. [32] διατύπωσαν αρκετά χρόνια αργότερα τον πρώτο πιθανοτικό αλγόριθμο με competitive ratio ίσο με $2H_k$, όπου H_k είναι ο k-οστός αρμονικός αριθμός. Ο αλγόριθμος λειτουργεί μαρκάροντας τις ζητούμενες σελίδες που βρίσκονται ήδη στην μνήμη και αφαιρεί τις μη-μαρκαρισμένες στην περίπτωση σφάλματος. Όταν πλέον μαρκαριστούν όλες οι σελίδες και συμβεί σφάλμα, οι σελίδες ξεμαρκάρονται και η διαδικασία ξεκινάει από την αρχή.

2.4.2 Πρόβλημα Ενημέρωσης Λίστας

Το πρόβλημα ενημέρωσης λίστας (list updating problem) είναι ένα από τα πρώτα online προβλήματα που μελετήθηκαν υπό το μοντέλο του competitive ratio. Στο πρόβλημα, ο αλγόριθμος διατηρεί μία μη ταξινομημένη λίστα στοιχείων η οποία τροποποιείται ανάλογα με το αίτημα που προκύπτει. Πιο συγκεκριμένα, κάθε αίτημα ανήκει σε μία από τις ακόλουθες κατηγορίες ενεργειών: πρόσβαση (access), προσθήκη (insertion), διαγραφή (deletion). Ανάλογα με το αίτημα, ο αλγόριθμος χρεώνεται με κόστος λόγω του χρόνου που απαιτείται για την εξυπηρέτηση του αιτήματος. Σε περίπτωση πρόσβασης και διαγραφής, ο αλγόριθμος διατρέχει την λίστα από την αρχή μέχρι να βρει το ζητούμενο στοιχείο και το διαβάζει/διαγράφει. Το κόστος τότε ισούται, με την θέση του στοιχείου στην λίστα. Σε περίπτωση προσθήκης, ο αλγόριθμος διατρέχει όλη την λίστα μέχρι το τέλος προκειμένου να βεβαιωθεί ότι το συγκεκριμένο στοιχείο δεν υπάρχει ήδη και έπειτα το προσθέτει στο τέλος, με συνολικό κόστος $(n+1)$.

Καθώς εξυπηρετεί ένα αίτημα, ο αλγόριθμος έχει την δυνατότητα να τροποποιήσει την λίστα. Μετά από ένα αίτημα πρόσβασης/προσθήκης, επιτρέπεται στον αλγόριθμο να μετακινήσει το στοιχείο σε οποιαδήποτε θέση μικρότερη από την αρχική, χωρίς επιπλέον κόστος. Έτσι, ένα μελλοντικό αίτημα προς το ίδιο στοιχείο θα κοστίσει λιγότερο για να εξυπηρετηθεί. Οι τρεις ντετερμινιστικές πολιτικές τροποποίησης που μελετήθηκαν πρώτα είναι: Move-To-Front, Transpose, Frequency-Count. Πάλι οι Sleator, Tarjan [21] διατύπωσαν τα ακόλουθα θεωρήματα αναφορικά με τις πολιτικές που αναφέραμε.

Θεώρημα 2.8. *[21] Ο αλγόριθμος Move-To-Front είναι* $2-$*competitive.*

Πρόταση 2.1. *[21] Οι αλγόριθμοι Transpose, Frequency-Count δεν είναι* $c-$*competitive για καμία σταθερά* c*.*

Αναφορικά με τους πιθανοτικούς αλγορίθμους αποδείχθηκε ότι δεν υπάρχει αλγόριθμος καλύτερος από $2-$competitive για adaptive αντιπάλους, βλπ [33], [28]. Αντιθέτως στην περίπτωση των oblivious αντιπάλων, διατυπώθηκαν διάφοροι αλγόριθμοι, όπου δύο εξ αυτών αποδείχθηκαν βέλτιστοι: πιθανοτικός Bit, ντετερμινιστικός Timestamp

Θεώρημα 2.9. *[33] Ο αλγόριθμος Bit είναι* $1.75-$*competitive ενάντια σε oblivious αντίπαλο.*

Θεώρημα 2.10. *[34] Ο αλγόριθμος Bit είναι* $1.75-$*competitive ενάντια σε oblivious αντίπαλο.*

Έπειτα, οι Bernhard von Stengel et al. [35] συνδυάζοντας τους δύο αλγορίθμους, διατύπωσαν έναν νέο, Combination, με καλύτερο competitive ratio.

Θεώρημα 2.11. *[35] Ο αλγόριθμος Combination είναι* $1.6-$*competitive ενάντια σε oblivious αντίπαλο.*

2.4.3 Πρόβλημα των $k-$ Εξυπηρετητών

Στο πρόβλημα του k-server μας δίνεται ένα μετρικός χώρος $\mathcal{S}$ και k εξυπηρετήτες σε κάποιες αρχικές θέσης του $\mathcal{S}$. Κάθε νέο αίτημα αντιστοιχεί και σε ένα σημείο $x \in \mathcal{S}$ στον μετρικό χώρο. Ο αλγόριθμος προκειμένου να εξυπηρετήσει το αίτημα θα πρέπει να μετακινήσει έναν από τους εξυπηρετητές στην θέση του αιτήματος με κόστος όσο η απόσταση των δύο σημειών. Ο στόχος είναι λοιπόν, να εξυπηρετήσει τα αιτήματα με την λιγότερη δυνατή μετακίνηση των εξυπηρετητών. Από τον ορισμό του πρόβληματος άμεσα διαπιστώνουμε ότι πρόκειται για γενίκευση του paging.

Το πρόβλημα των k-server διατυπώθηκε από τους Manasse et al. [36] οι οποίοι απέδειξαν το κάτω φράγμα για την περίπτωση των ντετερμινιστικών αλγορίθμων.

Θεώρημα 2.12. *[36] Κάθε ντετερμινιστικός αλγόριθμος είναι τουλάχιστον $k-$competitive για το πρόβλημα των k-server .*

Έπειτα, οι Bartal et al. απέδειξαν ένα νέο φράγμα για τους πιθανοτικούς αλγορίθμους.

Θεώρημα 2.13. *[37] Το competitive ratio κάθε πιθανοτικού αλγόριθμου σε ένα αυθαίρετο μετρικό χώρο είναι $\Omega(\frac{\log k}{log^2 \log k})$ ενάντια σε oblivious αντιπάλους.*

2.4.4 Πρόβλημα των Μετρικών Συστημάτων

Το πρόβλημα των metrical task systems, διατυπώθηκε από τους Borodin et al [29] και αποτελεί ένα μοντέλο για έναν μεγάλο αριθμό online προβλημάτων. Για την ακρίβεια, όλα τα προβλήματα που αναφέραμε προηγούμενως προκύπτουν ως ειδικές περιπτώσεις των metrical task systems.

Το πρόβλημα περιγράφεται από ένα ζεύγος (S, d), όπου το S είναι ένα σύνολο n καταστάσεων και d μία συνάρτηση κόστους που ικανοποιεί την τριγωνική ανισότητα, δηλαδή μια μετρική συνάρτηση 2.1. Συγκεκριμένα, η τιμή $d(i, j)$ αντιστοιχεί στο κόστος αλλαγής κατάστασης από την i στην j. Επίσης δίνεται και ένα σύνολο αιτημάτων/εργασίων $\mathcal{T}$ που πρέπει να πραγματοποιηθούν. Το συνολικό κόστος του αλγορίθμου υπολογίζεται από το άθροισμα των μεταβάσεων μεταξύ των καταστάσεων που έκανε ο αλγόριθμος, καθώς επίσης και του κόστους ολοκλήρωσης κάθε εργασίας $T \in \mathcal{T}$. Στόχος και πάλι είναι να ελαχιστοποιηθεί το συγκεκριμένο κόστος. Τα θεωρήματα που διατύπωσαν είναι να ακόλουθα.

Θεώρημα 2.14. *[29] Υπάρχει ντετερμινιστικός αλγόριθμος με $(2\,n-1)-$competitive ratio για κάθε metrical task system με $n-$καταστάσεις.*

Θεώρημα 2.15. *[29] Κάθε ντετερμινιστικός αλγόριθμος έχει competitive ratio τουλάχιστον $(2\,n-1)$, όπου n ο αριθμός των καταστάσεων.*

Αναφορικά με τους πιθανοτικούς αλγορίθμους οι Fiat et al. [38] διατύπωσαν τα ακόλουθα θεωρήματα.

Θεώρημα 2.16. *[38] Υπάρχει πιθανοτικός αλγόριθμος με $O(\frac{\log^2 n}{\log^2 \log n})-$competitive ratio για κάθε metrical task system με $n-$καταστάσεις, ενάντια σε oblivious αντιπάλους.*

Θεώρημα 2.17. *[38] Κάθε πιθανοτικός αλγόριθμος έχει competitive ratio τουλάχιστον $\Omega(\frac{\log^2 n}{\log^2 \log n})$ ενάντια σε oblivious αντίπαλο, όπου n ο αριθμός των καταστάσεων.*

Προτείνουμε στον αναγνώστη που επιθυμεί να μελετήσει εκτενέστερα τα συγκεκριμένα προβλήματα να ανατρέξει στα [39], [40].

Κεφάλαιο 3

Άμεσοι Αλγόριθμοι για το Πρόβλημα Χωροθέτησης Εγκαταστάσεων

Το Κεφάλαιο 3 επικεντρώνεται στο άμεσο (online) πρόβλημα Χωροθέτησης Εγκαταστάσεων. Αρχικά, παραθέτουμε τις δύο αρχές που διέπουν τους άμεσους αλγορίθμους και ακολούθως την απόδειξη του κάτω φράγματος που τους χαρακτηρίζει. Έπειτα κάνουμε μια σύντομη αναδρομή σε ορισμένους κλασσικούς αλγορίθμους με στόχο την ανάδειξη τεσσάρων διαφορετικών τεχνικών αντιμετώπισης του προβλήματος. Για λόγους απλότητας, οι αποδείξεις στηρίζονται στην ειδική περίπτωση όπου τα κόστη κατασκευής νέων εγκατάστασεων είναι κοινά (uniform facility cost) για κάθε σημείο του μετρικού χώρου.

3.1 Δύο Αρχές

Στον πυρήνα κάθε αλγορίθμου για το άμεσο Πρόβλημα Χωροθέτησης Εγκαταστάσεων βρίσκεται ο κανόνας κατασκευής νέων κέντρων/εγκαταστάσεων. Αυτό μονάχα καθορίζει την συμπεριφορά τους και κατ΄ επέκταση την επίδοση τους. Η μόνη απόφαση που καλείται να πάρει ένας αλγόριθμος είναι το **πότε** και του **που** θα κατασκευάσει ένα νέο κέντρο.

Όπως θα δούμε στις επόμενες ενότητες και οι τέσσερεις αλγόριθμοι υλοποιούν *φαινομενικά* διαφορετικούς κανόνες κατασκευής νέων κέντρων. Αν κοιτάξουμε βαθύτερα θα παρατηρήσουμε ότι όλοι τους υπακούν στις δύο ακόλουθες αρχές. Η όποια διαφορετικότητα τους έγκειται μονό στον τρόπο με τον οποίο τις εφαρμόζουν.

Αρχή 1. *[18] Κάθε αλγόριθμος διατηρεί μια ισορροπία μεταξύ του συνολικού κόστους σύνδεσης και του συνολικού κόστους των εγκαταστάσεων.*

Με άλλα λόγια, κάθε αλγόριθμος μοιράζει το κόστος κατασκευής ενός νέου κέντρου σε επιμέρους αιτήματα που την προκάλεσαν. Μπορούμε εύκολα να καταλάβουμε της σημαντικότητα της Αρχής 1, διότι διαφορετικά θα έπρεπε να «εντοπίζουμε» τις χρονικές στιγμές που κατασκευάζονται νέα κέντρα, πράγμα φυσικά αδύνατον. Η «ισορροπία»

για την οποία κάνει λόγο η Αρχή 1 υλοποιείται χρεώνοντας ένα επιπλέον κόστος, που ονομάζεται **potential**, σε κάθε νέο αίτημα για την μελλοντική συνεισφορά του σε ένα νέο κέντρο. Η τιμή του potential ισούται με κόστος σύνδεσης του αιτήματος την στιγμή της άφιξης του.

Αρχή 2. *[18] Κάθε αλγόριθμος απαγορεύει στο συσσωρευμένο κόστος σύνδεσης μιας περιοχής να ξεπεράσει το κόστος κατασκευής ενός νέου κέντρου.*

Εκ πρώτης όψεως η Αρχή 2 δείχνει κάπως αόριστη ως προς την έννοια «των περιοχων». Πως ορίζονται; Που βρίσκονται; Στην πραγματικότητα, η κάλυψη του κενού επαφίεται στον εκάστοτε αλγόριθμο και αποτελεί την ειδοποιό διαφορά μεταξύ των υπολοίπων. Σε κάθε περίπτωση η Αρχή 2 φαίνεται λογική ιδιότητα να ικανοποιείται από τον αλγόριθμο. Ωστόσο δεν πρόκειται για δική μας επινόηση, αλλά πηγάζει από τους περιορισμούς που προκύπτουν στο δυϊκό Γραμμικό Πρόβλημα. Για περισσότερες πληροφορίας προτρέπουμε τον αναγνώστη να ανατρέξει στην Ενότητα 3.5 και στο [18].

3.2 Κάτω Φράγμα

Η απόδειξη του κάτω φράγματος δόθηκε για πρώτη φορά από τον Fotakis [19] [1]. Αφορά τους πιθανοτικούς αλγόριθμους ενάντια σε oblivious adversaries με ομοιόμορφα κόστη κατασκευής νεών κέντρων σε έναν απλό μετρικό χώρο, την ευθεία (πιο συγκεκριμένα, ένα ευθύγραμμο τμήμα).

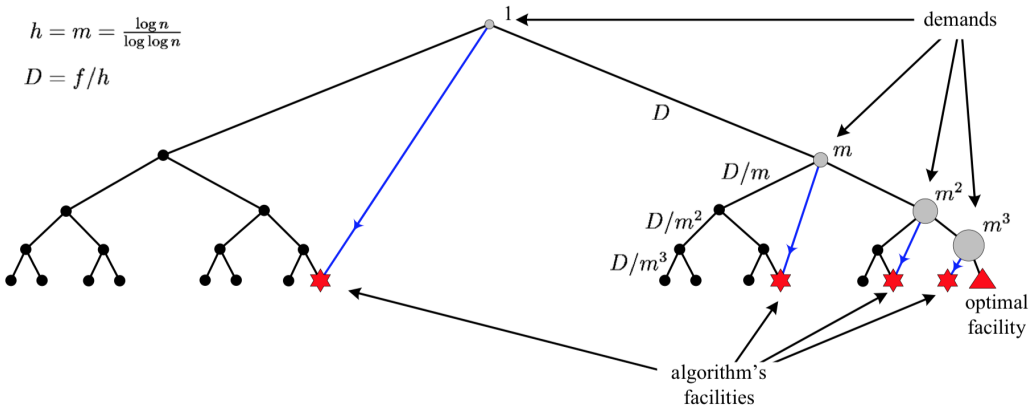
Ωστόσο η απόδειξη του θεωρήματος δεν γίνεται απευθείας πάνω στην ευθεία. Αρχικά, αποδεικνύεται για την περίπτωση ενός πολύ συγκεκριμένου Hierarchical Well-Separated Tree (Ενότητα 2.2) ως προς κάθε ντετερμινιστικό αλγόριθμο. Έπειτα μέσω της Αρχής του Yao 3.1 γενικεύεται στους πιθανοτικούς αλγόριθμους. Το τελικό στάδιο αφορά την ενσωμάτωση (embedding) του HST σε ένα κομμάτι της πραγματικής ευθείας. Εμείς επικεντρωνόμαστε μόνο στο αρχικό μέρος της απόδειξης, καθώς εκεί κρύβεται η ουσία του επιχειρήματος.

Η κατασκευή της απόδειξης, όπως αναφέραμε, χρησιμοποιεί ένα Hierarchical Well-Separated Tree με προσεκτικά επιλεγμένες παραμέτρους D, h, m. Συγκεκριμένα, κατασκευάζεται ένα πλήρες δυαδικό HST ύψους h τέτοιο ώστε να ισχύουν τα ακόλουθα:

- Η απόσταση από την ρίζα στους κόμβους-παιδιά είναι ίση με D.
- Σε κάθε μονοπάτι από την ρίζα ως κάποιο φύλλο, η απόσταση μεταξύ των επιπέδων μειώνεται κατά έναν παράγοντα m.

Ως αποτέλεσμα, κάθε κόμβος u του επιπέδου j έχει απόσταση το πολύ $\frac{D}{m^{j-1}}\frac{1}{m-1}$ από τους κόμβους που ανήκουν στο υποδέντρο T_u και τουλάχιστον $\frac{D}{m^{j-1}}$ από τους κόμβους που βρίσκονται εκτός του υποδέντρου. Σχετικά με την ακολουθία των αιτημάτων, εκείνη διαμορφώνει ένα μονοπάτι που ξεκινά από την ρίζα και καταλήγει στο φύλλο

που βρίσκεται το βέλτιστο κέντρο, όπως φαίνεται στο Σχήμα 3.1. Συνεπώς, ορίζεται μια ακολουθία $h+1$ φάσεων με την πρώτη να ξεκινά με την άφιξη του αιτήματος στην ρίζα, την δεύτερη με την άφιξη m αιτημάτων στον κόμβο του επιπέδου 1 κ.ο.κ.



Σχήμα 3.1: *HST: Κάτω φράγμα άμεσων αλγορίθμων.*

Θεώρημα 3.1 (Κάτω Φράγμα Άμεσων Αλγορίθμων [19],[1])**.** *Κάθε πιθανοτικός αλγόριθμος για το πρόβλημα της Άμεσης Χωροθέτησης Εγκαταστάσεων πετυχαίνει λόγο ανταγωνιστικότητας τουλάχιστον* $\Omega(\frac{\log n}{\log \log n})$ *ενάντια σε oblivious adversary ακόμα και στην περίπτωση της πραγματικής ευθείας.*

Για την πλήρη απόδειξη του Θεωρήματος 3.1 μπορεί ο αναγνώστης να ανατρέξει στα ακόλουθα: [19] [1], [18]. Εμείς θα αναφέρουμε μονάχα κάποια σημεία που κρίνουμε απαραίτητα για την καλύτερη κατανόηση της πολυπλοκότητας του προβλήματος.

Παρατηρούμε λοιπόν από το Σχήμα 3.1 ότι τα αιτήματα «φτάνουν» με σειρά από έξω προς τα μέσα. Επιπλέον, κάθε ντετερμινιστικός αλγόριθμος στο πέρας κάθε φάσης είναι σε θέση να γνωρίζει το υποδέντρο στο οποίο βρίσκεται το βέλτιστο κέντρο. Για παράδειγμα, ας υποθέσουμε ότι η j είναι η φάση που μόλις ολοκληρώθηκε. Τότε το υποδέντρο που ανήκει το βέλτιστο κέντρο είναι σίγουρα το T_{u_j}. Διαφορετικά, αν το βέλτιστο κέντρο δεν ανήκει εκεί τότε τα αιτήματα της j-οστής φάσης θα πλήρωναν κόστος σύνδεσης τουλάχιστον $\geq \frac{D}{m^{j-1}} m^j = D\,m = f$, πράγμα που αντιβαίνει στην βελτιστότητα της λύσης.

Ο αλγόριθμος λοιπόν **προσεγγίζει** την θέση του βέλτιστου κέντρου με τον ίδιο ρυθμό που η ακολουθία των αιτημάτων συγκλίνει προς αυτό. Συνεπώς, αποκαλύπτοντας τα αιτήματα από εξώ προς τα μέσα αναγκάζουμε τον αλγόριθμο να καθυστερεί όσο το δυνατόν περισσότερο να εντοπίσει το βέλτιστο κέντρο, με αποτέλεσμα να αυξάνεται ο λόγος ανταγωνιστικότητας. Αυτό συμβαίνει διότι στο τέλος κάθε φάσης, έστω j η φάση, ο αλγόριθμος καλείται να κατασκευάσει ένα νεό κέντρο στο υποδέντρο T_{u_j}. Επειδή δεν διαθέτει κάποια πληροφόρια πλην του υποδέντρου, επιλέγει ένα από τα φύλλα (σχεδόν) στην τύχη. Ωστόσο με πιθανότητα $1/2$ τα νέα αιτήματα της φάσης

$j+1$ θα βρεθούν στο αντίθετο υποδέντρο από εκείνο που επέλεξε ο αλγόριθμος, χρεώνοντας κόστος σύνδεσης τουλάχιστον f και υποχρεώνοντας πάλι τον αλγόριθμο να κατασκεύασει ένα ακόμη κέντρο (λόγω της Αρχής 2). Η διαδικασία αυτή επαναλαμβάνεται μέχρι να τελείωσουν τα αιτήματα οδηγώντας σε $h+1$ περισσότερα κέντρα. Θέτοντας $h = \frac{\log n}{\log \log n}$ καταλήγουμε στο ζητούμενο.

Σημείωση: Όλοι οι όροι με $*$ superscript αντιστοιχούν σε τιμές της βέλτιστης λύσης.

3.3 Πιθανοτικός Αλγόριθμος

Ο Meyerson [17] διατύπωσε το πρόβλημα του Online Facility Location (OFL) και παρουσίασε έναν ιδιαίτερα κομψό και διαισθητικό αλγόριθμο, που πετυχαίνει $O(\log n)$ competitive ratio ενάντια σε adversarial inputs. Ώστοσο, λίγο αργότερα με πιο προσεκτική ανάλυση, αποδείχθηκε από τον Fotakis [19] ότι ο αλγόριθμος πετυχαίνει πράγματι το βέλτιστο competitive ratio.

Κάθε νέο αίτημα που καταφθάνει, πρέπει να συνδεθεί αμετάκλητα με ένα προϋπάρχον κέντρο είτε να κατασκευάσει ένα νέο. Στην δεύτερη περίπτωση, το νέο κέντρο θα βρίσκεται στην θέση του ίδιο του αιτήματος με αποτέλεσμα το κόστος σύνδεσης να είναι μηδενικό. Συμβολίζουμε με F_i το σύνολο των ανοικτών εγκαταστάσεων facilities την χρονική στιγμή i που φτάνει το αίτημα. Το $d(F_i, u_i) = \min_{f \in F_i} d(f, u_i)$ αντιστοιχεί στην απόσταση του αιτήματος από το κοντινότερο κέντρο, και συμβολίζεται με δ_i.

Αλγοριθμος 3.1: *RandOFL: Πιθανοτικός Αλγόριθμος*

while New demand u_i at position x_i arrives **do**
  Open a new facility at x_i with probability $min(1, \frac{\delta_i)}{f})$;
  Otherwise
  Assign u_i to the nearest facility;
end while

Η ιδέα του αλγορίθμου είναι ιδιαίτερα απλή. Πρακτικά, όταν το νέο αίτημα βρεθεί κοντά σε ένα ανοικτό κέντρο, ο αλγόριθμος θα επιλέξει να το συνδέσει με το συγκεκριμένο κέντρο. Αντιθέτως, αν ένα μεγάλο μέρος των αιτημάτων βρεθεί σε μία περιοχή του χώρου, όπου δεν υπάρχει ήδη κάποιο κέντρο για να τα εξυπηρετήσει με μικρό κόστος, ο αλγόριθμος κατασκευάζει ένα νέο προκειμένου να διατηρήσει το *μελλοντικό* κόστος σύνδεσης χαμηλά. Το ερώτημα που προκύπτει είναι: πόσο μεγάλο κόστος σύνδεσης θα χρεωθεί ο αλγόριθμος προτού κατασκευάσει το νέο κέντρο; Την απάντηση στο ερώτημα την δίνουμε στην Πρόταση 3.2.

