ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
CoReLab

Αποδοτική Χωροθέτηση Σημείων Συγκέντρωσης σε Δίκτυα Ιδιοτελούς Δρομολόγησης

Facility Location in Selfish Routing Networks

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Μάριου Ν. Μερτζανίδη
Marios N. Mertzanidis

Επιβλέπων: Δημήτριος Φωτάκης
Αν. Καθηγητής ΕΜΠ

Συνεπιβλέπων: Θανάσης Λιανέας
Λέκτορας ΕΜΠ

Αθήνα, Νοέμβριος 2021

Εθνικο Μετσοβιο Πολυτεχνειο
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών

Αποδοτική Χωροθέτηση Σημείων Συγκέντρωσης σε Δίκτυα Ιδιοτελούς Δρομολόγησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Μάριου Ν. Μερτζανίδη

Επιβλέπων: Δημήτριος Φωτάκης
Αν. Καθηγητής ΕΜΠ

Συνεπιβλέπων: Θανάσης Λιανέας
Λέκτορας ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11 Νοεμβρίου 2021.

.....	.....	.....
Δημήτριος Φωτάκης	Αριστείδης Παγουρτζής	Ευάγγελος Μαρκάκης
Αν. Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Αν. Καθηγητής ΟΠΑ

Αθήνα, Νοέμβριος 2021.

.....

Μάριος Ν. Μερτζανίδης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σε αυτή την διπλωματική θα μας απασχολήσει ένα καινούργιο πρόβλημα που συνδυάζει τους τομείς της Χωροθέτησης Εγκαταστάσεων και των Παιγνίων Συμφόρησης. Ονομάζουμε το πρόβλημα Χωροθέτηση Εγκαταστάσεων για Ιδιοτελής Χρήστες και το εξετάζουμε τόσο από την σκοπιά της βελτιστοποίησης όσο και από την σκοπιά της Θεωρίας Παιγνίων. Έχοντας εξετάσει την υπάρχουσα διεθνή βιβλιογραφία πάνω στα προβλήματα Χωροθέτησης Εγκαταστάσεων, Σχεδίασης Δικτύων και Παιγνίων Συμφόρησης, εφαρμόζουμε την γνώση αυτή πάνω σε διαφορετικές εκδοχές του προβλήματος μας. Η συνεισφορά της παρούσας εργασίας είναι πολυεπίπεδη. Μέσω αντιπαραδειγμάτων δείχνουμε πως και γιατί οι γνωστές τεχνικές τοπικής αναζήτησης και γραμμικού προγραμματισμού αποτυγχάνουν να μας δώσουν καλές προσεγγιστικές λύσεις. Μελετώντας την δομή της βέλτιστης λύσης περιορίζουμε τον χώρο αναζήτησης βέλτιστων λύσεων. Παρουσιάζουμε έναν αλγόριθμο με λογαριθμικό λόγο προσέγγισης για μια εκδοχή του γενικού προβλήματος. Στρεφόμενοι στην παιγνιοθεωρητική όψη, βρίσκουμε προσεγγιστικές ισορροπίες και φράσσουμε το τίμημα της αναρχίας. Τέλος, εφαρμόζοντας τεχνικές αραίωσης δίνουμε μια προσεγγιστική λύση για μια απλή εκδοχή του προβλήματος δείχνοντας παράλληλα γιατί η εφαρμογή τέτοιων τεχνικών αποτυγχάνει στο γενικό πρόβλημα.

Λέξεις Κλειδιά: Χωροθέτηση Εγκαταστάσεων, Σχεδίαση Δικτύων, Παίγνια Συμφόρησης, Προσεγγιστικοί Αλγόριθμοι, Τοπική Αναζήτηση, Γραμμικός Προγραμματισμός, Πιθανοτικοί Αλγόριθμοι, Τεχνικές Αραίωσης, Προσεγγιστική Ισορροπία Νας, Τίμημα της Αναρχίας.

Abstract

In this thesis, we are preoccupied with solving a novel problem that combines the well-studied fields of Facility Location and Congestion Games. We call this problem Facility Location for Selfish Commuters and we examine it both from an optimization and a game-theoretic point of view. After examining the existing techniques for solving, facility location, network design and congestion game problems we provide insights and algorithms for different variants of the general problem. Our contribution in this thesis is manifold. Using counter-examples we show how and why the established techniques of local search and linear programming fail in giving us good approximation algorithms. By examining the structure of the optimal solution, we restrict the search space of optimal solutions. Also, we present a logarithmic approximation algorithm for a variant of the general problem. From a game-theoretic standpoint we find approximate Nash equilibria and we bound the Price of Anarchy. Finally, by applying sparsification techniques, we propose an approximate solution to a simpler variant of the problem, while at the same time we demonstrate why the use of such techniques fails in the more general setting.