Αν το αίτημα u_i προκαλέσει την δημιουργία νέου κέντρου ο αλγόριθμος χρεώνεται κόστος f για την κατασκευή ενώ το κόστος σύνδεσης είναι μηδενικό αφού το νέο κέντρο βρίσκεται στην θέση του αιτήματος u_i. Στην αντίθετη περίπτωση ο αλγόριθμος

χρεώνεται μόνο το κόστος σύνδεσης $d(F_i, u_i)$. Συνεπώς το αναμμενόμενο κόστος φράσσεται από:

$$\mathbb{E}[\text{COST}(u_i)] = \frac{\delta_i}{f}\delta_i + (1 - \frac{\delta_i}{f})\delta_i \overset{(1-\frac{\delta_i}{f})\in[0,1]}{\leq} \delta_i + 1 \cdot \delta_i = 2 \cdot \delta_i$$

Είναι λοιπόν προφανές ότι ο Αλγόριθμος 3.1 υπακούει στην Αρχή 1 δεσμεύοντας μία (αναμενόμενη) ποσότητα ίση με δ_i για την μελλοντική κατασκευή ενός κέντρου. Ωστόσο απαιτείται λίγη περισσότερη προσπάθεια για την απόδειξη της Αρχής 2. Η απόδειξη που ακολουθεί οφείλεται στον Lang [41].

Πρόταση 3.2 (Lang [41]). *Το συνολικό (αναμενόμενο) κόστος σύνδεσης μέχρι την κατασκευή του επόμενου κέντρου είναι μικρότερο από f.*

Απόδειξη. Για λόγους απλότητας θεωρούμε ότι $f = 1$. Επίσης με E_i συμβολίζουμε το αναμενόμενο κόστος σύνδεσης πριν την κατασκευή του νέου κέντρου όταν ο αλγόριθμος τρέχει στα αιτήματα $(u_i, u_{i+1}, \cdots, u_n)$. Ορίζουμε ως $P_i^j = \prod_{l=i}^{j}(1 - \delta_l)$ την πιθανότητα μη κατασκευής νέου κέντρου μέχρι την χρονική στιγμή j. Συνεπώς το E_i ισούται με $\sum_{j=i}^{n} \delta_j \cdot P_i^j$.

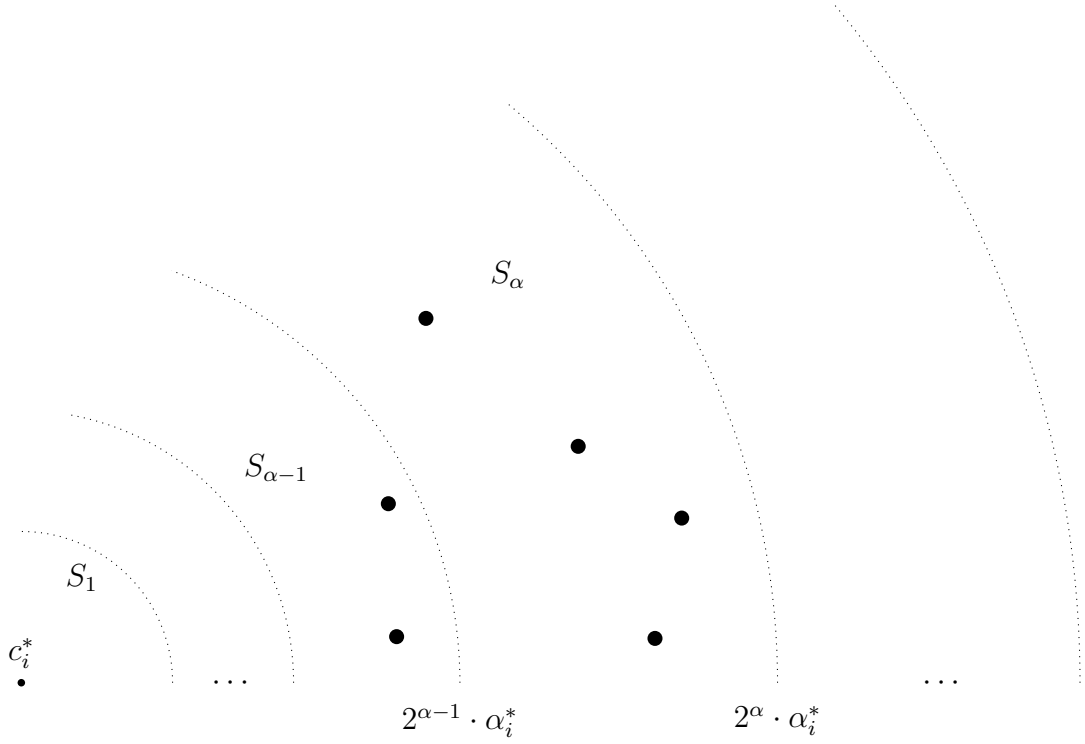
Το E_i επάγει επίσης την ακόλουθη αναδρομική σχέση: $E_i = (1 - \delta_i)(\delta_i + E_{i+1})$. Παρατηρούμε ότι $E_n \leq 1/4$ με την μέγιστη τιμή να εμφανίζεται ότι $\delta_n = 1/2$. Υποθέτοντας επαγωγικά ότι $E_{i+1} < 1$, προκύπτει ότι $E_i = (1 - \delta_i)(\delta_i + E_i) < 1 - \delta_i^2 \leq 1$. [1] Επομένως καταλήγουμε στο ζητούμενο $E_1 < 1$. □

Θεώρημα 3.2 (Meyerson [17]). *Ο αλγόριθμος 3.1 είναι $O(\log n)$ competitive ενάντια σε adversarial input.*

Πρόχειρη Απόδειξη. Θεωρούμε μία βέλτιστη συστάδα (cluster) C_i^*. Έστω S_α το σύνολο των σημειών της C_i^* που βρίσκονται σε απόσταση $(2^{\alpha-1}\alpha_i^*, 2^\alpha \alpha_i^*]$ από το βέλτιστο κέντρο c_i, όπου $\alpha_i^* = A_i^*/|C_i^*|$, με $A_i^* = \sum_{p \in C_i^*} \delta_p^*$, η μέση απόσταση των σημείων που συνδέονται στον κέντρο c_i^* στην βέλτιστη λύση. Τα σύνολα αυτά έχουν την μορφή ομόκεντρων κύκλων γύρω από το κέντρο με γεωμετρικά αυξανόμενες ακτίνες, βλπ Σχήμα 3.2 Παρατηρούμε ότι το σύνολο $S_{\log n+1}$ είναι υποχρεωτικά κένο, διότι διαφορετικά το κόστος σύνδεσης ενός και μόνο σημείου θα ξεπερνούσε το κόστος ολόκληρης της συστάδας.

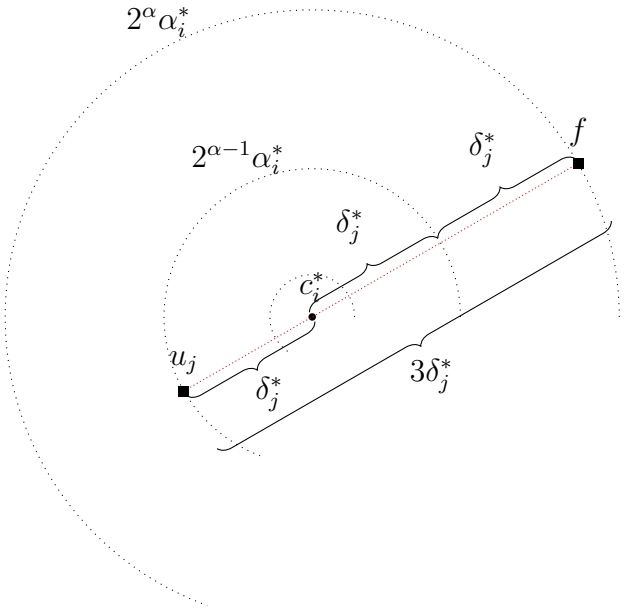
Εξετάζουμε τα σημεία του συνόλου S_α. Το αναμενόμενο κόστος που χρεώνεται ο αλγόριθμος μέχρι την κατασκευή ενός νέου κέντρου είναι το πολύ ίσο με f, σύμφωνα με την Πρόταση 3.2. Έπειτα κάθε άλλο σημείο u_j χρεώνεται απόσταση το πολύ δ_j^*. Άρα το συνολικό αναμενόμενο κόστος για το u_j φράσσεται από $3\delta_j^* + \frac{3\delta_j^*}{f}\delta_j^* = 6\delta_j^*$, βλπ

[1] Εφαρμόζουμε οπισθοδρομική-αναδρομή (Backwards Induction)



Σχήμα 3.2: *Συστάδα βέλτιστου κέντρου.*

Σχήμα 3.3 . Επόμενως για κάθε σύνολο στοιχείων ο αλγόριθμος χρεώνεται κόστος σύνδεσης έξι φορές μεγαλύτερο από το βέλτιστο και το κόστος κατασκευής f ενός νέου κέντρου. Αφού υπάρχουν το πολύ $\log n$ τέτοια σύνολα το συνολικό κόστος φράσσεται από: $6A^* + (1 + \log n)f$. □



Σχήμα 3.3: *Μέγιστη απόσταση αιτήματος από το κοντινότερο κέντρο.*

Παρατηρούμε λοιπόν ότι το δύσκολο σενάριο (worst case instance) συμβαίνει όταν

τα αίτημα φτάνουν με σειρά από μακρία προς κοντά. Το ίδιο φαινόμενο παρατηρήθηκε και στην περίπτωση του κάτω φράγματος (Ενότητα 3.2). Δεν πρόκειται φυσικά για σύμπτωση, αλλά προκύπτει από την συνδυαστική δομή του προβλήματος. Διαισθητικά, η μόνη πληροφορία που λαμβάνει ο αλγόριθμος για την θέση του βέλτιστου κέντρου, προέρχεται από την θέση των αιτημάτων. Επομένως, ένας αντίπαλος (adversary) που επιθυμεί να δυσκολέψει τον αλγόριθμο, αρκεί να αναδιατάξει τα αιτήματα με φθίνουσα σειρά ως προς την απόσταση τους από το βέλτιστο κέντρο. Αντιθέτως όσο νωρίτερα φτάσουν τα αιτήματα στη γύρω περιοχή από το βέλτιστο κέντρο, τόσο πιο άμεσα θα κατασκευαστεί ένα νέο facility και βελτιώνοντας συνεπώς το competitive ratio. Η παρατήρηση αυτή οδήγησε το Meyerson στην διατύπωση της παρακάτω πρότασης.

Θεώρημα 3.3. *[17] Ο αλγόριθμος 3.1 έχει σταθερό competitive ratio όταν τα αιτήματα έρχονται με τυχαία σειρά.*

Πρόχειρη Απόδειξη. Θεωρούμε μία βέλτιστη συστάδα (cluster) C_i^* με κέντρο το σημείο c_i^*. Διαχωρίζουμε το σύνολο των αιτημάτων που συνδέονται με το συγκεκριμένο κέντρο, σε δύο κατηγορίες ανάλογα με την απόσταση τους: καλά (good) & κακά (bad). Συμβολίζουμε με γ_p το κόστος που χρεώνεται ο αλγόριθμος για ένα καλό σημείο p. Υποθέτουμε ότι ακόμα και στην περίπτωση κατασκευής ενός νέου κέντρο, ο αλγόριθμος χρεώνεται το κόστος σύνδεσης, δηλαδή $\mathbb{E}[\gamma_p] = \mathbb{E}[\delta_p]$. Με $\alpha_i^* = A_i^*/|C_i^*|$, όπου $A_i^* = \sum_{p \in C_i^*} \delta_p^*$, συμβολίζουμε την μέση απόσταση. Τα καλά (good) σημεία γ είναι εκείνα που βρίσκονται εσωτερικά της σφαίρας $B(c_i^*, \alpha_i^*)$ και κακά γ_b (bad) τα υπόλοιπα.

Η απόδειξη συνεχίζεται φράσσοντας το αναμενόμενο κόστος για τις δύο κατηγορίες σημείων. Συγκεκριμένα, αποδεικνύεται ότι $\mathbb{E}[\sum_g \gamma_g] \leq 2f + 2A_i^* + 2\sum_g \delta_g^*$, $\mathbb{E}[\gamma_b] \leq 2\delta_b^* + \frac{2}{|C_i^*|}\big(f + \sum_g (\mathbb{E}[\gamma_g] + 2\delta_g^*)\big)$ για τα καλά (γ_g) και κακά (γ_b) σημεία αντίστοιχα. Συνδυάζοντας τις δύο σχέσεις καταλήγουμε στο ζητούμενο. □

Στην αρχή της ενότητας αναφέραμε ότι ο Αλγόριθμος 3.1 με πιο προσεκτική ανάλυση πετυχαίνει τελικά το βέλτιστο competitive ratio $O(\frac{\log n}{\log \log n})$. Μένει λοιπόν να αποδείξουμε το τελευταίο θεώρημα της Ενότητας 3.3.

Θεώρημα 3.4. *[19] Ο αλγόριθμος 3.1 έχει $O(\frac{\log n}{\log \log n})$ competitive ratio.*

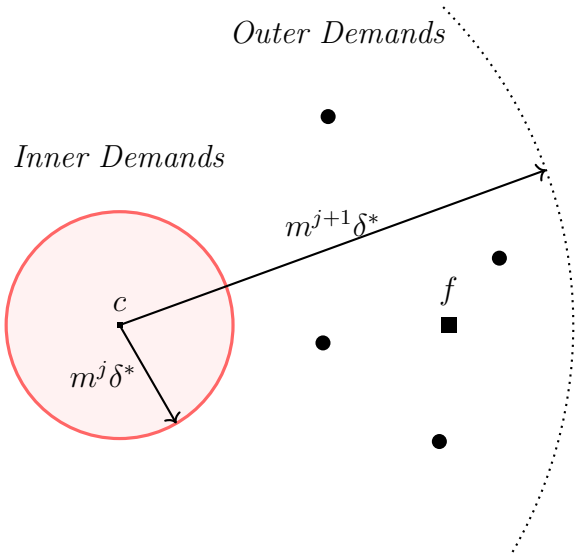
Απόδειξη. Έστω m, h θετικές σταθερές όπου $m^h > n$. Για λόγους ευκολίας υποθέτουμε την ύπαρξη ενός μόνο κέντρου, c, στην βέλτιστη λύση. Η ανάλυση σπάει σε $h+1$ φάσεις, όπου η κάθεμια αντιστοιχεί στην ελάχιστη απόσταση ανάμεσα στα κέντρα που κατασκευάζει ο αλγόριθμος και το βέλτιστο. Συγκεκριμένα, η φάση j διαρκεί όσο το κοντινότερο κέντρο βρίσκεται σε απόσταση $[m^j\delta^*, m^{j+1}\delta^*)$ και σταματά όταν η απόσταση γίνει μικρότερη από $m^j\,\delta^*$.

Τα αιτήματα κάθε φάσης χωρίζονται σε δύο κατηγορίες: εσωτερικά (inners), εξωτερικά (outers) ανάλογα με την απόσταση τους από το βέλτιστο κέντρο, c, βλπ

Σχήμα 5.2. Διαισθητικά, τα εξωτερικά αιτήματα κάθε φάσης είναι εκείνα που απέχουν την μέγιστη δυνατή απόσταση από το βέλτιστο κέντρο. Επομένως μέσω της τριγωνικής ανισότητας, το αναμενόμενο κόστος σύνδεσης φράσσεται εύκολα ως εξής:

$$d(F_{i-1}, u_i) \leq d(F_{i-1}, c) + d^*_{u_i} \leq (m+1)\, d^*_{u_i}$$

Διότι από τον ορισμό της φάσης j, ισχύει $d(F_{i-1}, c) \leq m^{j+1}\delta^* \leq (m+1)\, d^*_{u_i}$. Συνεπώς, συνυπολογίζοντας το κόστος για την πιθανή κατασκευή νέων κέντρων (potential, βλπ Αρχή 1), ο αλγόριθμος χρεωνέται συνολικά κόστος $\leq 2(m+1)\, d^*_{u_i}$ για την εξυπηρέτηση του u_i αιτήματος. Τα εσωτερικά αιτήματα από την άλλη, βρίσκονται πολύ κοντά στο βέλτιστο κέντρο και επομένως δεν υπάρχει κάποιος άμεσος τρόπος να φραχτεί το αναμενόμενο κόστος που επιφέρουν. Ωστόσο, από την Πρόταση 3.2 γνωρίζουμε ότι δεν ξεπερνάει (κατά μέση τιμή) το f. Άρα με κόστος το πολύ $2f$ ο αλγόριθμος έχει εξυπηρετήσει μερικά από τα εσωτερικά αιτήματα και έχει ταυτόχρονα κατασκευάσει ένα κέντρο στην γύρω περιοχή, οδηγώντας στον τερματισμό της τρέχουσας φάσης. Τελικά, από τον συνολικό αριθμό των φάσεων καταλήγουμε ότι τα εσωτερικά αιτήματα χρεώνουν κόστος το πολύ $2(h+1)\, f$. Συνδυάζοντας τις δύο σχέσεις και θέτοντας $h = m = \frac{\log n}{\log \log n}$ καταλήγουμε στο ζητούμενο. □



Σχήμα 3.4: *Εσωτερικά και Εξωτερικά αιτήματα*

3.4 Ντετερμινιστικός Αλγόριθμος

Ο Fotakis [1], [19] έπειτα διατύπωσε τον πρώτο ντετερμινιστικό αλγόριθμο για το πρόβλημα. Παρά το γεγονός ότι ο αλγόριθμος πετυχαίνει το βέλτιστο competitive ratio $O(\frac{\log n}{\log \log n})$ έχει ιδιαίτερα δύσκολη ανάλυση και δεν χρησιμοποιείται συχνά σε πρακτικές εφαρμογές λόγω της υψηλής χρονική πολυπλοκότητας του.

Η ιδέα του φυσικά είναι παρόμοια με αυτήν που παρουσιάσαμε στον αλγόριθμο 3.1. Ωστόσο ο αλγόριθμος 3.1 υλοποιεί την δεύτερη αρχή 2 πιθανοτικά (με τεχνική αναμενόμενου χρόνου (βλπ Πρόταση 3.2)) ενώ ο αλγόριθμος 3.2 ντετερμινιστικά. Πρακτικά, χρησιμοποιεί έναν τοπικό κανόνα κατασκευής νέων κέντρων.

Αλγοριθμος 3.2: *DetOFL: Ντετερμινιστικός Αλγόριθμος*

$F \leftarrow \emptyset,\ L \leftarrow \emptyset;$
for each new demand w **do**
  $L \leftarrow L \cup w,\ r_w \leftarrow\ d(F,w)/x;$
  $B_w \leftarrow Ball(w, f_w) \cap L;\ Pot(B_w) \leftarrow \sum_{u \in B_w} d(F,u);$
  if $Pot(B_w) \geq f$ **then**
    if $d(F,w) < f$ **then**
      Let ν be the smallest integer:
        either there exists one point $u \in B_w$ such that
          $Pot(B_w \cap Ball(u, r_w/2^\nu)) \leq Pot(B_w)/2,$
        or for every $u \in B_w : Pot(B_w \cap Ball(u, r_w/2^{\nu+1})) \leq Pot(B_w)/2.$
      Let $\hat{w}$ be any point in $B_w : Pot(B_w \cap Ball(\hat{w}, r_w/2^\nu)) > Pot(B_w)/2$
    else
      $\hat{w} \leftarrow w;$
      $F \leftarrow F \cup \hat{w}; L \leftarrow L \backslash B_w;$
    end if
  end if
end for

Με L συμβολίζουμε το σύνολο των ανικανοποίητων αιτήματα (unsatisfied demands), δηλαδή όσων διατηρούν ακόμα το potential (βλ, Αρχή 1) για την ενδεχομένη συμμετοχή τους στην κατασκευή ενός νέου κέντρου στην γύρω περιοχή. Διαισθητικά, για κάθε νέο αίτημα w, ο αλγόριθμος υπολογίζει το συσσωρευμένο potential και αποφασίζει αν θα κατασκεύασει ένα νέο κέντρο ή όχι. Αρχικά, ορίζει την μεγαλύτερη δυνατή σφαίρα και ελέγχει αν $Pot(B_w) \geq f$ (1). Αν η συνθήκη ικανοποιείται, τότε μειώνοντας γεωμετρικά την ακτίνα εντοπίζει την σφαίρα με την μικρότερη δυνατή ακτίνα για την οποία η συνθήκη (1) παραμένει αληθής. Το συγκεκριμένο βήμα αποδεικνύεται καίριο για την επίτευξη του βέλτιστου competitive ratio. Μετά την κατασκευή του νέου κέντρου όσα αιτήματα συνέβαλαν στο κόστος χάνουν το potential τους και γίνονται satisfied.

Θεώρημα 3.5. *Για $x \geq 10$ το competitive ratio του αλγορίθμου 3.2 είναι $O(\frac{\log n}{\log \log n})$.*

Για την ανάλυση της απλής περίπτωσης, όπου η βέλτιστη λύση αποτελείται από ένα μόνο κέντρο, ο Fotakis [1] ακολουθεί μια παρόμοια διαδικασία με εκείνη στην απόδειξη Θεωρήματος 3.4. Αρχικά, διαιρεί τον χώρο σε φάσεις ανάλογα με την απόσταση ως προς το βέλτιστο κέντρο και εξετάζει διαφορετικά την περίπτωση των εσωτερικών και εξωτερικών αιτημάτων. Παρόλα αυτά, απαιτείται η χρήση ενός μη-τετριμμένου επιχειρήματος συνάρτησης δυναμικού (potential function argument).

Στην άλλη περίπτωση, όπου η βέλτιστη λύση έχει πολλαπλά κέντρα, η ανάλυση είναι **ιδιαίτερα** πολύπλοκη. Για πρώτη φορά εισαγάγεται η έννοια των απομονωμένων συνασπισμών κέντρων (isolated coalitions). Η ιδέα είναι ότι κάθε isolated coalition μπορεί να αναλυθεί ανεξάρτητα από τα υπόλοιπα, σαν να πρόκειται για μεμονωμένο κέντρο. Η διαφορά έγκειται στο ότι πλέον η κατασκευή νέων κέντρων από τον αλγόριθμο επηρεάζει το potential ολόκληρης της ομαδάς και όχι ενός μόνο κέντρου. Διαφορετικά, αν ακολουθείτο παρόμοια ανάλυση με την περίπτωση του ενός κέντρου, θα καταλήγαμε σε κόστος $\Omega(k\,f)$, όπου k ο αριθμός των βέλτιστων κέντρων.

Το μόνο που μένει είναι ο αλγόριθμος να *περιμένει* ώστε τα coalitions να απομονωθούν, κάτι που συμβαίνει έπειτα από λίγο, σύμφωνα με το [1]. Αποδεικνύεται επίσης ότι ο αλγόριθμος 3.2 μπορεί να γενικευθεί για μη ομοιόμορφα κόστη κατασκευής νέων κέντρων, πετυχαίνοντας επίσης το βέλτιστο competitive ratio.