Key Words: Facility Location, Network Design, Congestion Games, Approximate Algorithms, Local Search, Linear Programming, Probabilistic Algorithms, Sparsification Techniques, Approximate Nash Equilibria, Price of Anarchy

Ευχαριστίες

Καταρχάς θα ήθελα να ευχαριστήσω τον κ. Φωτάκη. Η διορατικότητα και η εμπειρία του ήταν καθοριστικοί παράγοντες για την επιτυχή ολοκλήρωση της διπλωματικής αυτής. Θα ήθελα επίσης να τον ευχαριστήσω θερμά για την βοήθεια και την καθοδήγηση του σε αυτά τα πρώτα βήματα της ερευνητικής μου καριέρας.

Επίσης θα ήθελα να ευχαριστήσω πάρα πολύ τον Θανάση Λιανέα. Η υπομονή του να ακούει και να αναλύει κάθε εβδομάδα ότι μπορεί να είχα σκεφτεί και οι στοχευμένες του παρατηρήσεις έχουν παίξει αδιαμφισβήτητα καθοριστικό ρόλο στον όγκο και την ποιότητα της διπλωματικής αυτής.

Θα ήθελα να ευχαριστήσω τους Γονείς και τους Συγγενείς μου που ήταν εκεί να με στηρίζουν σε κάθε βήμα και επιλογή μου. Είναι δεδομένο ότι χωρίς αυτούς δεν θα είχα φτάσει ως εδώ και δεν θα είχα τους υψηλούς στόχους για το μέλλον που έχω τώρα.

Τέλος θα ήθελα να ευχαριστήσω τους φίλους μου οι οποίοι είναι μία πηγή δύναμης που με εμπνέει και με παρακινεί να εξελίσσομαι διαρκώς. Χωρίς αυτούς δεν θα ήμουν το άτομο που είμαι σήμερα και δεν θα ήμουν τόσο γεμάτος από χαρούμενες εμπειρίες και αναμνήσεις ζωής.

Σας ευχαριστώ.

Contents

Κεφάλαιο 1

Εκτεταμένη Ελληνική Περίληψη

1.1 Εισαγωγή

Ένας από τους πιο ενδελεχώς ερευνημένους τομείς της θεωρητικής επιστήμης των υπολογιστών είναι τα προβλήματα χωροθέτησης εγκαταστάσεων. Στα προβλήματα αυτά, συνήθως, ο βασικός στόχος είναι το άνοιγμα ενός συνόλου εγκαταστάσεων με τέτοιο τρόπο ώστε να ελαχιστοποιείται το κόστος τον εγκαταστάσεων αυτών και το κόστος της μεταφοράς πελατών στις εγκαταστάσεις αυτές. Ανάλογα με τον τρόπο που υπολογίζονται αυτά τα κόστη και τους πιθανούς περιορισμούς που μπορεί να επιβάλλονται στις αποδεκτές λύσεις του προβλήματος, προκύπτουν οι διαφορετικές εκδοχές των προβλημάτων αυτών. Για παράδειγμα υπάρχουν εκδοχές που το κόστος των εγκαταστάσεων είναι μια σταθερή ποσότητα και εξαρτάται αποκλειστικά από την τοποθεσία της εγκατάστασης και υπάρχουν εκδοχές που το κόστος της κάθε εγκατάστασης είναι μια συνάρτηση που εξαρτάται από το πλήθος των πελατών που χρησιμοποιούν την εγκατάσταση αυτή. Επίσης υπάρχουν εκδοχές το προβλήματος που επιβάλουν ένα κατώτατο ή ένα ανώτατο όριο πελατών που πρέπει να εξυπηρετεί κάθε ανοιγμένη εγκατάσταση και υπάρχουν εκδοχές που δεν επιβάλλονται τέτοιας φύσης περιορισμοί. Μπορούμε να δούμε όλα τα προβλήματα αυτά ως υποκατηγορία του ευρυτέρου κλάδου της σχεδίασης δικτύων. Για την επίλυση των προβλημάτων αυτών έχουν επιστρατευτεί πολλές διαφορετικές τεχνικές στην διεθνή βιβλιογραφία, με την πλειονότητα αυτών να βασίζονται σε γραμμικό προγραμματισμό ή στη τοπική αναζήτησης .