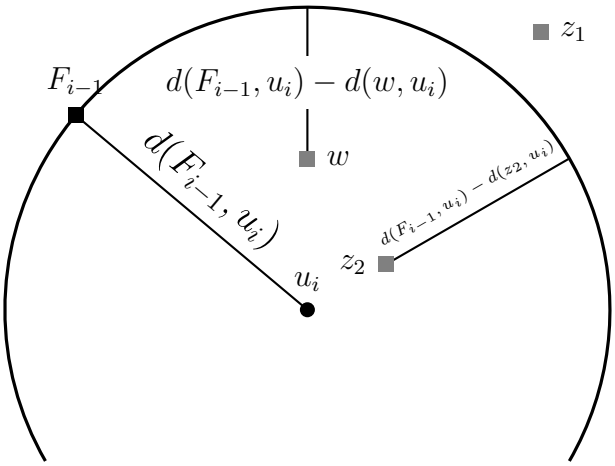
3.5 Αλγόριθμος Πρωτεύοντος-Δυϊκού Προβλήματος

Στο [42] ο Fotakis, διατύπωσε έναν νέο ντετερμινιστικό αλγόριθμο για το πρόβλημα του OFL. Παρομοίως με όσα αναφέραμε στις προηγούμενες ενότητες, ο αλγόριθμος διατηρεί ένα σύνολο F από κέντρα που κατασκευάζει κατά την διάρκεια άφιξης νέων αιτημάτων. Επίσης στηριζόμενος στην Αρχή 1 δεσμεύει για κάθε νέο αίτημα μία ποσότητα κόστους ίση με το κόστος σύνδεσης κατά την άφιξη του αιτήματος (potential). Η διαφορά με τον αλγόριθμο 3.2 είναι ότι πλέον το potential των αιτημάτων δεν μηδενίζεται πλήρως όταν ένα νέο κέντρο κατασκευάζεται αλλά μείωνεται σταδιακά. Συγκεκριμένα, το potential σε ένα σημείο $z \in \mathcal{M}$ του (μετρικού) χώρου, εώς την χρονική στιγμή i, ορίζεται ως:

$$p_i(z) = \sum_{l=1}^{i} (d(F_{i-1}, u_l) - d(z, u_l))_+$$

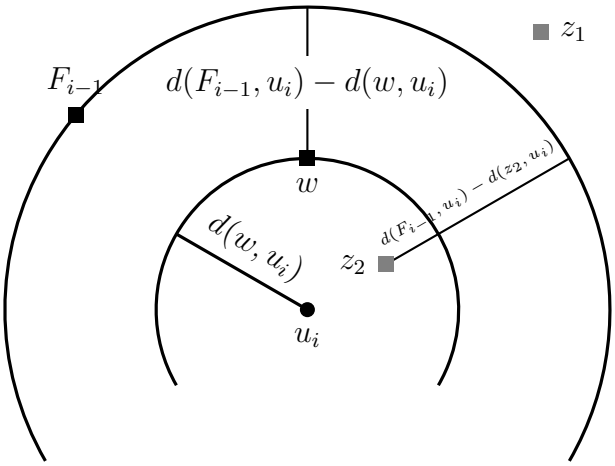
Ως παράδειγμα, ας φανταστούμε το ακόλουθο παιχνίδι. Κάθε αίτημα αντιστοιχεί σε ένα παίκτη u_i ο οποίος μετά την άφιξη του, διαθέτει τόσα χρήματα, όσα πλήρωσε για να συνδεθεί με το κοντινότερο κέντρο, $d(F_{i-1}, u_i)$. Με τα χρήματα αυτά επιθυμεί να συνδράμει στην κατασκευή νέων κέντρων στο μέλλον. Ώστοσο δεν ξοδεύει τα χρήματα του άσκοπα αλλά με τρόπο που οφελεί τον ίδιο μελλοντικά. Επομένως ενδιαφέρεται μόνο για τα κέντρα που κατασκευάζονται στην γύρω περιοχή, με άλλα λόγια πιο κοντά από εκείνο που τον εξυπηρέτησε κατά την άφιξη του. Ωστόσο ακόμα και για τα κοντινά ως προς εκείνον κέντρα, ο παίκτης u_i παραμένει επιφυλακτικός.

Τα χρήματα που είναι διατεθιμένος να πληρώσει για το νέο κέντρο w είναι ακριβώς όσα θα κέρδιζε, αν το w ήταν ήδη κατασκευασμένο κάτα την άφιξη, βλπ Σχήμα 3.5. Αν τελικά το νέο κέντρο κατασκευαστεί τότε ο παίκτης χρεώνεται το συγκεκριμένο



Σχήμα 3.5: *Potential πριν την κατασκευή νέου κέντρου.*

ποσό και όσα χρήματα περισσέψουν, τα ξοδεύει αντιστοίχως για μελλοντικά κέντρα, βλπ Σχήμα 3.6.



Σχήμα 3.6: *Potential μετά την κατασκευή νέου κέντρου.*

Παρόλα αυτά, η επιλογή του νέου κέντρου δεν γίνεται τυχαία. Με την άφιξη νέων αιτημάτων, το συσσωρευμένο potential σε κάποιο σημείο του χώρου ενδέχεται να ξεπεράσει το f με αποτέλεσμα να παραβιαστεί η Αρχή 2. Τότε, προκειμένου να επέλθει **ισορροπία** (δηλαδή να διατηρηθεί το feasibility της dual λύσης), ο αλγόριθμος κατασκευάζει ένα νέο κέντρο στην θέση όπου η Αρχή 2 παραβιάστηκε κατά το μέγιστο.

Η ιδέα, όπως προδίδει και το όνομα της Ενότητας 3.5, προέρχεται από το δυϊκό πρόβλημα, βλπ Υποενότητα 2.1.2. Η μέθοδος του πρωτεύοντος-δυϊκού προβλήματος είναι μια ιδιαίτερα χρήσιμη, και πολλές φορές ταχύτερη, μέθοδος για την επίλυση γραμμικών προβλημάτων. Συνήθως όταν θέλουμε να λύσουμε ένα γραμμικό πρόβλημα, το τροφοδοτούμε σε έναν αλγόριθμο, π.χ. τον simplex, και εκείνος το λύνει. Πολλές φορές όμως, αυτό δεν είναι αναγκαίο. Αντί να λύσουμε *ολόκληρο* το πρωτεύον γραμμι-

κό πρόβλημα, μπορούμε να απλά να κατασκεύασουμε μια αρχική (εφικτή) λύση για το δυϊκό πρόβλημα. Έπειτα, μεταβάλλοντας σιγά σιγά την λύση, προσέχοντας πάντα να παραμένει εφικτή (feasible), ελαχιστοποιούμε (μεγιστοποιούμε) όλο και περισσότερο το κόστος (κέρδος) μας. Εν γένει, αυτή είναι η βασική ιδέα της μεθόδου πρωτεύοντος-δυϊκού. Για περισσότερες πληροφορίες, παραμπέμπουμε τον αναγνώστη στο survey [43], όπου η συγκεκριμένη τεχνική εφαρμόζεται για μια πληθώρα online προβλημάτων.

Αναφορικά με τον αλγόριθμο στο [42], ξεκινάμε γράφοντας το ακέραιο πρόβλημα, βλπ Πίνακα 3.1.

$$
\begin{array}{lccl}
\min & \sum_{z\in\mathcal{M}} f_z y_z & + & \sum_{z\in\mathcal{M}}\sum_{i=1}^{n} x_{zi} d(z,u_i) \\
\text{subject to} & \sum_{z\in\mathcal{M}} x_{zi} = 1 & & \forall \text{ demand } u_i \\
& x_{zi} \le y_z & & \forall z \in \mathcal{M}, \forall \text{ demand } u_i \\
& y_z \in \{0,1\}, x_{zi} \in \{0,1\} & & \forall z \in \mathcal{M}, \forall \text{ demand } u_i
\end{array}
$$

Πίνακας 3.1: *Πρωτεύον Ακέραιο Πρόβλημα Χωροθέτησης*

Θέτοντας τις μετάβλητες y_z, x_{zi} ίσες με 1, συμβολίζουμε το άνοιγμα ενός κέντρου στο σημείο z και την σύνδεση του i αιτήματος με το z αντιστοίχως. Αφαιρώντας τον περιορισμό ακεραιότητας των μεταβλητών y_z, x_{zi} οδηγούμαστε στο γραμμικό πρόβλημα [2]. Το δυϊκό του πρόβλημα παρουσιάζεται στον Πίνακα 3.2.

$$
\begin{array}{ll}
\max & \sum_{i=1}^{n} \alpha_i \\
\text{subject to} & \alpha_i \le \beta_{zi} + d(z,u_i) \forall z \in \mathcal{M}, \forall \text{ demand } u_i \\
& \sum_{i=1}^{n} \beta_{zi} \le f_z \quad \forall z \in \mathcal{M} \\
& \alpha_i \ge 0, \beta_{zi} \ge 0 \quad \forall z \in \mathcal{M}, \forall \text{ demand } u_i
\end{array}
$$

Πίνακας 3.2: *Δυϊκό Γραμμικό Πρόβλημα Χωροθέτησης*

Χρησιμοποιώντας τον πρώτο περιορισμό του δυϊκού και λύνοντας ως προς β_{zi} καταλήγουμε στην σχέση: $\beta_{zi} \ge (\alpha_i - d(z,u_i))_+$, την οποία έπειτα αντικαθιστούμε στον δεύτερο περιορισμό. Το απλοποιημένο δυϊκό πρόβλημα φαίνεται στον Πίνακα 3.3.

Ο περιορισμός του δυϊκού, μας δίνει εν τέλει την Αρχή 2. Προκειμένου η λύση του δυϊκού να είναι εφικτή (feasible) θα πρέπει σε κάθε σημείο του χώρου να ισχύει $\sum(\alpha_i - d(z,u_i)) \le f_z$. Ερμηνεύουμε την μεταβλητή α_i ως την απόσταση του αιτήματος i ως προς το κοντινότερο κέντρο κατά την άφιξη του, δηλαδή ως προς το potential. Καθώς όλο και περισσότερα αίτημα καταφθάνουν, το αριστερό μέρος του περιορισμού αυξάνεται, μέχρι που τελικά ξεπερνά την τιμή f_z. Αυτό σηματοδοτεί την ανάγκη για

[2] Linear programming relaxation

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{n} (\alpha_i - d(z, u_i))_+ \leq f_z \forall z \in \mathcal{M} \\ & \alpha_i \geq 0 \qquad \forall z \in \mathcal{M}, \forall \text{ demand } u_i \end{aligned}$$

Πίνακας 3.3: *2ο Δυϊκό Γραμμικό Πρόβλημα Χωροθέτησης*

την δημιουργία ενός νέου κέντρου. Έτσι επανέρχεται η ισορροπία και η λύση γίνεται και πάλι εφικτή. Για να συμβεί πρέπει ο αλγόριθμος να επιλέξει ως νέο κέντρο το σημείο όπου ο περιορισμός παραβιαζόταν κατά το μέγιστο. Έτσι, επαναφέροντας την εφικτότητα σε εκείνο το σημείο, επαναφέρεται αυτόματα και στα υπολοιπά. Μια πιο διαισθητική ερμηνεία είναι η εξής: αν ο αλγόριθμος αγνοήσει τον περιορισμό και δεν κατασκευάσει νέο κέντρο, τα μελλοντικά αιτήματα θα συσσωρεύσουν ακόμα περισσότερο κόστος, με αποτέλεσμα το competitive ratio να μεγαλώσει περαιτέρω. Το βασικό θεωρήμα του [42] είναι το ακόλουθο.

Αλγοριθμος 3.3: *SNFL: Αλγόριθμος Πρωτεύοντος-Δυϊκού*

$F \leftarrow \emptyset,\ L \leftarrow \emptyset$;
$initializePotentials()$;
for each new demand u **do**
  $L \leftarrow L \cup u$;
  $updatePotentials(F, u)$;
  $w \leftarrow arg\max_{z \in \mathcal{M}} \{p(z) - f_z\}$;
  if $p(w) - f_w > 0$ **then**
    $F \leftarrow F \cup \{w\}$;
    $computeNewPotentials(F, L)$;
  end if
  Assign u to the nearest facility in F;
end for

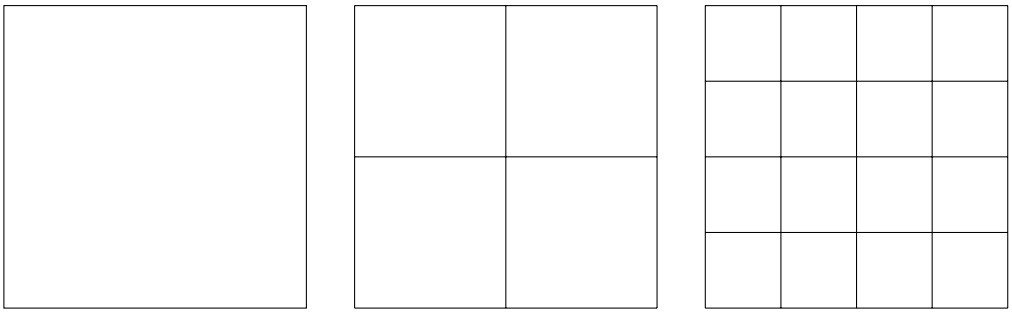
Θεώρημα 3.6. *[42] Το competitive ratio του αλγορίθμου 3.3 δεν είναι μεγαλύτερο από* $(4\ \log(n+1)+2)$.

Μια εναλλακτική ανάλυση από τους Williamson et al., βασισμένη στην μέθοδο του dual fitting, υπάρχει στο [44].

3.6 Ντετερμινιστικός Αλγόριθμος στον Ευκλείδειο Χώρο

Στο [20] οι Anagnostopoulos et al. παρουσιάσαν έναν απλό και υπολογιστικά αποδοτικό, ντετερμινιστικό αλγόριθμο για το OFL πρόβλημα. Ο αλγόριθμος έχει

competitive ratio $\Theta(\log n)$ στο επίπεδο και $\Theta(2^d \log n)$ σε ευκλείδιους χώρους d διαστάσεων.



Σχήμα 3.7: *Διάσπαση του χώρου σε τεταρτημόρια*

Ο αλγόριθμος στηρίζεται σε μια ιεραρχική αποδόμηση του χώρου σε τεταρτημόρια. Ξεκινώντας από το μηδενικό επίπεδο (level-0), ολόκληρος ο χώρος αντιστοιχεί σε ένα τετράγωνο με πλευρά f — η υπόθεση αυτή έπειτα γενικεύεται. Το τετράγωνο αυτό διασπάται σε τέσσερα νέα τετράγωνα (quadrants) με πλευρές $f/2$ το καθένα, βλπ Σχήμα 3.7. Τα νέα τετράγωνα αποτελούν τα τεταρτημόρια του επιπέδου-1 (level-1). Εν γένει, κάθε τεταρτημόριο του επιπέδου j με πλευρά $2^{-j}f$, διασπάται/αποδομείται σε τέσσερα νέα τεταρτημόρια (quadrants) του επιπέδου-$(j+1)$ με πλευρά $2^{j+1}f$ το κάθενα. Τα επίπεδα αυτά διαμορφώνουν ένα πλήρες δέντρο, με τους κόμβους να αντιστοιχούν στα τεταρτημόρια και τα παιδιά τους στα τέσσερα επιμέρους τεταρτημόρια που διασπάται το αρχικό.

Τα τεταρτημόρια ταξινομούνται σε τρείς κατηγορίες: *ανοικτά*, *ενεργά*, *ανενεργά* open, active, inactive). Στην πρώτη περίπτωση — ανοικτά — υπάρχει ανοικτό κέντρο στο εσωτερικό τους, συγκεκριμένα στο κέντρο του τεταρτημορίου. Στην δεύτερη περίπτωση — ενεργά — πρέπει ο κόμβος-γονιός τους, στο δέντρο, να είναι ανοικτός, δηλαδή να έχει ανοικτό κέντρο στο εσωτερικό. Στην τρίτη περίπτωση — ανενεργά — τα τεταρτημόρια περιμένουν απλά να γίνουν *ανοικτά*. Τυπικά, το τεταρτημόριο του επιπέδου-0 παραμένει ανοικτό από την στιγμή που ανοίξει το πρώτο κέντρο. Για κάθε $j \geq 1$, αν ο κόμβος-γονιός ενός τεταρτημορίου επιπέδου-j είναι ανοικτός, τότε το τεταρτημόριο θεωρείται ενεργό αν υπάρχουν λιγότερα από 2^{j+2} αιτήματα συνδεδεμένα με αυτό, και ανοικτό διαφορετικά.

Αλγοριθμος 3.4: *SimpleDFL: Αλγόριθμος στον Ευκλείδειο Χώρο*

$q \leftarrow FindQuadrant()$;
$support(q) \leftarrow support(q) \cup \{c\}$;
$cost(q) \leftarrow cost(q) + d(c, F)$;
if $cost(q) > \alpha \cdot f$ **then**
    $Partition(q)$;
end if

Ο αλγόριθμος 3.4 συνδέει κάθε νέο αίτημα με το ενεργό τεταρτημόριο στο οποίο περιέχεται. Έστω u το νεό αίτημα και q_u το ενεργό τεταρτημόριο, επιπέδου j_u, με το οποίο συνδέεται. Όταν ο αριθμός των συνδεδεμένων αιτημάτων ξεπεράσει το 2^{j+1} ένα νεό κέντρο ανοίγει στο εσωτερικό το q_u, και τότε το quadrant γίνεται *ανοικτό*. Όπως έχουμε επαναλάβει επαναλημμένα, οι διαφορές μεταξύ των αλγορίθμων για το OFL βρίσκονται στον τρόπο με τον οποίο υλοποιούν την Αρχή 2. Διασθήτικα, ο αλγόριθμος στο [20] διασπά το χώρο σε όλο και μικρότερα τεταρτημόρια, ώστε να εντοπίζει όλο και καλύτερα την θέση του βέλτιστου κέντρου. Ωστόσο δεν το κάνει παντού, αλλά μόνο εκεί που εμφανίζεται η πλειονότητα των αιτημάτων. Το κύριο θεώρημα είναι το ακόλουθο:

Θεώρημα 3.7. *[20] Ο αλγόριθμος 3.4 είναι* $O(\log n)$ *competitive.*

Κεφάλαιο 4

Αλγόριθμοι Υποβοηθούμενοι από Μηχανική Μάθηση

Το Κεφάλαιο 4 επικεντρώνεται στο πεδίο των online αλγορίθμων υποβοηθούμενων από μηχανική μάθηση. Ξεκινάμε με μια σύντομη ιστορική αναδρομή που οδήγησε στην γέννηση του πεδιού και στην συνεχεία παρουσιάζουμε τα κυριότερα αποτελέσματα.

4.1 Εισαγωγή

Το τυπικό μοντέλο του online computation [21] υποθέτει ότι ο αλγόριθμος δεν έχει καμία γνώση σχετικά με τα δεδομένα εισόδου. Παρόλα αυτά, η υπόθεση αυτή δεν είναι ιδιαίτερα ρεαλιστική σε πραγματικά προβλήματα. Ο αλγόριθμος συχνά γνωρίζει το μέγεθος της είσοδου είτε ακόμα και ορισμένα από τα μελλοντικά στοιχεία. Για παράδειγμα, ας θεωρήσουμε ένα στιγμιότυπο του προβλήματος χωροθέτησης εγκαταστάσεων. Έστω, ότι βρισκόμαστε στην επιτροπή για την κατασκευή νέων ιατρικών κέντρων (εγκαταστάσεις) σε ένα *σχετικά* απομακρυσμένο σημείο της χώρας. Οι πολίτες των γύρω περιοχών μπορούν να ψηφίσουν για τις νέες τοποθεσίες κατασκευής. Ανεξάρτητα όμως από τον αριθμό των ψήφων και τις προτεινόμενες νέες θέσεις των κέντρων, γνωρίζουμε *προσεγγιστικά* τον πληθυσμό των γύρω περιοχών (δημογραφικά στοιχεία) και έτσι μπορούμε να ξεκινήσουμε την διαδικάσια, κατασκευάζοντας πρώτα ιατρικά κέντρα στα σημεία όπου ο πληθυσμός είναι πιο αυξημένος. Τυπικά, οι πολίτες φαντάζουν σαν μια κατανομή στον ευκλείδειο χώρο τόσο με κέντρα έντονης συσσώρευσης (πληθυσμού) όσο και με αραιώματα.

Τα δύο *γνωστά* μοντέλα που λαμβάνουν υπόψιν τους το παραπάνω σενάριο είναι το advice complexity *model* και το machine learned predictions *model*

4.1.1 Πολυπλοκότητα Πρόβλεψης

Παρόλο που η ανάλυση ανταγωνιστικότητας (competitive analysis) παραμένει ισχύουσα, απαιτείται ένα αυστηρό μοντέλο για τον προσδιορισμό της δύναμης και των

περιορισμών της επιπρόσθετης πληροφορίας. Ο όρος *πολυπλοκότητα πρόβλεψης (advice complexity)* που αποδίδεται στους Dobrev et al. [45], καθώς και μία σειρά από μετέπειτα μοντέλα ([46],[47]) που προτάθηκαν, στηρίχτηκαν στην συγκεκριμένη ιδέα της ποσοτικοποίησης της πληροφορίας. Υπό αυτό το μοντέλο υπολογισιμότητας, ο online αλγόριθμος λαμβάνει έναν αριθμό από bit που κωδικοποιούν *κάποια* πληροφορία σχετικά με την ακολουθία τον αιτημάτων.

Το είδος της πληροφορίας μπορεί να αφορά οτιδήποτε σχετίζεται με την ακολουθία των αιτημάτων και το εκάστοτε πρόβλημα. Παρόλα αυτά δεν είναι κάθε πληροφορία χρήσιμη και ούτε κάθε πληροφορία απαιτεί τον ίδιο αριθμό από bits για να κωδικοποιηθεί. Συνεπώς ο στόχος του μοντέλου είναι να προσδιορίσει την σχέση (trade-off) μεταξύ του μεγέθους του advice και της επίδοσης του αλγορίθμου. Σαφώς, η επίδοση του αλγορίθμου μπορεί μόνο να βελτιωθεί με έναν μεγάλο αριθμό από bits. Για περισσότερες πληροφορίες παραπέμπουμε τον αναγώστη στα [48], [49], [50].

4.1.2 Αναξιόπιστη Πρόβλεψη

Παρά την σημαντικότητα, το μοντέλο του advice complexity παρουσιάζει δύο σημαντικά μειονέκτημα. Το advice θεωρείται ως πληροφορία χωρίς σφάλματα, η οποία κωδικοποιεί κάποια ιδιότητα άρικτα συνδεδεμένη με την βέλτιστη λύση. Παραδείγματος χάρη, στο πρόβλημα του Facility Location, η πληροφορία μπορεί να αφορά την θέση του βέλτιστου κέντρο. Για να συμβεί αυτό, πρέπει το oracle να είναι σε θέση να λύσει το πρόβλημα βέλτιστα, κάτι που είναι αδύνατον. Με άλλα λόγια, το oracle αντιστοιχεί σε έναν πολυωνυμικό αλγόριθμος για το facility location. Το δεύτερο μειονέκτητα σχετίζεται και πάλι με την φερόμενη αξιοπιστία της πρόβλεψης. Δεδομένου ότι το advice είναι αλάθητο, κανένας λογικός online αλγόριθμος δεν θα επέλεγε να το αγνοήσει, διότι τότε η επίδοση του αλγορίθμου απλά θα χειροτέρευε.

Γενικότερα, η υπόθεση ότι η πρόβλεψη είναι πάντοτε αξιόπιστη και χωρίς λάθη είναι μη ρεαλιστική, τόσο για λόγους μετάδοσης όσο και για κακόβουλους αντιπάλους (adversaries) οι οποίοι προσπαθούν ηθελημένα να οδηγήσουν το αλγόριθμο σε υποβέλτισες επιλογές.

Ας πάρουμε για παράδειγμα το ski rental πρόβλημα, όπου πρέπει να επιλέξουμε μεταξύ των δύο: είτε να αγοράσουμε το ski την πρώτη μέρα με κόστος $b(uy)$ είτε να το νοικιάζουμε κάθε μέρα με κόστος 1. Ωστόσο επειδή ο καιρός είναι λίγο άσταστος (βρισκόμαστε στο τέλος του χειμώνα) δεν γνωρίζουμε για πόσες μέρες θα μπορούμε να κάνουμε ski και άρα αν είναι προτιμότερο να αγοράσουμε το ski την πρώτη μέρα των διακοπών μας. Ωστόσο, ένα μόνο bit πληροφορίας είναι αρκετό ώστε να μας οδηγήσει στην σωστή απόφαση, δηλαδή με 0 νοικιάζουμε κάθε μέρα ενώ με 1 αγοράζουμε τα ski την πρώτη μέρα. Παρόλα αυτά, αν το bit αυτό είναι λανθασμένο, ενδέχεται να καταλήξουμε με αυθαίρετα μεγάλο κόστος (να κάνουμε ski για το υπόλοιπο της ζωής μας).

4.1.3 Πρόβλεψη Υποβοηθούμενη από Μηχανική Μάθηση

Οι παρατηρήσεις αυτές οδήγησαν σε ένα νέο μοντέλο για τους online αλγορίθμους, όπου προβλέψεις υποβοηθούμενες από μηχανική μάθηση ενδέχεται να έχουν έναν αυθαίρετο αριθμό από λάθη, αλλά οι αλγόριθμοι οφείλουν να διακρίνουν πότε πρέπει να εμπιστευτούν το oracle και πότε να δρουν δίχως πρόβλεψη. Οι Lykouris, Vassilvitskii [22] και Purohit et al. [23] παρουσιάσαν έναν τρόπο σχεδίασης και ανάλυσης αλγορίθμων με μη αναξιόπιστα oracles που ικανοποιούν δύο σημαντικές ιδιότητες: (1) όταν οι προβλέψεις είναι σωστές ο αλγόριθμος οδηγείται στην βέλτιστη offline λύση (consistency), (2) διαφορετικά συγκλίνει προς την βέλτιστη online λύση (robusteness). Διασθητικά, η πρώτη ιδιότητα εξασφαλίζει ότι ο αλγόριθμος, υπό αληθείς προβλέψεις, βελτιώνεται, ενώ στην αντίθετη περίπτωση απαγορεύεται να αποδίδει αυθαίρετα κακά.

Ορισμός 4.1. *Competitive of an $\epsilon-$ assisted algorithm [22]: Το competitive ratio ενός αλγορίθμου $\epsilon-$ assisted α με την χρήση ενός predictor h στην ακολουθία σ, ορίζεται ως:*

$$CR_{\mathcal{A},l}(\epsilon) = \max_{\sigma,h\in\mathcal{H}_l(\epsilon)} CR_{\mathcal{A}(h)}(\sigma)$$

Ορισμός 4.2. *Consistency [22]: Ο αλγόριθμος $\mathcal{A}$ είναι $\beta-$consistent αν $CR_{\mathcal{A},l}(0) = \beta$.*

Ορισμός 4.3. *Robustness [22]: Ο αλγόριθμος $\mathcal{A}$ είναι $\alpha-$robust ως προς την συνάρτηση $\alpha(\cdot)$ αν $CR_{\mathcal{A},l}(\epsilon) = \alpha(\epsilon)$.*

Ορισμός 4.4. *Competitiveness [22]: Ο αλγόριθμος $\mathcal{A}$ είναι $\gamma-$competitive αν $CR_{\mathcal{A},l}(\epsilon) \leq \gamma$.*

Πρακτικά, το μοντέλο των online algorithms with machine learned advice [22] επιτρέπει την απόδειξη κάποιων worst case guarantees όταν υποθέσεις για τη είσοδο ή το oracle δεν συμβαίνουν στην πράξη. Ένα παράδειγμα υπόθεσης, ότι τα δεδομένα προέρχονται από μια στοχαστική κατανομή, μελετήθηκε για το πρόβλημα του online ταιριάσματος [51] και το bandit learning [52]. Και στις δύο περιπτώσεις αποδείχθηκαν βελτιωμένα guarantees όταν η κατανομή των δεδομένων είναι στοχαστική, ενώ επίσης έδειξαν ότι διατηρούνται τα γνωστά worst case guarantees, όταν η υπόθεση παύει να ισχύει. Ουσιαστικά, το συγκεκριμένο μοντέλο προσπαθήσει να ενσωματάσει τα αποτέλεσματα από τα πεδία του *Learning Theory* και του *Machine Learning* στο *Computer Science* διατηρώντας παράλληλα κάποια *worst case guarantees* όταν τα μοντέλα προβλέψεων αποτυγχάνουν.

4.2 Πρόσφατα Αποτελέσματα

Τα τελευταία χρόνια το μοντέλο των online algorithms with machine learned advice έχει αποκτήσει έντονη ερευνητική δραστηριότητα με σημαντικά αποτελέσματα.

Στην Ενότητα 4.2 κάνουμε μια σύντομη αναδρομή σε θεμελιώδη προβλήματα των online αλγορίθμων υπό την χρήση του συγκεκριμένου μοντέλου.

Το *ski rental* είναι ένα τυπικό πρόβλημα της κλάσης των rent-or-buy προβλήματων, όπου κάποιος καλείται να αποφασίσει μεταξύ μιας φθηνής προσωρινής λύσης και μιας δαπανηρής, αλλά μακροχρόνιας, λύσης (4.1.2). Ο βέλτιστος ντετερμινιστικός αλγόριθμος, break-even, νοικιάζει το ski για τις πρώτες $b-1$ μέρες και στην συνέχεια το αγοράζει στην τιμή b. Άμεσα παρατηρούμε ότι το competitive ratio του break-even είναι 2 καθώς και ότι είναι το καλύτερο δυνατό για την περίπτωση των ντετερμινιστικών αλγορίθμων. Οι Karlin et al. διατύπωσαν έναν πιθανοτικό αλγόριθμο με competitive ratio ίσο με $\frac{e}{e-1} \approx 1.58$, ο οποίος έδειξαν ότι είναι και βέλτιστο. Υπό το νέο μοντέλο των online algorithms with machine learned advice, οι Purohit et al. [23] διατύπωσαν δύο απλούς αλγορίθμους με διαφορετικά worst case guarantees. Ο πρώτος, ο ντετερμινιστικός, έχει competitive ratio ισό με $min\{\frac{1+\lambda}{\lambda}, (1+\lambda)+\frac{\eta}{(1-\lambda)OPT}\}$ και είναι $(1+1/\lambda)-$robust, $(1+\lambda)-$consistent ως προς την παράμετρο $\lambda \in (0,1)$. Ο δεύτερος, ο πιθανοτικός, έχει competitive ratio ισό με $min\{\frac{1}{1-e^{-(\lambda-1/b)}}, \frac{1}{1-e^{-\lambda}}(1+\frac{\eta}{OPT})\}$ και είναι $(\frac{1}{1-e^{-(\lambda-1/b)}})-$robust, $(\frac{\lambda}{1-e^{-\lambda}})-$consistent. Το πρόβλημα μελετήθηκε λιγό μετά και από τους Gollapudi et al. στο [53].

Το πρόβλημα της κρυφής μνήμης (paging) (2.3) είναι επίσης ένα θεμελιώδες πρόβλημα του online computation. Οι Fiat et al. [32] διατύπωσαν τον πρώτο πιθανοτικό αλγόριθμο με competitive ratio ίσο με $2H_k$, όπου H_k είναι ο k-οστός αρμονικός αριθμός και k ο αριθμός των σελίδων. Ο αλγόριθμος (Marker) λειτουργεί μαρκάροντας τις ζητούμενες σελίδες που βρίσκονται ήδη στην μνήμη και αφαιρώντας ομοιόμορφα μία από τις μη-μαρκαρισμένες στην περίπτωση σφάλματος. Οι Lykouris και Vassilvitskii [22] προσαρμόζοντας καταλλήλως έναν predictor στον αλγόριθμος των Fiat et al. [32], απέδειξαν ότι ο νέος αλγόριθμος (Predictive Marker) πετυχαίνει competitive ratio ίσο με $\min(2+4\sqrt{\eta/OPT}, 4H)$. Ο predictor ενημερώνει τον αλγόριθμο αναφορικά με την επόμενη χρονική στιγμή που η σελίδα θα ζητηθεί στο μέλλον (next arrival predictions). Στην offline λύση, ο βέλτιστος αλγόριθμος αφαιρεί την σελίδα με τον μεγαλύτερο μελλοντικό χρόνο άφιξης. Έτσι, όταν οι πρόβλεψεις είναι σωστές, ακολουθώντας την συμβουλή του oracle καταλήγουμε σε $O(1)$ λύση. Ωστόσο, ένα μόνο μικρό λάθος είναι δυνατό να οδηγήσει σε αυθαίρετα μεγαλό competitive ratio. Συνεπώς, είναι αναγκαίο να βρεθεί ένας κατάλληλος τρόπος ώστε ο αλγόριθμος να διακρίνει πότε πρέπει να ακολουθεί την πρόβλεψη και πότε να δρα χωρίς αυτήν. Οι Lykouris και Vassilvitskii εισήγαγαν την έννοια της eviction chain. Η αλυσίδα αυτή περιέχει την ακόλουθια των evictions, η οποία ξεκινά με την πρώτη σελίδα (clean page) που αφαιρέθηκε από την μνήμη. Η επανεμφάνιση της σελίδας στο μέλλον, θα προκαλέσει την αφαίρεση μιας δεύτερης σελίδας και εκείνη με την σείρα θα αφαίρεση μία ακόμα κ.ο.κ. Με αυτόν τον τρόπο καταφέρνουν να χωρίσουν τα σφάλματα

μνήμης, σε ένα σύνολο ανεξαρτήτων αλυσίδων (evictions chains). Ο αλγόριθμος που πρότειναν λειτουργεί ως εξής: για κάθε evictions chain, ο αλγόριθμος ακολουθεί τις προβλέψεις που δέχεται μέχρι το μήκος της να γίνει ίσο με $\Omega(logk)$, μετά αφαιρεί τις μημαρκαρισμένες σελίδες με ομοιόμορφο τρόπο, όπως στο [32]. Έπειτα, ο Rohatgi [54] διατύπωσε δύο βελτιωμένους αλγορίθμους, έναν με τεχνική μαρκαρίσματος (LMarker) και έναν χωρίς (LNonMarker). Το competitive ratio που πετυχαίνουν ισούται με $O(1+\min(1, \frac{\eta/OPT}{k}) \log k)$ (LMarker) και $O(1+\frac{\eta/OPT}{k} \log k)$ (LNonMarker) αντιστοίχως. Τελικά, ο Wei [55] έδειξε ότι συνδυάζοντας σε black-box fashion τον αλγόριθμο BlindOracle [22] που ακολουθεί τυφλά τις προβλέψεις του oracle και έναν $O(\log k)$-competitive αλγόριθμο για το online caching πετυχαίνει ακόμα καλύτερο competitive ratio. Συγκεκριμένα, απέδειξε την ύπαρξη ενός ντετερμινιστικού αλγορίθμου με $2\min(\min(1+2\frac{\eta}{OPT}, 4+\frac{4}{k-1}\frac{\eta}{OPT}), k)$ competitive ratio, και ενός πιθανοτικού με $(1+\gamma)\min(\min(1+2\frac{\eta}{OPT}, 4+\frac{4}{k-1}\frac{\eta}{OPT}), H_k),\ \gamma \in (0, 1/4)$.

Άλλα προβλήματα όπως το repeated posted-price auctions, μελετήθηκε από τους Medina και Vassilvitskii [56] ή το online scheduling που μελετήθηκε από τους Purohit et al. [23] , Lattanzi et al. [57] και Mitzenmacher [58]. Εναλλακτικά, παραπέμπουμε τον αναγνώστη στο [50], για μια ολοκληρωμένη μελέτη του νέου μοντέλου με πληθώρα διαφορετικών προβλημάτων, όπως το ski rental, online bidding, bin packing, list update. Ακόμα, σε μια σχετικά πρόσφατη εργασία, οι Bamas et al. [59] παρουσιάσαν έναν νέο τρόπο σχεδιασμού και ανάλυσης primal-dual αλγορίθμων ενσωματώνοντας το μοντέλο των machine learned predictions. Εδώ, το oracle δεν παρέχει μεμονωμένες συμβουλές στον αλγόριθμο, αλλά αντιθέτως παράγει μια δική του λύση. Έτσι, ο αλγόριθμος μέσω μιας παραμέτρου εμπιστοσύνης λ, κατασκευάζει την δική του λύση, η οποία ανάλογα με την τιμή του λ συγκλίνει είτε προς την λύση του oracle είτε προς εκείνη του κλασσικού primal dual αλγορίθμου. Πρόκειται για μια διαφορετική προσέγγιση του μοντέλου που δεν σχετίζεται άμεσα με το σφάλμα της πρόβλεψης, αλλά με το συνολικό κόστος της λύσης του oracle.

Περά από τα κλασικά αλγοριθμικά προβλήματα, η ιδέα του learning augmentation βρίσκει εφαρμογή και σε προβλήματα δομών δεδομένων. Ενδεικτικά παράδειγματα είναι τα ακόλουθα: learned indices [60], bloom filter [61] frequency estimation [62], nearest neighbor search [63].

Κεφάλαιο 5

Αλγόριθμος LaOFL

Στο Κεφάλαιο 5 παρουσιάζουμε τον αλγόριθμο για το πρόβλημα του Online Facility Location σε δυαδικά (binary) Hierarchical Well-Separated Tree (HST) (2.2) με την βοήθεια ενός μοντέλου προβλέψεων, το oracle. Πρόκειται για έναν απλό και φυσικό αλγόριθμο δομημένο πάνω στις αρχές (3.1) των online αλγορίθμων, τις οποίες και ενισχύει ενσωματώνοντας την πληροφορία που δέχεται από το oracle.

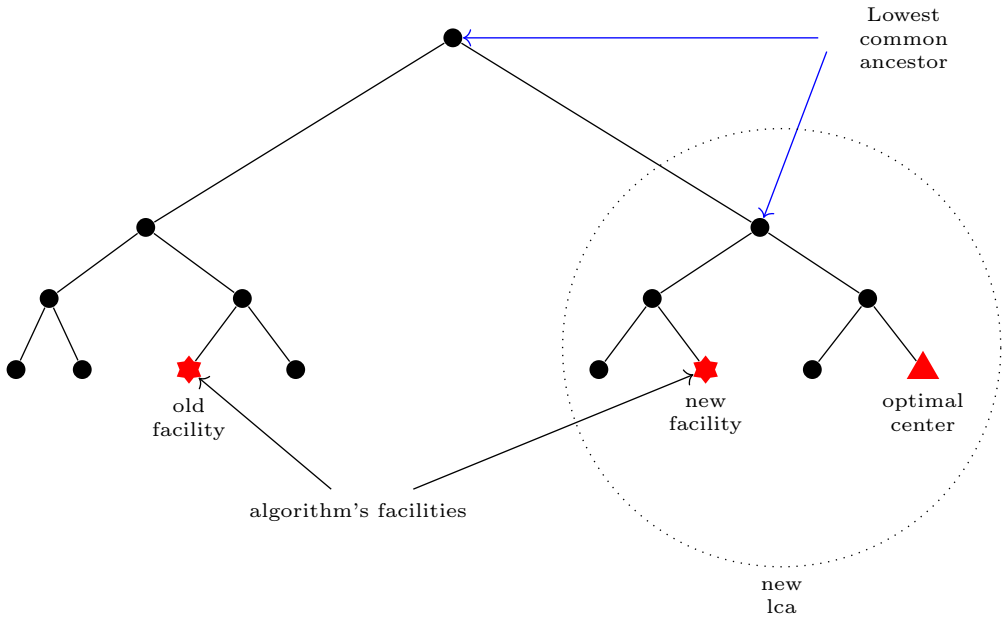
5.1 Εισαγωγή

Η ανάλυση του αλγορίθμου γίνεται σε φάσεις, με την κατασκευή ενός νέου κέντρου να σηματοδεί την έναρξη και τερματισμό τους. Εν γένει οι φάσεις αντιστοιχούν στις κοντινότερες αποστάσεις μεταξύ των κέντρων (facilities) που κατασκευάζει ο αλγόριθμος και του κέντρου της βέλτιστης λύσης (optimal center). Όταν όμως ο χώρος έχει την μορφή δέντρου, οι φάσεις αποκτούν μια απλή ερμηνεία. Μπορούμε άμεσα να τις ορίσουμε χρησιμοποιώντας τα επίπεδα του ίδιου του δέντρου. Πιο συγκεκριμένα, όταν ο κατώτατος κοινός πρόγονος (lowest common ancestor - lca) του νέου και του βέλτιστου κέντρου βρίσκεται στο επίπεδο i, θεωρούμε ότι ο αλγόριθμος διανύει την $i-$οστή φάση, βλπ Σχήμα 5.1.

Η χρήση των φάσεων εξυπηρετεί την ανάλυση του αλγορίθμου για δύο λόγους: αφενός γνωρίζουμε την ακριβή απόσταση από το βέλτιστο κέντρο, αφετέρου λόγω της ανεξαρτησίας (disjointness) τους μας επιτρέπεται να τις μελετάμε ξεχωριστά.

5.1.1 Συμβολισμοί και Ορισμοί

Η παραλλαγή που μελετήσαμε στην παρούσα εργασία αφορά την ύπαρξη ενός μόνο κέντρου στην βέλτιστη λύση, με ομοιόμορφα κόστη κατασκευής (uniform facility costs) νέων κέντρων, ίσο με f. Ο μετρικός χώρος $\mathcal{M}$ είναι ένα Hierarchical Well-Separated Tree με λόγο μείωσης (decreasing factor) της απόστασης μεταξύ των επιπέδων ίσο με k, δηλαδή ένα $k-$HST. Τα νέα αιτήματα επιτρέπεται να βρίσκονται σε οποιοδήποτε κόμβο του δέντρου, ωστόσο τα νέα κέντρα κατασκευάζονται αποκλειστικά στα φύλλα.



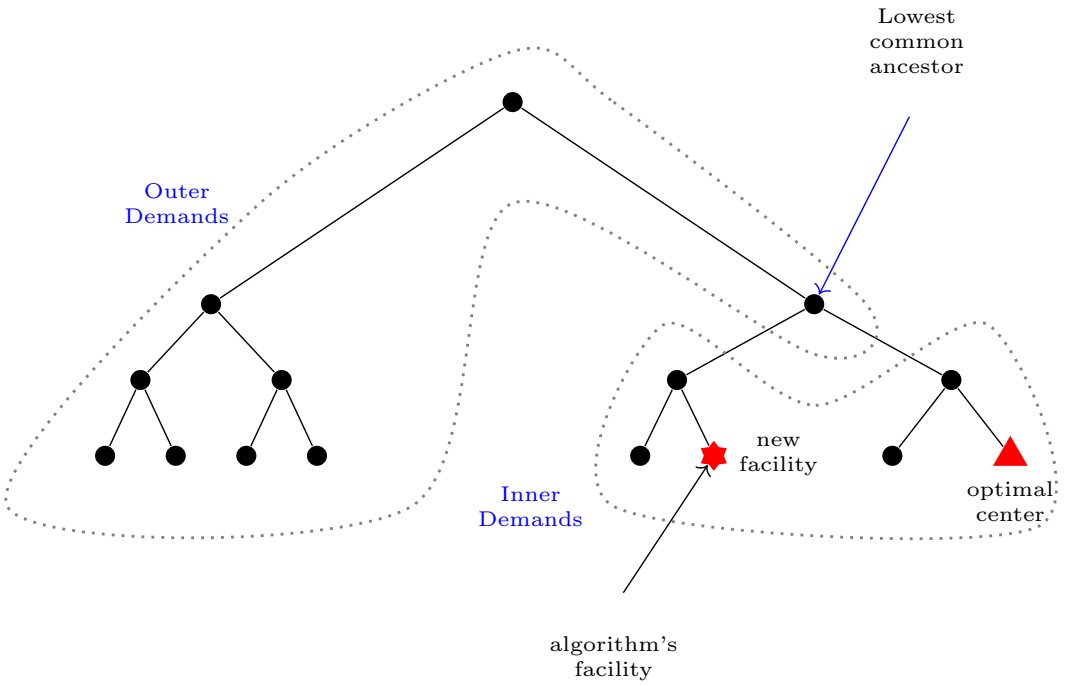
Σχήμα 5.1: *Δύο παραδείγματα κατώτερων κοινών προγόνων μεταξύ των εγκαταστάσεων του αλγορίθμου και του βέλιστου κέντρου.*

Προκείμενο να αποφευχθεί η σύγχυση θα χρησιμοποιούμε τον όρο **βέλτιστο κέντρο** (optimal center) για να αναφερόμαστε στο κέντρο της βέλτιστης λύσης και απλά τον όρο **κέντρα** (ή κέντρο) για τις εγκαταστάσεις (ή εγκατάσταση) (facilities) που κατασκευάζει ο αλγόριθμος. Με c συμβολίζουμε το βέλτιστο κέντρο, ενώ με $\mathcal{F}$ το σύνολο των εγκαταστάσεων/κέντρων που έχει κατασκευάσει ο αλγόριθμος. Η ελάχιστη απόσταση μεταξύ τους συμβολίζεται με $d(\mathcal{F}, c)$ και ορίζεται ως $d(\mathcal{F}, c) = \min_{f \in \mathcal{F}} d(f, c)$.

Οι κόμβοι του δέντρου συμβολίζονται με μικρά λατινικά γράμματα ενώ συχνά χρησιμοποιούμε έναν δείκτη για να δηλώσουμε το επίπεδο στο οποίο βρίσκονται. Για πάραδειγμα ο u_j είναι ένας κόμβος του $j-$επιπέδου. Αντίστοιχα, το υποδέντρο με ρίζα τον κόμβο u_j συμβολίζεται ως T_{u_j}.

Το σύνολο των αιτημάτων που βρίσκονται στους κόμβους του T_{u_j} επιφέρουν στον αλγόριθμο ένα κόστος σύνδεσης με τα κοντινότερα, ως προς αυτά, κέντρα, το οποίο ονομάζουμε potential. Με τον όρο $\text{Pot}(T_{u_j})$ συμβολίζουμε το συσσωρευμένο potential, διαφορετικά το συσσωρευμένο κόστος σύνδεσης, στο υποδέντρο T_{u_j}. Προφανώς, το συσσωρευμένο κόστος σύνδεσης στο T_{u_j} ενδέχεται να διαφέρει κατά πολύ από εκείνο της βέλτιστης λύσης. Αυτό οφείλεται βέβαια στην λανθασμένη επιλογή της θέσης του κέντρου που κατασκεύασε ο αλγόριθμος. Ωστόσο δεν επιφέρουν όλα τα αιτήματα του υποδέντρου T_{u_j} επιπρόσθετο κόστος στον αλγόριθμο. Ο διαχωρισμός τους διατυπώνεται στον ακόλουθο ορισμό και παρουσιάζεται στο Σχήμα 5.2.

Ορισμός 5.1. *Εξωτερικά (outer) ονομάζουμε τα αιτήματα όπου το κόστος σύνδεσης είναι ισό με εκείνο της βέλτιστς λύσης και εσωτερικά (inner) εκείνα όπου διαφέρει.*



Σχήμα 5.2: *Εσωτερικά και εξωτερικά αιτήματα*

Η σημαντικότητα της διάκρισης έγκειται στο ότι μόνο τα εσωτερικά αιτήματα ενδέχεται να αυξήσουν το κόστος του αλγορίθμου καθώς στα εξωτερικά το κόστος είναι ίσο με την βέλτιστη λύση. Παραδείγματος χάριν, αν ο αλγόριθμος εντοπίζει απευθείας την θέση του βέλτιστου κέντρου τότε όλα τα μελλοντικά αιτήματα χαρακτηρίζονται ως εξωτερικά και χρεώνουν κόστος ίσο με την βέλτιστη λύση.