Ένας ακόμα ενδιαφέρον τομέας της θεωρητικής πληροφορικής είναι η αλγοριθμική θεωρία παιγνίων. Η θεωρία παιγνίων αποτελεί έναν καλά μελετημένο τομέα έρευνας με επιστήμονες να ασχολούνται με αυτόν από τον 17ο αιώνα. Την σύγχρονη του μορφή την απέκτησε γύρω στο 1950, την απαρχή για ένα εκτεταμένο ερευνητικό έργο. Από την άλλη, ο συνδυασμός πληροφορικής και θεωρίας παιγνίων αποτελεί έναν σχετικά νέο τομέα έρευνας. Αρκεί να αναλογιστεί κανείς ότι βασικές έννοιες όπως το Τίμημα της Αναρχίας ([74]) προτάθηκαν μετά το 1999. Τα τελευταία χρόνια η αλγοριθμική

θεωρία παιγνίων έχει ερευνηθεί εκτεταμένα και πολλά διαφορετικά παρακλάδια του τομέα αυτού έχουν δημιουργηθεί. Το κοινό στοιχείο όλων των προβλημάτων αυτών είναι ότι έχουμε πολλούς παίκτες που ο καθένας πράττει ανάλογα με το προσωπικό του συμφέρον αδιαφορώντας για τις επιπτώσεις που μπορεί να έχει στο σύνολο. Για παράδειγμα, ένας παίκτης που συμμετέχει σε μία δημοπρασία μπορεί να ποντάρει μια τιμή που να διαφέρει από την αξία που θεωρεί ότι έχει το προσφερόμενο αγαθό προσπαθώντας με αυτό τον τρόπο να πετύχει μια πιο ευνοϊκή τιμή. Επίσης θα μπορούσαμε σε ένα παιχνίδι συμφόρησης να έχουμε έναν παίκτη που χρησιμοποιεί την ταχύτερη διαδρομή που έχει στην διάθεση του για να φτάσει στο προορισμό του αλλά με αυτό τον τρόπο μπορεί να προκαλέσει μποτιλιάρισμα με αποτέλεσμα να καθυστερήσει σημαντικά τους υπόλοιπους παίκτες. Εμάς μας ενδιαφέρουν κατά κύριο λόγο τα παίγνια συμφόρησης. Σε αυτού του είδους τα παίγνια συνήθως έχουμε ένα σύνολο παικτών, ένα σύνολο πόρων και για κάθε παίκτη έχουμε ένα σύνολο στρατηγικών όπου κάθε στρατηγική αποτελείται από ένα υποσύνολο των διαθέσιμων πόρων. Στην πιο ενδιαφέρουσα περίπτωση, μπορούμε να θεωρήσουμε ότι έχουμε ένα δίκτυο και κάθε παίκτης θέλει να φτάσει από την αρχική του θέση σε έναν τελικό προορισμό. Οι ακμές του δικτύου αποτελούν τους πόρους και οι στρατηγικές είναι τα διαφορετικά μονοπάτια που μπορούν να χρησιμοποιήσουν οι παίκτες για να φτάσουν στον τελικό τους προορισμό.

Το βασικό πρόβλημα που προσπαθούμε να λύσουμε (εξ όσων γνωρίζουμε) αποτελεί την πρώτη προσπάθεια να συγχωνευτούν οι δύο παραπάνω τομής έρευνας. Ονομάζουμε το πρόβλημα αυτό Χωροθέτηση Εγκαταστάσεων για Ιδιοτελής Χρήστες (ή ΧΕΙΧ για συντομία). Στο πρόβλημα αυτό μας δίνεται ένα δίκτυο, ένα κόστος για το άνοιγμα εγκατάστασης σε κάθε κόμβο του δικτύου και μία συνάρτηση καθυστέρησης για κάθε ακμή που εξαρτάται από το πλήθος των ατόμων που χρησιμοποιούν την ακμή αυτή. Το πρόβλημα αυτό και οι διάφορες παραλλαγές του έχουν μεγάλο ενδιαφέρον τόσο από την σκοπιά της βελτιστοποίησης όσο και από την σκοπιά της θεωρίας παιγνίων.