5.2 Περιγραφή Αλγορίθμου

Κατά την άφιξη ενός νέου αιτήματος ο αλγόριθμος **LaALG** 5.5 καλείται να αποφασίσει αν θα συνδέσει το αίτημα με ένα υπάρχον κέντρο ή αν πρώτα θα κατασκευάσει ένα νέο στην γύρω περιοχή. Η απόφαση του εξαρτάται από το συσσωρεύμενο potential στα υποδέντρα του HST.

Πρώτα λοιπόν, ενημερώνει το potential όλων των υποδέντρων που περιέχουν το αίτημα στο εσωτερικό τους, βλπ Σχήμα 5.3. Αν σε κάποιο από αυτά παρατηρηθεί ότι παραβιάστηκε η συνθήκη $\text{Pot}(T) \geq f$, τότε ο αλγόριθμος καλείται να κατασκευάσει ένα νέο κέντρο ώστε να εξυπηρετήσει τα μελλοντικά αιτήματα με λιγότερο κόστος. Ο λόγος για την απόφαση του είναι φυσικά η Αρχή 2.

Στο επόμενο βήμα ο αλγόριθμος αναζητά το χαμηλότερο υποδέντρο που επιβαρύνεται περισσότερο από το συνολικό κόστος σύνδεσης. Ξεκινά, διασχίζοντας το δέντρο και επιλέγοντας τον κόμβο-παιδί που ικανοποιεί την συνθήκη $\text{Pot}(T) \geq f$ (1). Όταν πλέον κανένας από τους δύο κόμβους-παιδιά δεν ικανοποιούν την συνθήκη (1), τότε

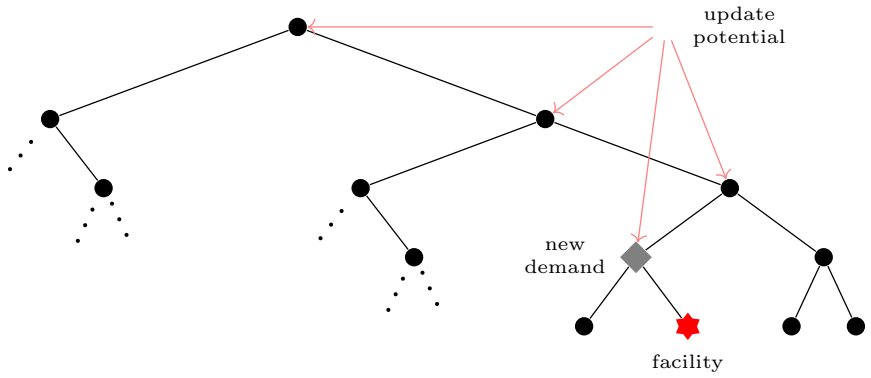
Αλγοριθμος 5.5: *LaOFL Ντετερμινιστικός αλγόριθμος Υποβοηθούμενος από Μηχανική Μάθηση*

$\mathcal{F} \leftarrow \emptyset$
while a new demand u_j arrives **do**
  $UpdatePotential(u_j)$;
  if $\exists$ subtree $T : Pot(T) \geq f$ **then**
    // Find the lowest violated subtree
    $T_{v_i} \leftarrow FindLowestSubtree()$;

    // Ask the oracle for advice
    $advice \leftarrow NewAdvice(T_{v_i})$;

    // Construct advices path
    $path \leftarrow ConstructPath(advice)$;

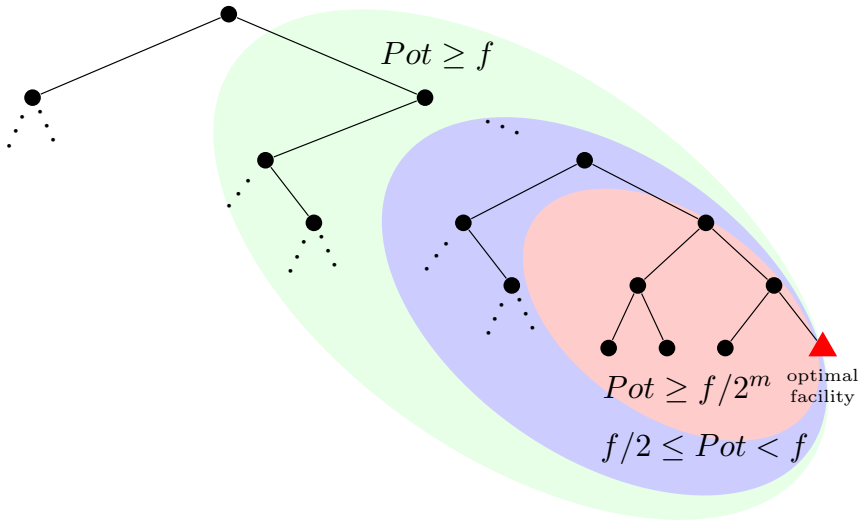
    // Open a new facility at v_h
  end if
  // Assign u_j *to the closest facility*
  $UpdatePotential(u_j)$;
end while



Σχήμα 5.3: **Βήμα 1**: *Ενημέρωση του potential των υποδέντρων που περιέχουν το νέο αίτημα.*

εκείνη τροποποιείται σε $\text{Pot}(T) \geq f/2$ και η διαδικασία επαναλαμβάνεται. Κάθε φορά που η συνθήκη παύει να ισχύει, τότε τροποποιείται εκ νέου μειώνοντας στο μισό το κατώφλι της τιμής του potential ($\text{Pot}(T) \geq f/2^m$), με την σχέση $2^m \leq k$ να παραμένει ισχύουσα. Σημειώνουμε ότι η τιμή k αντιστοιχεί στον λόγο μείωσης της απόστασης μεταξύ των επιπέδων του $k-$HST. Τελικά, ο κόμβος που θα σταματήσει ο αλγόριθμος, v_j, θα είναι ο εκπρόσωπος (representative) της επόμενης φάσης: σε ένα από τα φύλλα του υποδέντρου T_{v_j} θα κατασκευαστεί το νέο κέντρο, βλπ Σχήμα 5.4.

Σε αυτό το σημείο ο αλγόριθμος έχει κατορθώσει, αξιοποιώντας **μόνο** την πληροφορία από την θέση των αιτημάτων, να εντοπίσει το χαμηλότερο υποδέντρο που περιέχει το βέλτιστο κέντρο. Προκειμένου όμως να ξεπεράσει το κάτω φράγμα (Ενότητα 3.2)



Σχήμα 5.4: **Βήμα 2**: *Εντοπισμός του χαμηλότερου επιβαρυμένου υποδέντρου.*

των κλασσικών online αλγορίθμων απαιτείται η συμβολή του oracle.

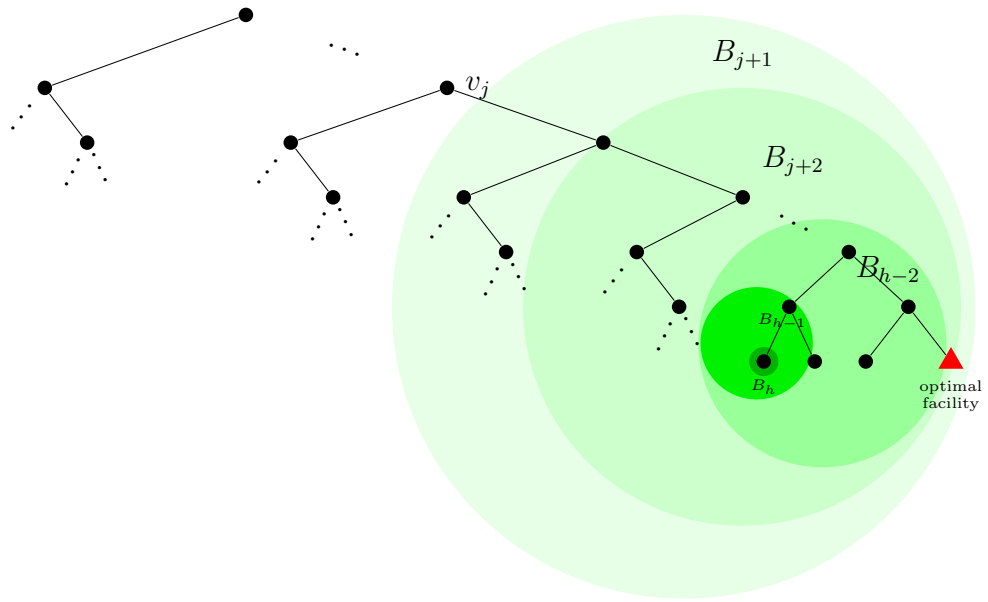
5.2.1 Η έννοια της Πρόβλεψης

Παρουσιάζουμε αρχικά την ποιοτική μορφή των προβλέψεων (advices) που εμπίπτει στους γενικούς μετρικούς χώρους και στην συνέχεια την ειδικεύουμε για την περίπτωση του Hierarchical Well-Separated Tree (HST). Υπενθυμίζουμε ότι ο v_j είναι ο κόμβος στον οποίο τερμάτισε η διαδικασία εντοπισμού του χαμηλότερου επιβαρυμένου υποδέντρου *(Βήμα 2)*.

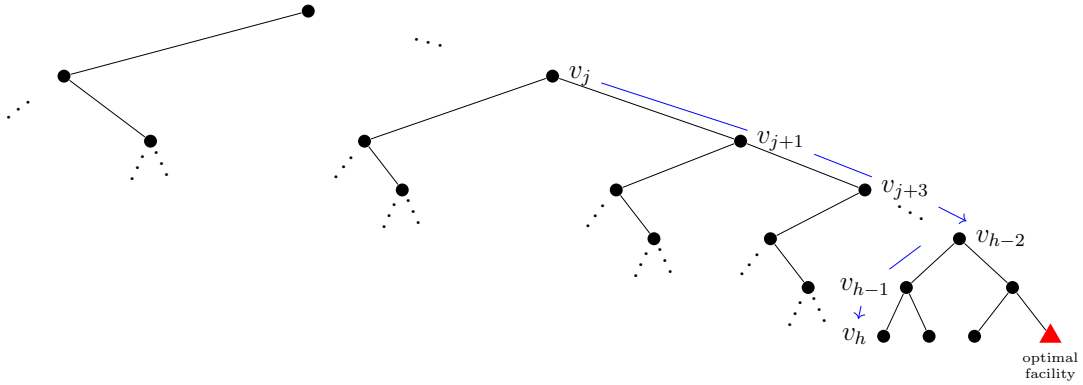
Ο αλγόριθμος ζητά από το oracle να υποδείξει την σφαίρα B_j= Ball($\frac{L_j}{2}$) με ακτίνα ($\frac{L_j}{2}$) όπου μελλοντικά θα βρεθούν τα περισσότερα αιτήματα (majority of demands). Έπειτα, ζητά μία μικρότερη σφαίρα B_{j+1}, εσωτερική πλέον στην B_j, με ακτίνα $\frac{L_j}{2k}$ ώστε αναδρομικά να καταλήγει στην μικρότερη σφαίρα με ακτίνα L_h, πρακτικά σημείο, βλπ Σχήμα 5.5. Στο σημείο αυτό ο αλγόριθμος κατασκευάζει το νέο κέντρο.

Στην περιπτωσή του Hierarchical Well-Separated Tree (HST) οι σφαίρες αποκτούν μία απλή και διασθητική ερμηνεία. Εν γένει, κάθε σφαίρα αντιστοιχεί σε ένα από τα δύο παιδιά του κόμβου. Ειδικότερα, το κέντρο της σφαίρας B_j αντιστοιχεί σε έναν από τους δύο κόμβους-παιδιά του v_j, έστω τον v_{j+1}. Αντιστοίχως το κέντρο της B_{j+1} αντιστοιχεί σε έναν κόμβο-παιδί του v_{j+1}. Έτσι σχηματίζεται ένα μονοπάτι (path=) $p_{v_j} = v_j, v_{j+1}, v_{j+2}, \cdots, v_{h-1}, v_h$ που ξεκινάει από τον κόμβο v_j και καταλήγει στο φύλλο v_h, βλπ Σχήμα 5.6.

Ορισμός 5.2. *Η έννοια της Πρόβλεψης (Notion of Advice): Το oracle υποδεικνύει το μονοπάτι με την πλειονότητα (majority) των μελλοντικών αιτημάτων να περιέχεται στα υποδέντρα που ορίζει.*



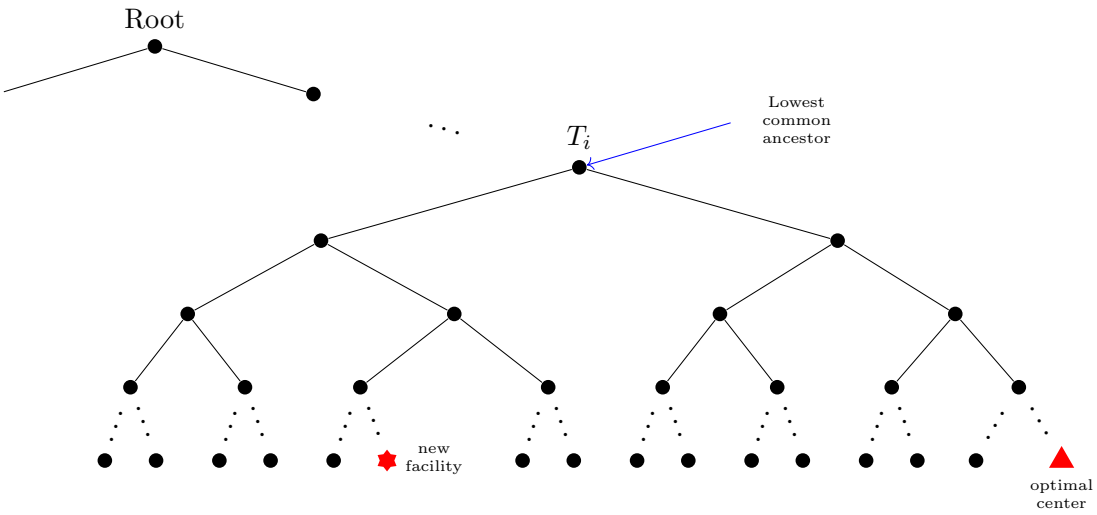
Σχήμα 5.5: **Βήμα 3**: *Οι προβλέψεις του oracle.*



Σχήμα 5.6: **Βήμα 4**: *Το μονοπάτι που σχηματίζουν οι προβλέψεις του oracle.*

Μετά λοιπόν την ολοκλήρωση των βημάτων, το νέο κέντρο βρίσκεται σε απόσταση $2\,L_i$ από το βέλτιστο, δηλαδή ο $lca(c, v_h)$ (least common ancestor) είναι ένας κόμβος του επιπέδου i, βλπ Σχήμα 5.7. Φυσικά ο αλγόριθμος δεν έχει τρόπο να γνωρίζει το επίπεδο i του lca, διότι αυτομάτως θα γνώριζε και την ακριβή θέση του βέλτιστου κέντρου. Η μόνη πληροφορία που έχει, και είναι μάλιστα βέβαιος για την γνησιότητα της (Λήμμα 6.1), είναι για τον κόμβο v_j. Δηλαδή, ότι το υποδέντρο T_{v_j} περιέχει σε κάποιο από τα φύλλα του το βέλτιστο κέντρο.

Η ανάλυση που ακολουθεί αφορά εξ ολοκλήρου το υποδέντρο T_i καθώς οποιοδήποτε αίτημα βρίσκεται εκτός του υποδέντρου χαρακτηρίζεται ως εξωτερικό (outer) και σύμφωνα με τον Ορισμό 5.1 έχει κόστος σύνδεσης ίσο με την βέλτιστη λύση.



Σχήμα 5.7: **Βήμα 5**: *Κατασκευή νέου κέντρου.*

Κεφάλαιο 6

Ανάλυση LaOFL Αλγορίθμου

Το Κεφάλαιο 6 επικεντρώνεται στην απόδειξη της ορθότητα του αλγορίθμου LaOFL [5.5]. Στην Ενότητα 6.1 παρουσιάζουμε την ανάλυση του LaOFL, όπου για λόγους ευκολίας, διασπάται στην απόδειξη των επιμέρους βημάτων του αλγορίθμου. Στην Ενότητα 6.2 αναφέρουμε, και ύστερα αποδεικνύουμε, τα απαραίτητα κάτω φράγματα αναφορικά με την πρόβλεψη του oracle και το σφάλμα στην περίπτωση λάθους.

6.1 Απόδειξη Ορθότητας του LaOFL Αλγορίθμου

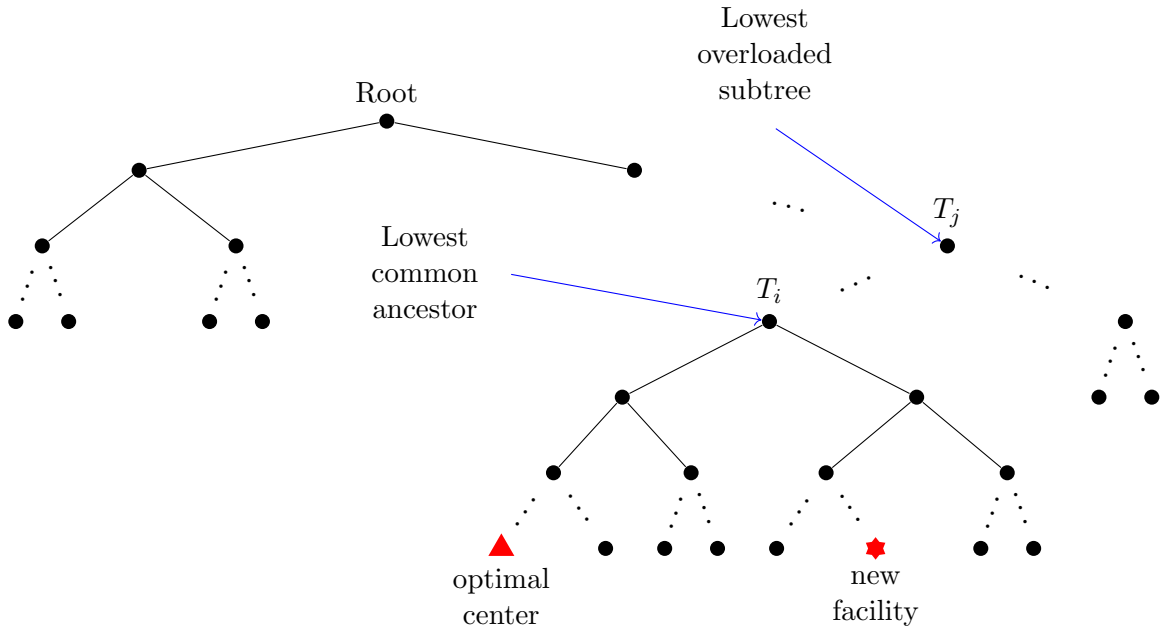
Υπενθυμίζουμε ότι η ανάλυση πραγματοποιείται απλοκειστικά στο υποδέντρο που ορίζει ο κατώτατος κοινός πρόγονος του νέου κέντρου και του βέλτιστου, βλπ Σχήμα 6.1. Φυσικά ο αλγόριθμος δεν έχει τρόπο να γνωρίζει το επίπεδο i του lca, διότι αυτομάτως θα γνώριζε και την ακριβή θέση του βέλτιστου κέντρου. Η μόνη πληροφορία που έχει είναι για τον κόμβο v_j, στον οποίο τερμάτισε η διαδικασία εντοπισμού του χαμηλότερου επιβαρυμένου υποδέντρου T_j.

6.1.1 Απόδειξη Ορθότητας του Βήματος Εντοπισμού

Το πρώτο κομμάτι της ανάλυση αφορά την ορθότητα του βήματος εντοπισμού του χαμηλότερου επιβαρυμένου υποδέντρου. Η σημαντικότητα του συγκεκριμένου βήματος έγκειται στο ότι ο αλγόριθμος χρησιμοποιώντας **αποκλειστικά** την πληροφορία των αιτημάτων εντοπίζει το χαμηλότερο υποδέντρο που περιέχει το βέλτιστο κέντρο. Ουσιαστικά μέχρι αυτό το σήμειο, συμπεριφέρεται σαν κλασσικός online αλγόριθμος. Ακολουθεί το Λήμμα 6.1 που αποδεικνύει την ορθότητα του βήματος.

Λήμμα 6.1. *Το υποδέντρο T_j περιέχει το βέλτιστο κέντρο.*

Απόδειξη. Το βήμα εντοπισμού ξεκινάει μόλις παρατηρηθεί ότι σε κάποιο υποδέντρο του HST το συσσωρευμένο potential υπερβαίνει την τιμή f, δηλαδή μόλις παραβιαστεί η



Σχήμα 6.1: *Το δέντρο της απόδειξης.*

Αρχή 2. Το υποδέντρο T_j στο οποίο καταλήγει ικανοποιεί την συνθήκη $\text{Pot}(T_j) \geq \frac{f}{2^m}$ για την μεγαλύτερη τιμή του m, ώστε $2^m \leq k(1)$.

Τότε, το T_j περιέχει πράγματι το βέλτιστο κέντρο. Διαφορετικά, τα αιτημάτα του υποδέντρου χρεώνουν την βέλτιστη λύση με κόστος σύνδεσης $\text{Asg}^* \geq k\frac{f}{2^m} \overset{(1)}{\Longrightarrow} \text{Asg}^*(T_j) \geq f$. Άτοπο, διότι τότε η βέλτιστη λύση δεν είναι πράγματι βέλτιστη. □

6.1.2 Απόδειξη Ορθότητας της Πρόβλεψης

Το δεύτερο βήμα αφορά την ορθότητα της πρόβλεψης. Ο στόχος του oracle είναι να συμβουλέψει τον αλγόριθμο ώστε να εντοπίσει ταχύτερα την θέση του βέλτιστου κέντρο και συνεπώς να μειώσει το competitive ratio. Επομένως, όταν η πρόβλεψη είναι αληθής, το κόστος της λύσης που παράγει ο αλγόριθμος θα πρέπει να ισό, κατά ένα σταθερό παράγοντα, με εκείνο της βέλτιστης λύσης.

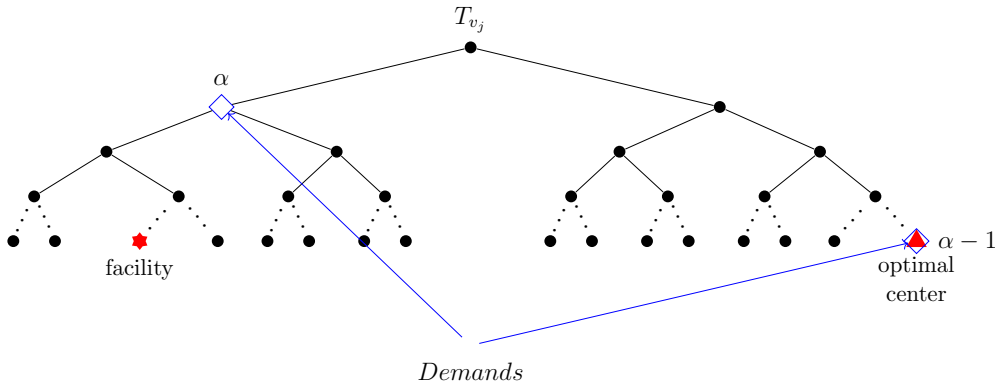
Λήμμα 6.2. *Κάθε σωστή πρόβλεψη οδηγεί τον αλγόριθμο ένα επίπεδο πιο κοντά στο βέλτιστο κέντρο ή λύνει το πρόβλημα με $O(1)$ competitive ratio.*

Απόδειξη. Το προηγούμενο βήμα οδήγησε τον αλγόριθμο στο υποδέντρο T_{v_j}. Από το Λήμμα 6.1, το βέλτιστο κέντρο βρίσκεται σε κάποιο από τα φύλλα του T_{v_j}. Σε αυτό το σημείο, ο αλγόριθμος δέχεται την πρόβλεψη από το oracle, η οποία του υποδεικνύει το υποδέντρο (αριστερό/δεξιό) όπου ανήκει η πλειονότητα (majority) των μελλοντικών αιτημάτων.

Υπάρχουν λοιπόν δύο περιπτώσεις: είτε αριστερά είτε δεξία. Η τετριμμένη περίπτωση συναντάται όταν το υποδέντρο με την πλειονότητα των αιτημάτων είναι εκείνο που

περιέχει και το βέλτιστο κέντρο. Από την υπόθεση του Λήμματος (η πρόβλεψη είναι σωστή), τότε πράγματι ο αλγόριθμος οδηγείται ένα επίπεδο πιο κοντά στο βέλτιστο κέντρο. Θα δείξουμε ότι και στην άλλη περίπτωση το κόστος της λύσης που παράγει ο αλγόριθμος είναι εώς ένα σταθερό παράγοντα (up to a constant factor) ίσο με της βέλτιστη λύση.

Υποθέτουμε, χωρίς βλάβη της γενικότητας, ότι το βέλτιστο κέντρο ανήκει στο δεξί υποδέντρο ενώ το oracle υπέδειξε το αριστερό. Το *χειρότερο σενάριο* (worst case scenario) ισοδυναμεί με α αιτήματα στον αριστερό κόμβο-παιδί του v_j και $(\alpha - 1)$ στην **θέση** του βέλτιστου κέντρου. Ο λόγος που χαρακτηρίζει το σενάριο αυτό ως χειρότερο είναι ότι, ενώ η πρόβλεψη του oracle παραμένει αληθής, η βέλτιστη λύση χρεώνεται το μικρότερο δυνατό κόστος.



Σχήμα 6.2: *Το χειρότερο σενάριο του Λήμματος 6.2*

Το κρίσιμο σημείο είναι ότι το κόστος που χρεώνεται ο αλγόριθμος σε αυτό το σενάριο, είναι το μέγιστο δυνατό. Με αυτό τον τρόπο εξασφαλίζεται ότι οποιοδήποτε άλλο στιγμιότυπο (instance) οδηγεί σε competitive ratio το πολύ ίσο με του worst case scenario (WCS).

$$\begin{aligned} \mathrm{OPT} &= (\alpha - 1)\, 0 + \alpha\, (2L_i - L_{i+1}) \\ &\geq \alpha\, (2k\, L_{i+1} - L_{i+1}) = \alpha\, (2k - 1)\, L_{i+1} \end{aligned} \tag{6.1}$$

$$\begin{aligned} \mathrm{ALG} &= (\alpha - 1)\, 2\, L_i + \alpha\, L_{i+1} \\ &\leq (\alpha - 1)\, (2\, 2k\, L_{i+1}) + \alpha\, L_{i+1} \\ &< \alpha\, (4k + 1)\, L_{i+1} \end{aligned} \tag{6.2}$$

$$\frac{\mathrm{ALG}}{\mathrm{OPT}} < \frac{4k+1}{2k-1} \leq \frac{4k-2+3}{2k-1} = 2 + \frac{3}{2k-1} \leq 2 + \frac{3}{2\,(2-1)} = 3 \tag{6.3}$$

□

Από το Λήμμα 6.2 παρατηρούμε ότι ακόμα και μία μόνο πρόβλεψη ενδέχεται να λύσει το Online Facility Location (OFL). Για παράδειγμα αν βρισκόμαστε στο επίπεδο j και το υποδέντρο που υποδεικνύει το oracle, δηλαδή εκείνο με την πλειονότητα των αιτημάτων, δεν περίεχει το βέλτιστο κέντρο, τότε οποιοδήποτε από τα 2^{h-i} φύλλα και αν επιλεγεί για την θέση του νέου κέντρου, η λύση που προκύπτει έχει $O(1)$ competitive ratio. Μπορούμε λοιπόν να απορρίψουμε αυτήν την κατηγορία στιγμιότυπων ως τετριμμένη καθώς απαιτείται μικρότερος αριθμός από προβλέψεις για την λύση του προβλήματος. Συνοψίζοντας, καταλήγουμε στην ακόλουθη πρόταση.

Πρόταση 6.3. *Σε κάθε στιγμιότυπο το βέλτιστο κέντρο βρίσκεται πάντα στο φύλλο που καταλήγει το μονοπάτι με την πλειονότητα των μελλοντικών αιτημάτων.*

Σύμφωνα με όσα αναφέραμε προηγούμενως, αν το βέλτιστο κέντρο δεν βρίσκεται στο φύλλο που καταλήγει η ακολουθία των προβλέψεων, τότε *είτε* η λύση δεν είναι βέλτιστη *είτε* το στιγμιότυπο είναι τετριμμένο, δηλαδή υπάρχουν παραπάνω από μία ισοδύναμες λύσεις. **Σημειώνουμε** ότι με τον όρο «το μονοπάτι με την πλειονότητα» των αιτημάτων, δεν αναφερόμαστε συγκεκριμένα στους κόμβους του μονοπατιού αυτούς καθαυτούς, αλλά στα υποδέντρα που ορίζουν. Με αλλά λόγια, η πλειονότητα των μελλοντικών αιτημάτων βρίσκεται εσωτερικά των υποδέντρων που ορίζουν οι κόμβοι και όχι στους κόμβους καθεαυτούς.

Για το υπόλοιπο της ανάλυσης θεωρούμε, χωρίς βλάβη της γενικότητας, ότι το μονοπάτι που περιέχει την πλειονότητα (majority) των μελλοντικών αιτημάτων καταλήγει πάντα στην θέση του βέλτιστου κέντρου. Δύο άμεσα συμπεράσματα του Λήμματος 6.2 είναι τα ακόλουθα.

Συμπέρασμα 6.1. *Αν ο αλγόριθμος βρίσκεται στο επίπεδο j, $h - j$ σωστές προβλέψεις αρκούν για να τον οδηγήσουν στο βέλτιστο κέντρο.*

Συμπέρασμα 6.2. *Ο αλγόριθμος 5.5 είναι $O(1)$ consistent.*

Ο στόχος του αλγορίθμου είναι ξεκάθαρος: προσπαθεί να εντοπίσει το μονοπάτι που καταλήγει στο βέλτιστο κέντρο με τον ταχύτερο δυνατό τρόπο. Έτσι όσο πιο γρήγορη είναι η μετάβαση μεταξύ των επιπέδων, άρα και η εναλλαγή των φάσεων, τόσο μικρότερο θα είναι το κόστος και κατ΄ επέκταση το competitive ratio. Προκειμένου να το πετύχει απαιτείται τόσο πληροφορία από την θέση των αιτημάτων όσο και οι προβλέψεις που δέχεται από το oracle.

Το επόμενο βήμα στην ανάλυση της ορθότητας του αλγορίθμου απαιτεί την επιλογή ενός κατάλληλου σφάλματος που αποζημιώνει τον αλγόριθμο για το κόστος που επιφέρουν οι λανθασμένες προβλέψεις του oracle. Λόγω της φύσης των προβλέψεων δεν μας ενδιαφέρει το πλήθος ή θέση των αιτημάτων αλλά μόνο το επίπεδο όπου γίνεται πρώτη φορά λάθος. Σε εκείνο το σημείο ο αλγόριθμος αναγκάζεται επιλέξει το λάθος υποδέντρο για την κατασκευή του νέου κέντρου. Ως αποτέλεσμα, ο κατώτατος κοινός πρόγονος $\text{lca}(v_h, c)$, v_i, απέχει από το βέλτιστο κέντρο απόσταση ίση με $2 \cdot L_i$, βλπ

Σχήμα 6.1. Πριν περάσουμε λοιπόν στο σφάλμα κρίναμε απαραίτητο να απαντήσουμε σε ένα κρίσιμο ερώτημα που προκύπτει.

Ας υποθέσουμε ότι ο v_j είναι πράγματι ο κόμβος στον οποίο σταμάτησε το βήμα εντοπισμού, Ενότητα 6.1.1. Το Συμπέρασμα 6.1 μας λέει ότι απαιτούνται $h - j$ προβλέψεις για την διαμορφωση μονοπατιού από τον κόμβο v_j εώς ένα από τα φύλλα του υποδέντρου T_{v_j}. Επιλέγοντας όμως ένα μονάχα φύλλο το μονοπάτι διαμορφώνεται αυτομάτως. Γενικότερα, επιλέγοντας κόμβους χαμηλότερων επιπέδων τα μονοπάτια που συνδέουν τον v_j με οποιοδήποτε κόμβο χαμηλότερου επιπέδου ορίζονται μονοσήμαντα στο δέντρο. Με άλλα λόγια, δυο κόμβοι συνδέονται υποχρεωτικά με ένα μόνο μονοπάτι.

Συνεπώς, ζητώντας από το oracle προβλέψεις για κάποιο ενδιάμεσο επίπεδο, έστω k, το μονοπάτι μεταξύ των v_j, v_k ορίζεται μονοσήμαντα. Εύλογα γεννάται η ακόλουθη απορία: κατά πόσο μπορεί ο αλγόριθμος να κερδίσει σε φάσεις παραλείποντας προβλέψεις ενδιάμεσων επιπέδων; Στο Λήμμα 1.3 εξετάζουμε το σενάριο αυτό και αποδεικνύουμε ότι η πληροφορία που χάνει ο αλγόριθμος για την κατανομή των αιτημάτων στα ενδιάμεσα επίπεδα είναι αρκετή ώστε το competitive ratio να παραμείνει $\Omega(\frac{\log n}{\log \log n})$. Συνεπώς, δεν επιτρέπεται στον αλγόριθμο να παραλείψει κανένα επίπεδο διότι ενδέχεται να πληρώσει πολύ πιο ακριβά τα μελλοντικά αιτήματα. Άμεσα συνεπάγεται και η ακόλουθη πρόταση.

Συμπέρασμα 6.3. *Προκειμένου ο αλγόριθμος να παραμείνει $O(1)$ competitive, οι πρόβλεψεις του oracle πρέπει να αφορούν διαδοχικά επίπεδα στο δέντρο.*

6.1.3 Απόδειξη του Competitive Ratio

Πλεόν, συνεχίζουμε με την απόδειξη του competitive ratio που πετυχαίνει ο αλγόριθμος 5.5. Λόγω του Συμπεράσματος 6.1 χαρακτηρίζουμε κάθε ακολουθία απο προβλέψεις που δεν οδηγεί απευθείας στο βέλτιστο κέντρο ως λανθασμένη. Μένει να ορίσουμε ένα **κατάλληλο** και **δίκαιο** σφάλμα που χρεώνουμε στο oracle για τις λάθος προβλέψης του. Συνεχίζουμε με τον ορισμό του σφάλματος και την απόδειξη του βασικού θεωρήματος του Κεφαλαίου 6.

Ορισμός 6.1. *Το σφάλμα της πρόβλεψης (Oracle's Error): Το σφάλμα του oracle ισούται με το κόστος μεταφοράς όσων αιτημάτων απαιτούνται ώστε η πρόβλεψη να γίνει αληθής.*

Θεώρημα 6.1. *Ο αλγόριθμος 5.5 έχει $O(1)$ competitive ratio.*

Απόδειξη. Εφόσον ο αλγόριθμος δεν κατέληξε στο βέλτιστο κέντρο και τα αιτήματα δεν τελειώσαν ακόμα, από το Συμπέρασμα 6.1 συνεπάγεται ότι κάποια πρόβλεψη ήταν λανθασμένη. Υποθέτουμε ότι κατώτερος κοινός πρόγονος του νέου κέντρου και του βέλτιστου κέντρου βρίσκεται στο επίπεδο i, $lca(v_h, c) = v_i$. Άρα η (πρώτη) λανθασμένη πρόβλεψη είναι εκείνη του v_{i+1}. Επισημαίνουμε ότι οποιαδήποτε από τις

ακόλουθες προβλέψεις και να ΄ναι λανθασμένη δεν μπορεί να επηρεάσει περαιτέρω το competitive ratio. Συνεπώς στην απόδειξη του θεωρήματος θα ασχοληθούμε μόνο με την συγκεκριμένη πρόβλεψη.

Υποθέτουμε χωρίς βλάβη της γενικότητας ότι το νέο κέντρο βρίσκεται στο αριστερό υποδέντρο του v_i ενώ το βέλτιστο κέντρο στο δεξί, όπως ακριβώς στο Σχήμα 6.1. Ορίζουμε ως n_L, n_R το **πραγματικό** πλήθος των αιτημάτων στο αριστερό και στο δεξί υποδέντρο του T_{v_i} και $v_L(= v_{i+1}), v_R$ τους αντίστοιχους κόμβους-παιδιά. Λόγω του λάθους της πρόβλεψης, ισχύει η σχέση: $n_R > n_L$.

Παρόμοια με το Λήμμα 6.2, εξετάζουμε μόνο το χειρότερο δυνατό σενάριο (worst case instance). Το competitive ratio που καταλήγουμε, λειτουργεί ως άνω-φράγμα (upper bound) για οποιοδήποτε άλλο στιγμιότυπο του προβλήματος. Τώρα, το wcs είναι το ακόλουθο: n_R αιτήματα βρίσκονται στην θέση του βέλτιστου κέντρου και n_L στον v_L.

Αφού λοιπόν η πρόβλεψη που λάβαμε ήταν λανθασμένη, θα χρεώσουμε στο oracle το κόστος μεταφοράς $(n_R - n_L)$ αιτημάτων από την *δεξία* στην *αριστερή* πλευρά. Τα συγκεκριμένα αιτήματα τοποθετούνται σε κόμβους αντίστοιχων επιπέδων με εκείνων από όπου τα παρέλαβε. Εκτελώντας τις πράξεις άμεσα καταλήγουμε στο ζητούμενο του θεωρήματος.

Διαισθητικά, το σφάλμα μετασχηματίζει εικονικά την κατανομή των αιτημάτων στο αριστερό υποδέντρο σε μία κατανομή παρόμοια με του δεξιού. Ωστόσο, όπως αποδεικνύεται και στην συνέχεια, η συγκεκριμένη μορφή σφάλματος ισχύει αυστηρά, καθώς οποιοδήποτε άλλο σφάλμα δεν αρκεί να οδηγήσει σε $O(1)$ competitive ratio.

□

6.2 Κάτω Φράγματα

Πλέον, αποδείξαμε την ύπαρξη ενός αλγορίθμου με σταθερό (constant) competitive ratio για το Online Facility Location με την βοήθεια ενός μοντέλου προβλέψεων, oracle. Δεν αρκεί όμως μόνο αυτό. Πρέπει επίσης να αποδειχθεί ότι ο αλγόριθμος 5.5 είναι ο μοναδικός που το πετυχαίνει. Με άλλα λόγια, οποιοσδήποτε άλλος συνδυασμός πρόβλεψης (advice) και σφάλματος (error) οδηγεί σε μεγαλύτερο competitive ratio.

Προκειμένου να το πετύχουμε κινούμαστε σε δύο επίπεδα. Πρώτα δείχνουμε ότι η πρόβλεψη (advice) παρέχει τον ελάχιστο αριθμό πληροφορίας και έπειτα, κρατώντας σταθερό το είδος της πρόβλεψης, δηλαδή την πλειονότητα των μελλοντικών αιτημάτων, αποδεικνύουμε ότι οποιοδήποτε άλλο σφάλμα δεν αρκεί να αποζημείωσει το κόστος που χρεώνεται ο αλγόριθμος στην περίπτωση λανθασμένης πρόβλεψης.

6.2.1 Πολυπλοκότητα Πρόβλεψης

Προκειμένου να αποδείξουμε ότι η πρόβλεψη παρέχει τον ελάχιστο αριθμό πληροφορίας, θα πρέπει πρώτα να βρούμε έναν τρόπο να την μετρήσουμε ποσοτικά. Ο πιο απλός είναι και ο ακόλουθος: ο αριθμός των bits πληροφορίας που απαιτούνται για να εκφραστεί το advice. Σε αυτό το πλαίσιο, θεωρούμε ότι η πρόβλεψη του oracle είναι πάντα σωστή και αναζητάμε έναν ελάχιστο αριθμό από bitsπληροφορίας για να μας οδηγήσει στην βέλτιστη λύση. Το συγκεκριμένο μοντέλο υπολογισμού, advice complexity, περιγράφεται στην Ενότητα 4.1.1.

Θεώρημα 6.2. *Κάθε βέλτιστος αλγόριθμος χρειάζεται τουλάχιστον h bits πληροφορίας, όπου h το ύψος του HST.*

Απόδειξη. Για να δείξουμε ότι ο ελάχιστος αριθμός πληροφορίας είναι πράγματι h, θα πρέπει να κατασκευάσουμε ένα σύνολο στιγμιοτύπων $\mathcal{I}$ το οποίο απαιτεί τόσα bits πληροφορίας για να λυθεί βέλτιστα, από κάθε ντετερμινιστικό αλγόριθμο.

Το σύνολο αυτό, $\mathcal{I}$, αποτελείται από όλα τα διαφορετικά μονοπάτια που σχηματίζονται από την ρίζα του HST στα φύλλα. Επομένως οποιοσδήποτε αλγόριθμος, προκειμένου να λύσει βέλτιστα κάθε στιγμιότυπο, θα πρέπει πρώτα να μπορεί να τα διακρίνει. Για να το καταφέρει απαιτείται διαφορετικό advice string για το καθένα. Τα στιγμιότυπα, ή αλλιώς μονοπάτια, ορίζονται μονοσήμαντα από τα φύλλα στα οποία καταλήγουν. Επομένως η διάκριση μεταξύ των στιγμιότυπων είναι ισοδύναμη με την διάκριση μεταξύ φύλλων.

Αφού το δέντρο είναι δυαδικό με ύψος h, υπάρχουν ακριβώς 2^h φύλλα, άρα $\log_2(2^h) = h$ *bits* αρκούν για να τα κωδικοποιήσουμε και τελικά να λύσουμε το πρόβλημα βέλτιστα. □

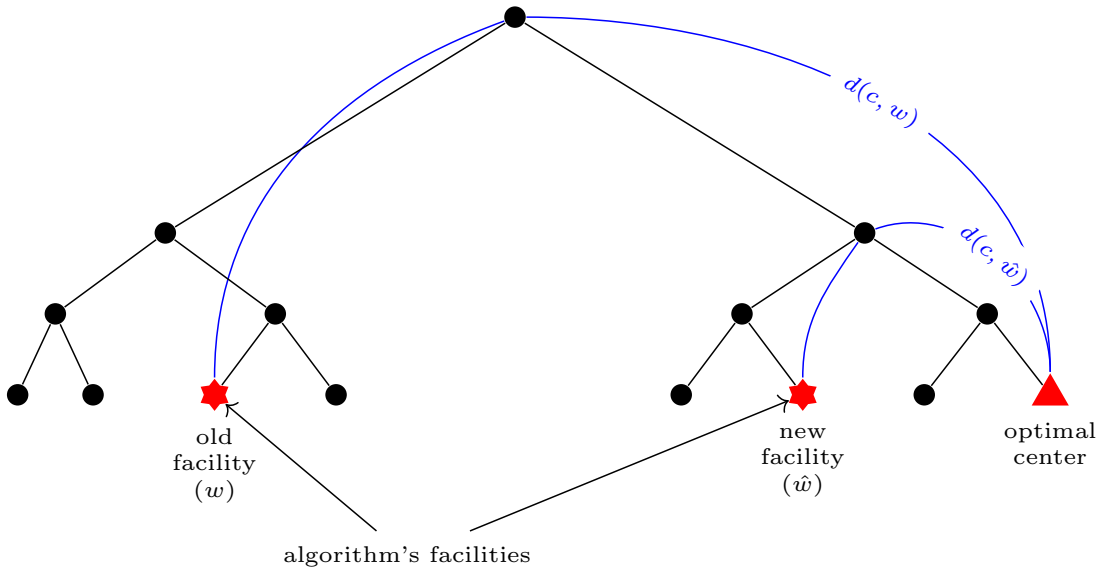
Ο αλγόριθμος 5.5 χρησιμοποιεί ως πρόβλεψη (advice) το μονοπάτι που κατέχει την πλειονότητα (majority) των μελλοντικών αιτημάτων. Από το Συμπέρασμα 6.2 γνωρίζουμε ότι η πληροφορία αυτή, αρκεί για να λυθεί το πρόβλημα βέλτιστα, δηλάδη με $O(1)$ competitive ratio. Επειδή όμως κάθε φύλλο ορίζει μονοσήμαντα ένα μονοπάτι προς την ρίζα του δέντρου, το advice που χρησιμοποιεί ο αλγόριθμος απαιτεί επίσης h bits για να εκφραστεί, άρα τον ελαχιστό δυνατό αριθμό πληροφορίας. Ήτοι, το υποδέντρο που υποδεικνύει κάθε πρόβλεψη κωδικοποιείται με ένα bit πληροφορίας, ***0***-*αριστερά ή* ***1***-*δεξιά*, συνεπώς το μεγαλύτερο μονοπάτι (ρίζα-φύλλο) χρειάζεται το πολύ h bits.

6.2.2 Κάτω Φράγματα Σφάλματος

Στο δεύτερο μέρος μελετάμε την επίπτωση που έχει στο competitive ratio η μεταβολή του σφάλματος που χρεώνεται το oracle για τις λανθασμένες προβλέψεις του. Παρόλο που το σφάλμα του αλγορίθμου 5.5 αποδείχθηκε εξίσου *επαρκές* και *λογικό*,

δεν παύει να είναι αρκετά ισχυρό. Οπότε επιθυμούμε να μελετήσουμε την επίδοση του αλγορίθμου για εναλλακτικές επιλογές. Όλα τα σφάλματα που εξετάζουμε σχετίζονται με την απόσταση των κέντρων που κατασκευάζει ο αλγόριθμος και εκείνου στην βέλτιστη λύση.

Το πλαίσιο στο οποίο δουλεύουμε είναι το ακόλουθο. Η τελευταία πρόβλεψη που έλαβε ο αλγόριθμος οδήγησε στην δημιουργία του κέντρου w. Ωστόσο, μετά την άφιξη των μελλοντικών αιτημάτων παρατηρήθηκε ότι σε κάποιο υποδέντρο του HST η Αρχή 2 παραβιάστηκε ξανά, οπότε ο αλγόριθμος κλήθηκε να κατασκευάσει εκ νέου ένα κέντρο, το $\hat{w}$. Τα σφάλματα που εξετάζουμε αφορούν την ποσότητα που χρεώνει στο oracle ο αλγόριθμος, για την λανθασμένη πρόβλεψη, w. Η απόσταση του παλιού και του νέου κέντρου από το βέλτιστο ορίζεται ως $d(c, w), d(c, \hat{w})$ αντιστοίχως, βλπ Σχήμα 6.3.



Σχήμα 6.3: *Το δέντρο της ανάλυση των κάτω φραγμάτων.*

Σε αυτό το πλαίσιο, εξετάζουμε τέσσερα νέα σφάλματα που κρίνουμε ως πιο λογικά για το πρόβλημα και αποδεικνύουμε την αδυναμία τριών εξ αυτών να πετύχουν σταθερό (constant) competitive ratio. Τα επιχειρήματα μας φυσικά δεν αφορούν έναν συγκεκριμένο αλγόριθμο. Αντιθέτως, ισχύουν σε κάθε αλγόριθμο που χρησιμοποιεί παρόμοιο advice ανεξαρτήτως του τρόπου που διαχειρίζεται την πληροφορία του oracle και των demands.