1.1.1 Η συνεισφορά μας

Εξετάσαμε την εφαρμογή διάφορων αλγοριθμικών τεχνικών για την επίλυση του προβλήματος μας. Αρχικά εξετάσαμε τεχνικές τοπικής αναζήτησης και παραθέτουμε παραδείγματα και επιχειρήματα γιατί αυτές οι τεχνικές δεν δουλεύουν στο πρόβλημα μας. Για ειδικές περιπτώσεις του προβλήματος αποδεικνύουμε κάποια θεωρήματα σχετικά με την δομή της βέλτιστης λύσης. Στην συνέχεια όμως δείχνουμε ότι παρ' όλη την επιπρόσθετη πληροφορία οι τεχνικές γραμμικού προγραμματισμού δεν μπορούν να προσφέρουν επιπρόσθετη βοήθεια για την επίλυση του προβλήματος. Για την εκδοχή του προβλήματος όπου οι συναρτήσεις καθυστέρησης είναι φθίνουσες μπορέσαμε χρησιμοποιώντας τεχνικές από την υπάρχουσα βιβλιογραφία σε συνδυασμό με τις δικές μας ιδέες να καταλήξουμε σε έναν $log|S|$ προσεγγιστικό αλγόριθμο για το πρόβλημα. Σχετικά με το κομμάτι της θεωρίας παιγνίων αποδείξαμε την ύπαρξη κάποιον προσεγγιστικών ισορροπιών Νας. Τέλος χρησιμοποιώντας τεχνικές αραίωσης καταλήξαμε σε μία προσεγγιστική λύση σε μία παραλλαγή του προβλήματος μας όπου έχουμε μόνο

έναν κόμβο με παίκτες. Επίσης παρέχουμε ένα γενικό παράδειγμα που συνοψίζει την δυσκολία του προβλήματος και παρουσιάζει γιατί οι τεχνικές αραίωσης δεν μπορούν να εφαρμοστούν στην γενικότερη εκδοχή του προβλήματος.

1.1.2 Οργάνωση της Εργασίας

Το βασικό κομμάτι της εργασίας μπορεί να χωριστεί σε τέσσερις διαφορετικές ενότητες. Στην πρώτη ενότητα εξετάζουμε τις διαφορετικές τεχνικές και τις διαφορετικές εκδοχές των προβλημάτων χωροθέτησης εγκαταστάσεων που έχουν εξεταστεί από την διεθνή βιβλιογραφία. Στην δεύτερη ενότητα εξετάζουμε τεχνικές που έχουν χρησιμοποιηθεί σε προβλήματα σχεδίασης δικτύων. Στη τρίτη ενότητα αναλύουμε αναλύουμε διαφορετικά κομμάτια της υπάρχουσας βιβλιογραφίας πάνω στα παίγνια συμφόρησης. Τέλος, στην τέταρτη ενότητα παρουσιάζουμε τα δικά μας αποτελέσματα σχετικά με το πρόβλημα της Χωροθέτησης Εγκαταστάσεων για Ιδιοτελής Χρήστες.

1.2 Ορισμός του Προβλήματος

Στο πρόβλημα Χωροθέτησης Εγκαταστάσεων για Ιδιοτελής Χρήστες μας δίνετε ως είσοδος ένας γράφος $G = (V, E)$, ένα σετ από πελάτες $S \subseteq V$ και για κάθε πελάτη $s \in S$ ένα βάρος w_s. Για κάθε ακμή $e \in E$ μας δίνετε μία συνάρτηση καθυστέρησης $l_e : \Re \mapsto \Re$. Τέλος για κάθε κόμβο του γράφου $i \in V$ έχουμε ένα κόστος ανοίγματος εγκατάστασης f_i.

Στόχος μας είναι να ανοίξουμε ένα σετ $F \subseteq V$ από εγκαταστάσεις και να δρομολογήσουμε τους πελάτες στις εγκαταστάσεις αυτές με τέτοιο τρόπο ώστε να ελαχιστοποιούμε την έκφραση:

$$\sum_{e \in E} x_e \cdot l_e(x_e) + \sum_{j \in F} f_j$$

Όπου x_e είναι το συνολικό βάρος χρηστών που χρησιμοποιούν την ακμή e.