Ήδη από το Λήμμα 6.2 γνωρίζουμε ότι η σωστή πρόβλεψη οδηγεί σε $O(1)$ competitive ratio. Ο στόχος μας όμως δεν είναι να εξετάσουμε το consistency αλλά το robustness, δηλαδή την συμπεριφορά του αλγορίθμου καθώς το σφάλμα των προβλέψεων αυξάνει.

6.2.2.1 Σφάλμα Μεμονωμένης Πρόβλεψης

Το σφάλμα που εξετάζεται πρώτα, είναι το πιο απλό που μπορεί να υποθέσει κανείς. Συγκεριμένα, χρεώνεται η ποσότητα $d(c, w)$ (είτε $d(c, \hat{w})$) την στιγμή δημιουργίας του νέου κέντρου, $\hat{w}$. Ανεξαρτήτως αλγορίθμου είναι προφανές ότι οδηγούμαστε σε αυθαίρετα μεγαλό competitive ratio. Μια τόσο μικρή ποσότητα είναι αδύνατο να αποζημειώσει το κόστος που χρεώνεται ο αλγόριθμος σε μία φάση.

Πρόταση 6.4. *Κάθε αλγόριθμος με σφάλμα $d(c, w)$ (είτε $d(c, \hat{w})$) έχει competitive ratio $\Omega(\frac{\log n}{\log \log n})$.*

Απόδειξη. Η απόδειξη της πρότασης είναι πράγματι τετριμμένη. Ως παράδειγμα μπορούμε να θεωρήσουμε την περίπτωση όπου η πρόβλεψη υποδεικνύει την θέση του πιο απομακρυσμένου αιτήματος για το νέο κέντρο. Έτσι, ο αλγόριθμος επιβραδύνει την σύγκλιση του στο βέλτιστο κέντρο, ενώ παραλλήλως αποζημιώνεται για το κόστος ενός μόνο αιτήματος. □

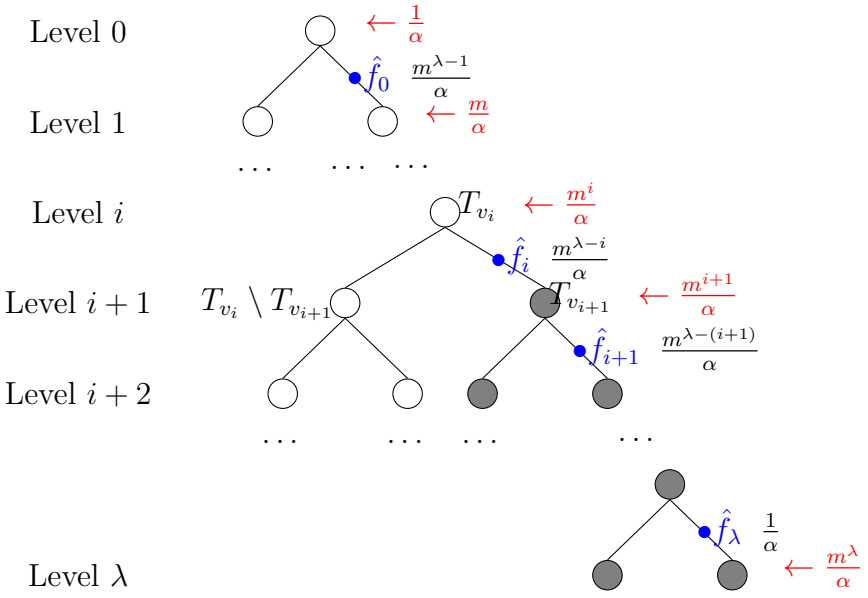
6.2.2.2 Σφάλματα Πολλαπλών Προβλέψεων

Συμπεραίνουμε λοιπόν, ότι το σφάλμα που χρεώνουμε πρέπει να εξυπηρετεί δύο σκοπούς: να περιορίζει την επίδραση ενός μικρού σφάλματος στην πρόβλεψη, και ταυτόχρονα να συσσωρεύει μια μεγαλύτερη ποσότητα σφάλματος επαρκής να αποζημειώσει το επιπλέον κόστος. Ο απλούστερος τρόπος είναι να δεχόμαστε μια νέα πρόβλεψη σε κάθε demand. Έτσι, επιτυγχάνουμε και τους δυο μας στόχους. Από την μία, ένα μικρό λάθος αδυνατεί να επηρεάσει ολοκληρωτικά την τοποθεσία του νέου κέντρου, εφόσον η πλειονότητα των προβλέψεων υποδεικνύει το σωστή τοποθεσία. Από την άλλη, όσο περισσότερες προβλέψεις είναι λανθασμένες τόσο μεγαλύτερο σφάλμα συσσωρεύουμε για να χρεώσουμε στο oracle. Δυστυχώς, ούτε αυτή η επιλογή οδηγεί σε $O(1)$ λύση.

Πρόταση 6.5. *Κάθε αλγόριθμος που δέχεται νέα πρόβλεψη σε κάθε αίτημα έχει competitive ratio τουλάχιστον $\Omega(\frac{\log n}{\log \log n})$.*

Απόδειξη. Η απόδειξη είναι απλή, αρκεί να ανακαλύψουμε το κατάλληλο worst case scenario. Επιλέγουμε λοιπόν ως την θέση των προβλέψεων την θέση των ίδιων των αιτημάτων. Το oracle χρεώνεται για το λάθος του μία ποσότητα ίση με Asg*, δηλαδή συνολικό κόστος σύνδεσης στην βέλτιστη λύση. Αν τώρα θεωρήσουμε το παράδειγμα του κάτω φράγματος των online αλγορίθμων (βλπ Ενότητα 3.2), συμπεραίνουμε ότι λογαριθμικό competitive ratio παραμένει.

Το πιο σημαντικό αποτελέσματα ωστόσο, είναι ότι το φράγμα παραμένει ανεξάρτητα του τρόπου που ο αλγόριθμος διαχειρίζεται τις προβλέψεις. Αν δεν συνέβαινε αυτό τότε αυτόματως θα σήμαινε ότι υπάρχει αλγόριθμος που πετυχαίνει μικρότερο competitive ratio χωρίς την χρήση κάποιου oracle, καθώς η πρόβλεψη ταυτίζεται με την θέση του αιτήματος, άρα η πληροφορία που παρέχεται είναι ήδη γνωστή. □



Σχήμα 6.4: *Κατασκευή HST*

6.2.2.3 Συσχετισμένο Κάτω Φράγμα Σφάλματος

Δυστυχώς στην προηγούμενη πρόταση δεν αναδεικνύεται η μεταβολή του κάτω φράγματος ως προς το συνολικό σφάλμα. Έτσι, στο ακόλουθο θεώρημα παρουσιάζουμε μια διαφορετική κατασκευή που ενσωματώνει το συνολικό σφάλμα στον τελικό τύπο του κάτω φράγματος.

Θεωρούμε την ύπαρξη ενός μόνο κέντρου c^* και ορίζουμε ως $\eta_1 = \sum_x d(x, c^*)$, $\eta_\infty = \max_x d(x, c^*)$ το συνολικό σφάλμα και την μέγιστη απόσταση των προβλέψεων αντίστοιχα. Επίσης ορίζουμε ως $err_1 = \frac{\eta_1}{f}$, $err_\infty = \frac{n\eta_\infty}{f}$.

Θεώρημα 6.3. *Για κάθε τιμή σφάλματος $\eta_1 > 0$, υπάρχει ένα στιγμιότυπο αιτημάτων και προβλέψεων, ώστε κανείς αλγόριθμος να μην έχει competitive ratio καλύτερο από*

$$\Omega\left(\frac{\log(err_\infty)}{\log(err_1^{-1}\,\log(err_\infty))}\right)$$

.

Απόδειξη. Η απόδειξη ακολουθεί παρόμοια κατασκευή με εκείνη του κάτω φράγματος για τους online αλγορίθμους στο [1]. Συγκεκριμένα, ο μετρικός χώρος περιγράφεται από ένα συνεχές [1] δυαδικό HST με $\lambda = m\,\alpha$ επίπεδα, όπου $\alpha = \eta_1/f$. Η απόσταση της ρίζας του δέντρου είναι ίση με $\frac{m^{\lambda-1}}{\alpha}$ και τα μήκη των ακμών μεταξύ κόμβων διαδοχικών επίπεδων μειώνονται κατά m. Έτσι, η απόσταση δύο κόμβων επιπέδου $i, i+1$ ισούται με $m^{(\lambda-1)-i}/\alpha$, βλπ Σχήμα 6.4. □

Παρατηρούμε ότι ικανοποιούνται οι παρακάτω δύο ιδιότητες

[1] Συνεχές λέγεται το δέντρο στο οποίο οι θέσεις των αιτημάτων, προβλέψεων

1 Η απόσταση ενός κόμβου v_i ως προς κάθε κόμβο στο υποδέντρο T_{v_i} είναι το πολύ $\frac{m^{\lambda-i}}{\alpha(m-1)}$.

2 Η απόσταση ενός κόμβου v_i ως προς κάθε κόμβο εκτός του υποδέντρο T_{v_i} είναι τουλάχιστον $m^{\lambda-i+1}/\alpha$.

Κατασκευή ακολουθίας αιτημάτων: Η ακολουθία των αιτημάτων χωρίζεται σε $\lambda+1$ φάσεις. Η Φάση-0 αποτελείται από $1/\alpha$ αιτήματα στην ρίζα του HST. Μετά το τέλος της φάσης $i-1$, αν ο κόμβος v_{i-1} δεν είναι κάποιο φύλλο του δέντρου, ο adversary επιλέγει ομοιομόρφα ένα από τα δύο παιδιά του v_{i-1}, ως τον κόμβο της επόμενης φάσης v_i. Σε αυτόν τον κόμβο θα βρεθούν τα μελλοντικά $\frac{m^i}{\alpha}$ αιτήματα. Αφού ο συνολικός αριθμός των αιτημάτων δεν υπερβαίνει το n, συμπεραίνουμε

$$n = \sum_{i=0}^{\lambda} \frac{m^i}{a} = \frac{1}{\alpha}\frac{m^{\lambda+1}-1}{m-1} \approx m^\lambda/\alpha \tag{6.4}$$

Κατασκευή ακολουθίας προβλέψεων: Αντιστοίχως, η ακολουθία των προβλέψεων χωρίζεται σε $\lambda+1$ φάσεις. Για κάθε αίτημα της φάσης i, ο αλγόριθμος λαμβάνει την πρόβλεψη f_{v_i} με απόσταση $\alpha\, d^*_{v_i}$ από το βέλτιστο κέντρο c^*. Παρατηρούμε ότι καθώς οι απόστασεις μεταξύ διαδοχικών επιπέδων αυξάνονται γραμμικά ως προς το $\frac{1}{\alpha}$. κάθε πρόβλεψη βρίσκεται ενδιάμεσα της ακμής που συνδέει τους v_i, v_{i+1}, βλπ Σχήμα 6.4.

Κόστος βέλτιστης λύσης: Η βέλτιστη λύση ανοιγεί αποκλειστικά ένα κέντρο στην θέση c^* και σε κάθε φάση i χρεώνεται κόστος σύνδεσης το πολύ $\frac{m^i}{\alpha}\cdot\frac{m^{\lambda-i}}{\alpha(m-1)} = \frac{m^\lambda}{\alpha^2(m-1)}$, καθώς $u_h \in T_{v_i}\ \forall\ i \le h$ και χρησιμοποιώντας την Ιδιότητα 1. Επομένως το συνολικό κόστος της βέλτιστης λύσης φράσσεται άνω από $f + \lambda\frac{m^\lambda}{\alpha^2(m-1)} = f + \frac{m^{\lambda-1}}{\alpha(m-1)}$, αντικαθιστώντας $\lambda = m\alpha$. Θέτοντας $f = \frac{m^\lambda}{\alpha}$ καταλήγουμε

$$\mathrm{OPT} \le 2f\frac{m}{m-1}. \tag{6.5}$$

Κόστος κάθε αλγορίθμου: Στο τέλος της φάσης i **κάθε** ντετερμινιστικός αλγόριθμος γνωρίζει ότι το βέλτιστο δέντρο βρίσκεται σε κάποιο φύλλο του T_{v_i}, αλλά δεν μπορεί να διακρίνει σε ποιο από τα δύο υποδέντρα ανήκει το ανήκει το συγκεκριμένο φύλλο. Σταθεροποιούμε την ακολουθία των αιτημάτων μέχρι την φάση i και υπολογίζουμε την αναμενόμενη τιμή του κόστους για τα facilities,demands που δεν ανήκουν στο $T_{v_{i+2}}$, δηλαδή $\mathbb{E}[T_{v_i} \setminus T_{v_{i+2}}]$. Έπειτα διακρίνουμε περιπτώσεις.

- Ο αλγόριθμος δεν έχει facility στο υποδέντρο T_{v_i} κατά την άφιξη των αιτημάτων στον κόμβο v_{i+2}:

- Το κόστος σύνδεσης για τα αιτήματα στους κόμβους $v_i, v_{i+1} \in T_{v_i} \setminus T_{v_{i+2}}$ είναι τουλάχιστον $\frac{m^i}{\alpha}\frac{m^{\lambda-i+1}}{\alpha} + \frac{m^{i+1}}{\alpha}\frac{m^{\lambda-(i+1)+1)}}{\alpha} = 2\frac{m^{\lambda+1}}{\alpha^2} \ge 2\frac{m^\lambda}{\alpha} = 2f$.

- Ο αλγόριθμος έχει ανοικτό facility στο υποδέντρο $T_{v_i} \setminus T_{v_{i+2}}$:

  Ο αλγόριθμος ανοίγει facility στον κόμβο v_{i+1}, καθώς η πρόβλεψη $\hat{f}_{v_i}$ φανερώνει το πραγματικό υποδέντρο $T_{v_{i+1}}$ στο οποίο βρίσκεται το βέλτιστο κέντρο. Σε αυτήν την περίπτωση, ο αλγόριθμος υφίσταται μηδενικό κόστος για τα αιτήματα στον v_{i+1}, ενώ για την περίπτωση των αιτημάτων στον v_i το κόστος σύνδεσης είναι τουλάχιστον $(1-\alpha)\frac{m^\lambda}{\alpha}$. Επομένως, το συνολικό κόστος είναι τουλάχιστον $f + (1-\alpha)\frac{m^\lambda}{\alpha}$.

Ωστόσο με πιθανότητα 1/2 ο adversary επιλέγει τον κόμβο της φάσης $i+2$, ώστε τουλάχιστον ένα από τα facility του αλγορίθμου να μην ανήκει στο υποδέντρο $T_{v_{i+2}}$. Επομένως το κόστος κατασκευής facilities που χρεώνεται ο αλγόριθμος ανά δύο φάσεις είναι ίσο με $f/2$. Λαμβάνοντας υπόψιν τις λ πρώτες φάσεις, καταλήγουμε ότι το κόστος $\geq \frac{\lambda}{4} f$. Συνδυάζοντας επίσης και την Σχέση (6.5) καταλήγουμε σε competitive ratio $\Omega(\lambda) = \Omega(m\frac{\eta_1}{f})$.

Τελικά, μέσω της Σχέσης (6.4) και της ιδιότητας της συγκεκριμένης κατασκευής, $\eta_1/\eta_\infty = m$, λογαριθμούμε και καταλήγουμε στην σχέση

$$m \log m - \frac{\eta_1}{f} \log m = \frac{f}{\eta_1} \log(\frac{\alpha n \eta_\infty}{f}) \tag{6.6}$$

Από την διαδικασία κατασκευής των προβλέψεων ισχύει η σχέση $\eta_1 = Asg^* \approx f$. Έτσι η Σχέση 6.6 μετασχηματίζεται ως $m \log m = \frac{f}{\eta_1} \log(\frac{\alpha n \eta_\infty}{f})$. Έπειτα θέτουμε $m = \exp(W(B))$ όπου $B = \frac{f}{\eta_1} \log(\frac{\eta_\infty n}{f})$ και $W(x)$ η συνάρτηση Lambert W. Χρησιμοποιώντας το κάτω φράγμα της συνάρτησης $W(x) > \log x - \log\log x,\ x > e$ από [64], καταλήγουμε ότι $m = \Theta(\frac{B}{logB})$.

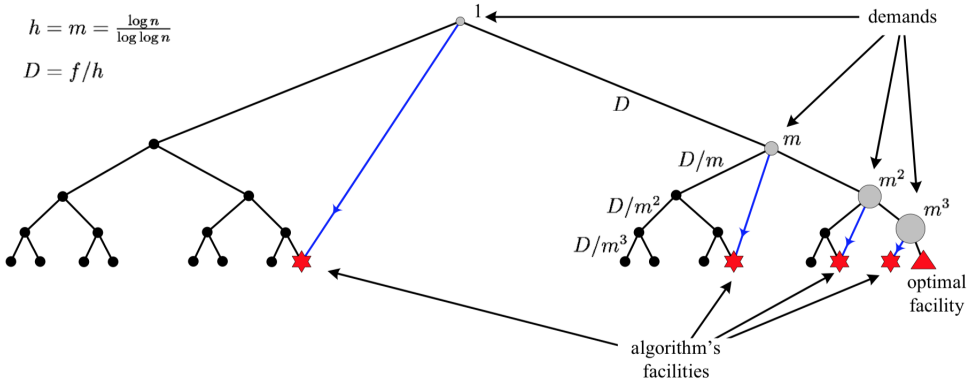
6.2.2.4 Σφάλμα Νέας Πρόβλεψης για Πολλαπλά Αιτήματα

Συμπεραίνουμε λοιπόν ότι απαιτείται κάτι ακόμα ισχύρότερο για να ξεπεραστεί το λογαριθμικό κάτω φράγμα. Στο επόμενο σφάλμα χρεώνουμε την απόσταση του νέου κέντρου $d(c, \hat{w})$ τόσες φορές όσα το πλήθος των unsatisfied demands, αυτών δηλαδή που οδήγησαν στην δημιουργία ενός νέου κέντρου. Η αδυναμία και αυτού του σφάλματος αποδεικνύεται στην συνέχεια.

Για την απόδειξη της Πρότασης 6.6 θα βασιστούμε στο Σχήμα 6.5, που απεικονίζει το HST στο οποίο πραγματοποιείται η απόδειξη του κάτω φράγματος των online αλγορίθμων στο [19] .

Πρόταση 6.6. *Κάθε αλγόριθμος με σφάλμα την ποσότητα $d(c, \hat{w})$ για κάθε unsatisfied demand έχει competitive ratio τουλάχιστον $\Omega(\frac{\log n}{\log\log n})$.*

Απόδειξη. Η απόδειξη ακολουθεί παρόμοια κατασκευή με εκείνη του κάτω φράγματος για τους online αλγορίθμους στο [1]. Στο τέλος της φάσης i κάθε ντετερμινιστικός αλγόριθμος γνωρίζει ότι πρέπει να κατασκευάσει ένα νέο κέντρο σε κάποιο φύλλο



Σχήμα 6.5: *Το HST της απόδειξης του κάτω φράγματος για τους online αλγορίθμους (Πηγή: [1])*

του υποδέντρου T_i. Ωστόσο δεν μπορεί να γνωρίζει σε ποιο από τα δύο υποδέντρα. Δεχόμενος την νέα πρόβλεψη κατασκευάζει το κέντρο στην θέση που του καταλήγει το μονοπάτι που του υπέδειξε το oracle. Αν λοιπόν η πρόβλεψη υποδεικνύει κάθε φορά το λάθος υποδέντρο, το κάτω φράγμα παραμένει. Ωστόσο το κόστος που χρεώνεται το oracle δεν επαρκεί να καλύψει το επιπλέον κόστος του αλγορίθμου.

Συγκεκριμένα, το κόστος *κάθε* ντετερμινιστικού αλγορίθμου είναι τουλάχιστον $\min\{f, m\,D\} = m\,D$ (διότι $f = \frac{D}{h} = \frac{D}{m}$) σε κάθε φάση. Από την άλλη το oracle χρεώνεται σφάλμα το πολύ $2\,D$. Συνεπώς το συνολικό κόστος του αλγορίθμου είναι τουλάχιστον $(h+1)\,(m-2)\,D$, ενώ της βέλτιστης λύσης είναι το πολύ $f + h\,D\frac{m}{m-1} = m\,D(1+\frac{m}{m-1})$.

$$\begin{aligned}\frac{\text{ALG} - err}{\text{OPT}} &\geq (h+1)\,\frac{(m-2)\,D}{m\,D\,(1+\frac{m}{m-1})} = (h+1)\,\frac{(m-2)\,(m-1)}{m\,(2m-1)} \\ &\geq (h+1)\,\frac{(m-2)\,(m-1)}{m\,(2m-2)} = (h+1)\,\frac{(m-2)\,(m-1)}{2\,m\,(m-1)} \\ &= (h+1)\,\frac{m-2}{2\,m} \geq \frac{h+1}{2} \geq \frac{h}{2} = \Omega(\frac{\log n}{\log \log n}) \end{aligned} \tag{6.7}$$

Ο λόγος που το κάτω φράγμα παραμένει οφείλεται στο γεγονός ότι το νέο κέντρο έχει βρεθεί τουλάχιστον ένα επίπεδο πιο κοντά στο βέλτιστο κέντρο. Επομένως το κόστος που χρεώνεται στο oracle είναι, κατά προσέγγιση, m φορές μικρότερο από εκείνο που χρεώθηκε ο αλγόριθμος. □

6.2.2.5 Σφάλμα Παλιάς Πρόβλεψης για Πολλαπλά Αιτήματα

Από την προηγούμενη απόδειξη συμπεραίνουμε ότι το πρόβλημα έγκειται στον συντελεστή m. Αυτό οφείλεται στην διαφορά μεταξύ της απόστασης των δύο κέντρων, $w, \hat{w}$ από το βέλτιστο, βλ Σχήμα 6.3. Έτσι καταλαβαίνουμε ότι το σφάλμα πρέπει

να χρεώνει μία πόσοτητα της τάξης της του $d(c, w)$ σε κάθε unsatisfied demands Το *τέταρτο σφάλμα* χρεώνει αυτήν ακριβώς την ποσότητα. Είναι προφανές ότι αυτή η περίπτωση οδηγεί σε $O(1)$ competitive λύση. Ολοκληρώνουμε με την τελευταία πρόταση.

Πρόταση 6.7. *Κάθε βέλτιστος αλγόριθμος χρεώνει σφάλμα ίσο με $d(c, w)$ για κάθε unsatisfied demand.*

Εύκολα αποδεικνύεται ότι το συγκεκριμένο σφάλμα είναι ισοδύναμο, κατά ένα σταθερό παράγοντα, με εκείνο που επιλέξαμε αρχικά.

Κεφάλαιο 7

Συμπεράσματα

Συνοψίζοντας, μελετήσαμε για πρώτη φορά την learning augmented (υποβοηθούμενη από μηχανική μάθηση) εκδοχή του προβλήματος online facility location. Διατυπώσαμε έναν ντετερμινιστικό αλγόριθμο με σταθερό competitive ratio στον μετρικό χώρου του Hierarchical Well-Separated Tree με ομοιόμορφα (uniform) κόστη κατασκευής νέων κέντρων. Ακόμη, δείξαμε τα απαραίτητα κάτω φράγματα τόσο για την πληροφορία του oracle όσο και για το σφάλμα με το οποίο χρεώνεται στην περίπτωση λάθους. Εν ολίγοις, αποδείξαμε την βελτιστότητα του αλγορίθμου ως προς κάθε δυνατό συνδυασμό *πρόβλεψης* και *σφάλματος*.

Υπάρχουν αρκετές ενδιαφέρουσες κατευθύνσεις για μελλοντική έρευνα και μελέτη της συγκεκριμένης εκδοχής. Οι πιο βασικές αφορούν εξ ολοκλήρου τον αλγόριθμο που διατυπώσαμε και τους περιορισμούς που επιβάλλαμε στην δομή του προβλήματος. Παραδείγματος χάριν, έχει ιδιαίτερη σημασία να μελετήσει κανείς την επέκταση του αλγορίθμου σε γενικούς μετρικούς χώρους και να ερευνήσει τους περιορισμούς που ενδεχομένως να προκύπτουν. Ακόμη, στην περίπτωση των πολλαπλών (βέλτιστων) κέντρων έχει αρκετό ενδιαφέρον η αλληλεπίδραση μεταξύ των νέων κέντρων που κατασκευάζει ο αλγόριθμος, επηρεασμένος βέβαια από την συμβουλή του oracle, και εκείνων στην βέλτιστη λύση. Όπως γνωρίζουμε από την απλή online εκδοχή [19], η περίπτωση των πολλαπλών κέντρων δεν έχει *πάντα* τετριμμένη ανάλυση. Τέλος, παρά την φαινομενική ευκολία του, θα μπορούσε κανείς να εξετάσει την περίπτωση μη ομοιόμορφου κόστους κατασκευής νέων κέντρων.

Παραρτήματα

Παράρτημα Α΄

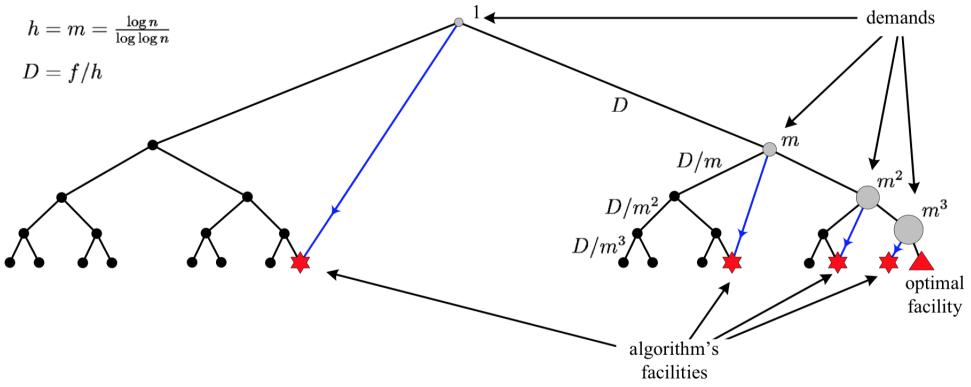
Παράρτημα Α

Λήμμα 1.3. *Αν οι προβλέψεις του oracle αφορούν το επίπεδο k τότε το competitive ratio είναι τουλάχιστον $\Omega(\frac{\log n}{\log\log n} - i - 2)$, όπου i η απόσταση μεταξύ του επιπέδου k και h.*

Απόδειξη. Η απόδειξη πραγματοποιείται σε μία κατασκευή παρόμοια με εκείνη που χρησιμοποιεί στο [19],[1] για την απόδειξη του κάτω φράγματος. Η ιδέα φυσικά είναι η ίδια. Όλοι οι αλγόριθμοι για το Online Facility Location, βλπ [18], που δεν κάνουν χρήση κάποιου oracle διαθέτουν ως **μοναδική** πηγή πληροφορίας για την θέση του βέλτιστου κέντρου, την θέση των ίδιων των αιτημάτων. Εν μέρει, τα αιτήματα καθορίζουν την επίδοση του. Όσο πιο γρήγορα συγκλίνουν προς το βέλτιστο κέντρο τόσο πιο κοντά κατασκευάζεται κάθε νέο κέντρο. Προκειμένου να καθυστερήσουμε την εν λόγω σύγκλιση, και κατ᾽ επέκταση να αυξήσουμε το competitive ratio, αρκεί να διατάξουμε την σειρά την αιτημάτων ως προς την απόσταση τους από το κέντρο. Έτσι, πρώτα θα εμφανιστούν εκείνα με απόσταση λ, μετά εκείνα με απόσταση λ/k κ.ο.κ.

Στην δική μας περίπτωση, πέρα από την θέση των αιτημάτων, ο αλγόριθμος δέχεται προβλέψεις, άρα επιπλέον πληροφορία, σχετικά με το πλειονότητα των μελλοντικών αιτημάτων σε διάφορα σημεία του δέντρου. Αποδεικνύουμε ότι παρά την πληροφορία των προβλέψεων το κάτω φραγμα μπορεί να μειωθεί το πολύ κατά ένα σταθερό όρο. Διαισθητικά, η ερμηνεία του αποτελέσματος είναι η εξής. Η πληροφορία των προβλέψεων δεν επαρκεί για να κάνει τον αλγόριθμο να φτάσει γρηγορότερα στον κόμβο του επιπέδου k, ο οποίος ανήκει στο μονοπάτι του βέλτιστου κέντρου, απ΄ ότι αν χρησιμοποιούσε μονάχα την πληροφορία που διαθέτει από την θέση των αιτημάτων. Στην συνέχεια υποθέτουμε χωρίς βλάβη της γενικότητας ότι το βέλτιστο κέντρο βρίσκεται στο δεξί φύλλο του δέντρου, βλπ Σχήμα Α΄.1.

Η διαδικασία κατασκευής είναι η ακόλουθη. Ξεκινάμε αφαιρώντας τα αιτήματα των k τελευταίων επιπέδων. Ο αριθμός των αιτήματα που παραμένουν πλέον στο δέντρο θα είναι πλέον $\frac{m^{k-1+1}-1}{m-1} = \frac{m^k-1}{m-1}$. Πρώτα, χωρίζουμε τους κόμβους του επιπέδου k στην μέση και επιλέγουμε έναν κόμβο ως αντιπρόσωπο για τους πρώτους μισούς. Έπειτα, χωρίζουμε το δέυτερο μισό πάλι στην μέση και επιλέγουμε έναν αντιπρόσωπο για το πρώτο μισό. Συνεχίζοντας αναδρομικά, καταλήγουμε με k αντιπροσώπους με



Σχήμα Α΄.1: *Το HST της απόδειξης του κάτω φράγματος για τους online αλγορίθμους (Πηγή: [1])*

τον τελευταίο δεξιά να συμπίπτει με τον $k-$οστό κόμβο του μονοπατιού από την ρίζα στο βέλτιστο κέντρο.

$$\alpha_1, \alpha_2, \cdots, \alpha_k$$

Αρκεί να δείξουμε ότι ανεξαρτήτως της ορθότητας των προβλέψεων το κάτω φράγμα συνεχίζει να ισχύει. Συνεπώς θα θεωρησούμε ότι το oracle δεν κάνει ποτέ λάθος και θα κατασκευάσουμε μία ακολουθία που θα αναγκάσει τον αλγόριθμο να συγκλίνει στο α_k με τον ίδιο ρυθμό αν χρησιμοποιούσε μόνο την πληροφορία από τις θέσεις των αιτημάτων. Προκειμένου να το πετύχουμε αρκεί να τοποθετήσουμε k νεά αιτήματα στους κόμβους α_k. Συγκεκριμένα, k αιτήματα τοποθετούνται στο α_1, $k-1$ στον α_2 κ.ο.κ.

Σε κάθε φάση i, ο αλγόριθμος αφού λάβει την πρόβλεψη για το επίπεδο k θα κατασκευάζει ένα νέο κέντρο σε κάποιο φύλλο του υποδέντρου T_{α_i}. Με αυτόν τον τρόπο κερδίσει μερικώς ένα μέρος από το κόστος σύνδεσης των αιτημάτων στον α_p ($\leq 2f$) αλλά θα χρεώνεται με κόστος τουλάχιστον $\min\{f, m\,D\}$ σε κάθε φάση $p \in \{1, \cdots, k\}$. Έτσι το συνολικό κόστος, παρόμοια με το [1], φράσσεται από κάτω ως εξής:

$$\mathrm{C[ALG]} \geq k \, \min\{f, m\,D\} \tag{Α΄.1}$$

Η βέλτιστη λύση αναγκαστικά χρεώνεται επιπλέον κόστος για τα νέα αιτήματα στους κόμβους α_i.

$$
\begin{aligned}
\mathrm{C}[\mathrm{OPT}] &= \sum_{i=0}^{k-1} L_i m^i + \sum_{i=0}^{k-2} (2L_i - L_k)\,(k-i) \\
&\leq k \cdot D\frac{m}{m-1} + \sum_{i=0}^{k-2} (2L_i)\,k \\
&\leq k\,D\frac{m}{m-1} + 2k\,D\frac{m}{m-1}\frac{m}{m-1}
\end{aligned}
\tag{A′.2}
$$

Το k εκφρασμένο ως προς την απόσταση του επιπέδου των πρόβλεψεων και των φύλλων του δέντρο ισούται με $k = h-i$. Θέτοντας $D = \frac{f}{m}$, $k = h-i$ και $m = h-i-1$ στις σχέσεις A′.1, A′.2.

$$
\begin{aligned}
\mathrm{C}[\mathrm{ALG}] &\geq (h-i)\,f \\
\mathrm{C}[\mathrm{OPT}] &\leq (h-i)\frac{f}{m-1} + 2(h-i)\frac{m\,f}{(m-1)^2} \\
&\leq (h-i)\,\frac{f}{h-i-2} + 2\,(h-i)\,\frac{(h-i-1)\,f}{(h-i-2)^2} \\
\frac{\mathrm{C}[\mathrm{ALG}]}{\mathrm{C}[\mathrm{OPT}]} &\geq (h-i-2)
\end{aligned}
$$

Ο μόνος περιορισμός που έχουμε είναι το πλήθος των αιτημάτων να μην υπερβαίνει το n. Όμως στην περίπτωση μας το πλήθος των αιτημάων είναι περίπου $(h-i-1)^{h-i-1}$, όπου για $h = \frac{\log n}{\log\log n}$ το $(h-i-1)^{h-i-1}$ φράσετται από το h^h, το οποίο με την σειρά του φράσσεται από το n, βλπ [1].

□

Βιβλιογραφία

[1] Dimitris Fotakis. *On the Competitive Ratio for Online Facility Location. Algorithmica*, 50(1):1–57, 2008.

[2] Zvi Drezner και Horst W. Hamacher. *Facility location - applications and theory.* Springer, 2002.

[3] Andreas Klose και Andreas Drexl. *Facility location models for distribution system design. Eur. J. Oper. Res.*, 162(1):4–29, 2005.

[4] Christian Frank και Kay Römer. *Distributed Facility Location Algorithms for Flexible Configuration of Wireless Sensor Networks. Distributed Computing in Sensor Systems, Third IEEE International Conference, DCOSS 2007, Santa Fe, NM, USA, June 18-20, 2007, Proceedings*James Aspnes, Christian Scheideler, Anish Arora και Samuel Madden, επιμελητές, τόμος 4549 στο *Lecture Notes in Computer Science*, σελίδες 124–141. Springer, 2007.

[5] Nevena Lazic, Inmar E. Givoni, Brendan J. Frey και Parham Aarabi. *FLoSS: Facility location for subspace segmentation. IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, σελίδες 825–832. IEEE Computer Society, 2009.

[6] Hongdong Li. *Two-View Motion Segmentation from Linear Programming Relaxation. 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. IEEE Computer Society, 2007.

[7] David B. Shmoys, Éva Tardos και Karen Aardal. *Approximation Algorithms for Facility Location Problems (Extended Abstract). Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*Frank Thomson Leighton και Peter W. Shor, επιμελητές, σελίδες 265–274. ACM, 1997.

[8] Moses Charikar και Sudipto Guha. *Improved Combinatorial Algorithms for the Facility Location and k-Median Problems. 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, σελίδες 378–388. IEEE Computer Society, 1999.

[9] Maxim Sviridenko. *An Improved Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. Integer Programming and Combinatorial Optimization, 9th International IPCO Conference, Cambridge, MA, USA, May 27-29, 2002, Proceedings*William J. Cook και Andreas S. Schulz, επιμελητές, τόμος 2337 στο *Lecture Notes in Computer Science*, σελίδες 240–257. Springer, 2002.

[10] Jaroslaw Byrka και Karen Aardal. *An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. SIAM J. Comput.*, 39(6):2212–2231, 2010.

[11] Sudipto Guha και Samir Khuller. *Greedy Strikes Back: Improved Facility Location Algorithms. Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*Howard J. Karloff, επιμελητής, σελίδες 649–657. ACM/SIAM, 1998.

[12] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala και Vinayaka Pandit. *Local Search Heuristics for k-Median and Facility Location Problems. SIAM J. Comput.*, 33(3):544–562, 2004.

[13] Kamal Jain και Vijay V. Vazirani. *Approximation algorithms for metric facility location and* k*-Median problems using the primal-dual schema and Lagrangian relaxation. J. ACM*, 48(2):274–296, 2001.

[14] Sudipto Guha και Samir Khuller. *Greedy Strikes Back: Improved Facility Location Algorithms. Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, σελίδα 649–657, USA, 1998. Society for Industrial and Applied Mathematics.

[15] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi και Vijay V. Vazirani. *Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. J. ACM*, 50(6):795–824, 2003.

[16] Kamal Jain, Mohammad Mahdian και Amin Saberi. *A new greedy approach for facility location problems. Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*John H. Reif, επιμελητής, σελίδες 731–740. ACM, 2002.

[17] Adam Meyerson. *Online Facility Location. 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, σελίδες 426–431. IEEE Computer Society, 2001.

[18] Dimitris Fotakis. *Online and incremental algorithms for facility location. SIGACT News*, 42(1):97–131, 2011.

[19] Dimitris Fotakis. *On the Competitive Ratio for Online Facility Location. Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow και Gerhard J. Woeginger, επιμελητές, τόμος 2719 στο *Lecture Notes in Computer Science*, σελίδες 637–652. Springer, 2003.

[20] Aris Anagnostopoulos, Russell Bent, Eli Upfal και Pascal Van Hentenryck. *A simple and deterministic competitive algorithm for online facility location. Inf. Comput.*, 194(2):175–202, 2004.

[21] Daniel D. Sleator και Robert E. Tarjan. *Amortized Efficiency of List Update and Paging Rules. Commun. ACM*, 28(2):202–208, 1985.

[22] Thodoris Lykouris και Sergei Vassilvitskii. *Competitive Caching with Machine Learned Advice. Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*Jennifer G. Dy και Andreas Krause, επιμελητές, τόμος 80 στο *Proceedings of Machine Learning Research*, σελίδες 3302–3311. PMLR, 2018.

[23] Manish Purohit, Zoya Svitkina και Ravi Kumar. *Improving Online Algorithms via ML Predictions. Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi και Roman Garnett, επιμελητές, σελίδες 9684–9693, 2018.

[24] Yair Bartal. *Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications. 37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, σελίδες 184–193. IEEE Computer Society, 1996.

[25] Leonid G Khachiyan. *Polynomial algorithms in linear programming. USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[26] Narendra Karmarkar. *A new polynomial-time algorithm for linear programming. Proceedings of the sixteenth annual ACM symposium on Theory of computing*, σελίδες 302–311, 1984.

[27] Howard Karloff. *Linear Programming*. Birkhäuser Basel, 1στ εδ. 1991. 2νδ πριντινγη έκδοση, 2008.

[28] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos και Avi Wigderson. *On the Power of Randomization in Online Algorithms (Extended*

*Abstract). Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*Harriet Ortiz, επιμελητής, σελίδες 379–386. ACM, 1990.

[29] A. Borodin, N. Linial και M. Saks. *An Optimal Online Algorithm for Metrical Task Systems. Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, σελίδα 373–382, New York, NY, USA, 1987. Association for Computing Machinery.

[30] P. Raghavan και M. Snir. *Memory versus Randomization in On-Line Algorithms. IBM J. Res. Dev.*, 38(6):683–707, 1994.

[31] A. C. Yao. *Probabilistic computations: Toward a unified measure of complexity. 18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, σελίδες 222–227, 1977.

[32] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator και Neal E. Young. *Competitive Paging Algorithms. CoRR*, ςς.ΔΣ/0205038, 2002.

[33] Sandy Irani, Nick Reingold, Jeffery R. Westbrook και Daniel Dominic Sleator. *Randomized Competitive Algorithms for the List Update Problem. Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California, USA*Alok Aggarwal, επιμελητής, σελίδες 251–260. ACM/SIAM, 1991.

[34] Susanne Albers. *Improved Randomized On-Line Algorithms for the List Update Problem. Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1995. San Francisco, California, USA*Kenneth L. Clarkson, επιμελητής, σελίδες 412–419. ACM/SIAM, 1995.

[35] Christoph Ambühl, Bernd Gärtner και Bernhardvon Stengel. *A new lower bound for the list update problem in the partial cost model. Theor. Comput. Sci.*, 268(1):3–16, 2001.

[36] Mark S. Manasse, Lyle A. McGeoch και Daniel Dominic Sleator. *Competitive Algorithms for On-line Problems. Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*Janos Simon, επιμελητής, σελίδες 322–333. ACM, 1988.

[37] Y. Bartal, B. Bollobas και M. Mendel. *A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, σελίδες 396–405, 2001.

[38] Amos Fiat και Manor Mendel. *Better algorithms for unfair metrical task systems and applications. Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*F. Frances Yao και Eugene M. Luks, επιμελητές, σελίδες 725–734. ACM, 2000.

[39] Allan Borodin και Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.

[40] Susanne Albers. *Online algorithms: a survey. Math. Program.*, 97(1-2):3–26, 2003.

[41] Harry Lang. *Online Facility Location against a* t*-Bounded Adversary. Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*Artur Czumaj, επιμελητής, σελίδες 1002–1014. SIAM, 2018.

[42] Dimitris Fotakis. *A primal-dual algorithm for online non-uniform facility location. J. Discrete Algorithms*, 5(1):141–148, 2007.

[43] Niv Buchbinder και Joseph Naor. *The Design of Competitive Online Algorithms via a Primal-Dual Approach. Found. Trends Theor. Comput. Sci.*, 3(2-3):93–263, 2009.

[44] Mário César San Felice, Sin-Shuen Cheung, Orlando Lee και David P. Williamson. *The Online Prize-Collecting Facility Location Problem. Electron. Notes Discret. Math.*, 50:151–156, 2015.

[45] Stefan Dobrev, Rastislav Královič και Dana Pardubská. *Measuring the problem-relevant information in input. RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications*, 43(3):585–613, 2009.

[46] Hans Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič και Tobias Mömke. *On the advice complexity of online problems. International Symposium on Algorithms and Computation*, σελίδες 331–340. Springer, 2009.

[47] Yuval Emek, Pierre Fraigniaud, Amos Korman και Adi Rosén. *Online computation with advice. Theoretical Computer Science*, 412(24):2642–2656, 2011.

[48] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen και Jesper W. Mikkelsen. *Online Algorithms with Advice: A Survey. SIGACT News*, 47(3):93–129, 2016.

[49] Dennis Komm. *An Introduction to Online Computation - Determinism, Randomization, Advice.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016.

[50] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali και Marc P. Renault. *Online Computation with Untrusted Advice. 11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*Thomas Vidick, επιμελητής, τόμος 151 στο *LIPIcs*, σελίδες 52:1–52:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[51] Vahab S. Mirrokni, Shayan Oveis Gharan και Morteza Zadimoghaddam. *Simultaneous approximations for adversarial and stochastic online budgeted allocation. Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*Yuval Rabani, επιμελητής, σελίδες 1690–1701. SIAM, 2012.

[52] Sébastien Bubeck και Aleksandrs Slivkins. *The Best of Both Worlds: Stochastic and Adversarial Bandits. COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*Shie Mannor, Nathan Srebro και Robert C. Williamson, επιμελητές, τόμος 23 στο *JMLR Proceedings*, σελίδες 42.1–42.23. JMLR.org, 2012.

[53] Sreenivas Gollapudi και Debmalya Panigrahi. *Online Algorithms for Rent-Or-Buy with Expert Advice. Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*Kamalika Chaudhuri και Ruslan Salakhutdinov, επιμελητές, τόμος 97 στο *Proceedings of Machine Learning Research*, σελίδες 2319–2327. PMLR, 2019.

[54] Dhruv Rohatgi. *Near-optimal bounds for online caching with machine learned advice. Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, σελίδες 1834–1845. SIAM, 2020.

[55] Alexander Wei. *Better and Simpler Learning-Augmented Online Caching. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*Jaroslaw Byrka και Raghu Meka, επιμελητές, τόμος 176 στο *LIPIcs*, σελίδες 60:1–60:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[56] Andres Muñoz Medina και Sergei Vassilvitskii. *Revenue Optimization with Approximate Bid Predictions. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*Isabelle Guyon, Ulrikevon Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan και Roman Garnett, επιμελητές, σελίδες 1858–1866, 2017.

[57] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley και Sergei Vassilvitskii. *Online Scheduling via Learned Weights. Proceedings of the 2020 ACM-SIAM*

*Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*Shuchi Chawla, επιμελητής, σελίδες 1859–1877. SIAM, 2020.

[58] Michael Mitzenmacher. *Scheduling with Predictions and the Price of Misprediction. 11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*Thomas Vidick, επιμελητής, τόμος 151 στο *LIPIcs*, σελίδες 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[59] Étienne Bamas, Andreas Maggiori και Ola Svensson. *The Primal-Dual method for Learning Augmented Algorithms. Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan και Hsuan-Tien Lin, επιμελητές, 2020.

[60] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean και Neoklis Polyzotis. *The Case for Learned Index Structures. Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*Gautam Das, Christopher M. Jermaine και Philip A. Bernstein, επιμελητές, σελίδες 489–504. ACM, 2018.

[61] Michael Mitzenmacher. *A Model for Learned Bloom Filters and Optimizing by Sandwiching. Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi και Roman Garnett, επιμελητές, σελίδες 462–471, 2018.

[62] Chen-Yu Hsu, Piotr Indyk, Dina Katabi και Ali Vakilian. *Learning-Based Frequency Estimation Algorithms. 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[63] Yihe Dong, Piotr Indyk, Ilya P. Razenshteyn και Tal Wagner. *Learning Space Partitions for Nearest Neighbor Search. 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[64] Abdolhossein Hoorfar και Mehdi Hassani. *Inequalities on the Lambert W function and hyperpower function. J. Inequal. Pure and Appl. Math.*

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
FL	Facility Location
lca	lowest common ancestor
wcs	worst case scenario
la	learning augmented
HST	Hierarchical Well Separated Tree
OFL	Online Facility Location
LaOFL	Learning augmented Online Facility Location

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
άμεσος αλγόριθμος, αλγόριθμος άμεσης απόκρισης	online algorithm
πρόβλημα χωροθέτησης εγκαταστάσεων	facility location problem
άμεσο πρόβλημα χωροθέτησης	online facility location problem
κόστος σύνδεσης	assignment cost
κόστος κατασκευής	facility cost
γραμμικός προγραμματισμός	linear programming
δυικό πρόβλημα	dual problem
μετρικός χώρος	metric space
λεαστ ϲομμον ανϲεϲτορ	κατώτατος κοινός πρόγονος
δυαδικό	binary
δεδομένα εισόδου	input
εγκατάσταση,κέντρο	facility
αίτημα	demand
στιγμιότυπο προβλήματος	instance
πιθανοτικός αλγόριθμος	randomized algorithm
ντετερμινιστικός αλγόριθμος	deterministic algorithm
λόγος ανταγωνιστικότητας	competitive ratio
κ-ανταγωνιστικός	k-competitive
αντίπαλος	adversary
κρυφή μνήμη	cache
πρόβλημα σελιδοποίησης	paging
ομοιόμορφος	uniform
συστάδα	cluster
μαντείο προβλέψεων	oracle,predictor
συμβουλή	advice
πολυπλοκότητα	complexity
υποβοηθούμενος από μηχανική μάθηση	learning augmented