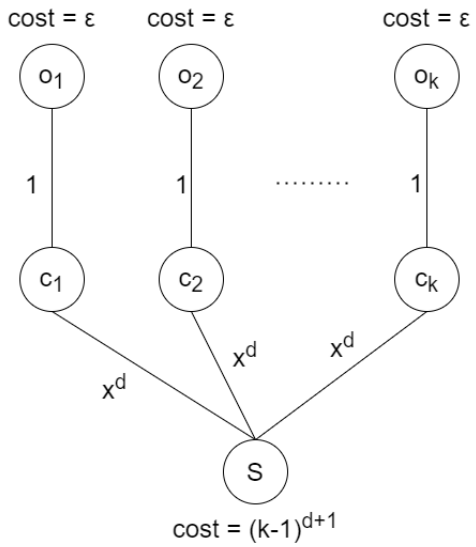
1.3 Τεχνικές Τοπικής Αναζήτησης

Όπως θα δούμε και στο κεφάλαιο 3.5, τεχνικές τοπικής αναζήτησης έχουν εφαρμοστεί για την επίλυση προβλημάτων Χωροθέτησης Εγκαταστάσεων. Έχουν χρησιμοποιηθεί και για την επίλυση προβλημάτων όπου το κόστος της εγκατάστασης εξαρτάται από το πλήθος των πελατών που εξυπηρετεί. Βλέπουμε λοιπόν ότι υπάρχει άμεση συνάφεια με το δικό μας πρόβλημα όπου η καθυστέρηση στις ακμές εξαρτάται από το πλήθος των πελατών που χρησιμοποιούν την ακμή.

Θα παρουσιάσουμε στο κεφάλαιο αυτό ένα αντιπαράδειγμα που δείχνει ότι για πολυωνυμικές συναρτήσει καθυστέρησης, ο λόγος το κόστους της τοπικά βέλτιστης λύσης με την βέλτιστη λύση είναι απαγορευτικά μεγάλος. Θα πάρουμε ως τοπικά

βήματα το **άνοιγμα** εγκατάστασης, το **κλείσιμο** εγκατάστασης και την **εναλλαγή** εγκατάστασης. Από την στιγμή που το κόστος μίας εγκατάστασης είναι σταθερό, άπαξ και έχουμε το πλήθος των ανοιχτών εγκαταστάσεων η δρομολόγηση πελατών σε αυτές μπορεί να βρεθεί αποδοτικά. Άρα δεν χρειάζονται τοπικά βήματα που να λαμβάνουν υπόψιν το πλήθος πελατών που λαμβάνει η κάθε εγκατάσταση.

Στο παράδειγμα του σχήματος 1.1 έχουμε στους κόμβους $c_1, c_2, ..., c_k$ πελάτες μοναδιαίου βάρους. Το κόστος ανοίγματος εγκατάστασης στους κόμβους $o_1, o_2, ..., o_k$ είναι ϵ, η η καθυστέρηση των ακμών (c_i, o_i) είναι σταθερή και ίση με 1 και η καθυστέρηση των ακμών $e = (c_i, S)$ είναι $l_e(x) = x^d$. Τέλος το κόστος ανοίγματος εγκατάστασης στο S είναι $(k-1)^{d+1}$. Η βέλτιστη λύση είναι να ανοίξουν όλες οι εγκατάστασης των κόμβων o_i και ο κάθε πελάτης να δρομολογηθεί στην αντίστοιχη εγκατάσταση με συνολικό κόστος $k \cdot (\epsilon + 1)$. Ας αναλογιστούμε την περίπτωση όπου έχουμε μόνο την εγκατάσταση S ανοιχτή. Το **άνοιγμα** οποιασδήποτε άλλης εγκατάστασης απλά θα επέφερε ένα έξτρα ϵ κόστος. Η **εναλλαγή** της εγκατάστασης με οποιοδήποτε άλλη θα επέφερε ένα έξτρα $2 \cdot k - 2 + \epsilon$ κόστος. Το κόστος της κατάστασης αυτής είναι $3 \cdot (k-1) + 1 + (k-1) \cdot (k-1)^d$. Άρα η κατάσταση αυτή είναι τοπικά βέλτιστη και ο λόγος του κόστους της προς το κόστος της βέλτιστης λύσης είναι $O(k^d)$.



Σχήμα 1.1: Ένα αντιπαράδειγμα για την χρήση Τοπικής Αναζήτησης.

1.4 Γραμμικός Προγραμματισμός

Για τους σκοπούς το κεφαλαίου αυτού θα απλουστεύσουμε το πρόβλημα εξετάζοντας αποκλειστικά μία ειδική του περίπτωση. Θα θεωρήσουμε ότι το κόστος όλων των

εγκαταστάσεων είναι B ($f_j = B, \forall j \in V$) και ότι όλες οι συναρτήσεις καθυστέρησης είναι γραμμικές με θετικούς συντελεστές ($l_e(x) = a \cdot x + b$ όπου $a, b \geq 0, \forall e \in E$).

Στο λήμμα 6.3.1 αποδεικνύουμε ότι στις περιπτώσεις που πληρούνται τα παραπάνω κριτήρια υπάρχει μία λύση με κόστος το πολύ τέσσερις φορές της βέλτιστης λύσης ($4 * OPT$) όπου κάθε ανοιχτή εγκατάσταση δέχεται τουλάχιστον $\min_{s \in S} \frac{w_s}{2}$ ζήτηση.

Είναι προφανές ότι εξετάζουμε την περίπτωση όπου οι πελάτες μπορούν να 'σπάσουν' το βάρος τους σε μικρότερα κομμάτια και να το δρομολογήσουν ξεχωριστά. Οι περιορισμοί που παίρνουμε είναι αναγκαίοι καθώς δεν είναι δύσκολο να δει κανείς ότι στην γενικότερη περίπτωση κάτι τέτοιο δεν ισχύει.

Επί της ουσίας ο τρόπος που το αποδεικνύουμε είναι ότι χωρίζουμε τις εγκαταστάσεις της βέλτιστης λύσης σε αυτές που δέχονται πάνω από $\min_{s \in S} \frac{w_s}{2}$ ζήτηση και σε αυτές που δέχονται λιγότερη ζήτηση. Αρχικά ασχολούμαστε με τους πελάτες που στέλνουν το μεγαλύτερο κομμάτι του βάρους τους στην πρώτη κατηγορία εγκαταστάσεων. Τους βάζουμε να στείλουν όλο τους το βάρος σε εγκαταστάσεις της πρώτης κατηγορίας αυξάνοντας έτσι το πολύ κατά 2 τις ροές σε κάθε ακμή και άρα το πολύ κατά 4 φορές το κόστος. Στην συνέχεια κλείνουμε ότι εγκαταστάσεις έχουν μείνει χωρίς ζήτηση. Τέλος ασχολούμαστε με τους πελάτες που στέλνουν το μεγαλύτερο κομμάτι του βάρους τους στην δεύτερη κατηγορία εγκαταστάσεων. Με απαγωγή σε άτοπο δείχνουμε ότι το πλήθος των εγκαταστάσεων της δεύτερης κατηγορίας που έχουν μείνει ανοιχτά είναι περισσότερα σε πλήθος από τους πελάτες αυτούς. Άρα κλείνουμε όλες τις εγκαταστάσεις της δεύτερης κατηγορίας και ανοίγουμε από μία εγκατάσταση πάνω σε κάθε πελάτη της κατηγορίας αυτής. Κρατώντας την ροή των πελατών αυτών προς τις εγκαταστάσεις της πρώτης κατηγορίας σταθερή και στέλνοντας την υπόλοιπη ροή τους στην εγκατάσταση που άνοιξε πάνω τους έχουμε καταλήξει σε μία λύση που ικανοποιεί τους περιορισμούς του λήμματος.

Στην συνέχεια αναλογιζόμαστε την ακόλουθη διατύπωση του προβλήματος:

$$min \sum_{e \in E} (a_e \cdot x_e^2 + b_e \cdot x_e) + \sum_{e \in F} z_e \cdot B$$

$$\begin{array}{lrcll} s.t. & \sum_{e \in \delta^-(v)} x_e & = & \sum_{e \in \delta^+(v)} x_e, & \forall v \in V \text{ (διατήρηση ροής)} \\ & x_e & \leq & z_e \cdot \sum_{s \in S} w_s, & \forall e \in F \text{ (χρήση μόνο αγορασμένων εγκαταστάσεων)} \\ & x_e & = & w_s & \forall e \in S \\ & z_e & \in & \{0.1\} & \forall e \in F \end{array}$$

Για να μπορούμε να λύσουμε το πρόβλημα αυτό με χρήση Γραμμικού Προγραμματισμού πρέπει να αφαιρέσουμε τον ακέραιο περιορισμό και να τον αντικαταστήσουμε με έναν συνεχή. Όμως παρατηρούμε ότι με το που το κάνουμε αυτό τότε το κόστος τον εγκαταστάσεων θα είναι ανεξάρτητα από την τοποθεσία τους και το πλήθος τους ίσο με B. Άρα οι παίκτες πάντα θα ανοίγουν μια εγκατάσταση στον κόμβο τους θα δρομολογούν όλη την ροή τους εκεί και συνολικά θα έχουμε κόστος B. Άρα η βέλτιστη λύση του προβλήματος με συνεχείς περιορισμούς δεν παρέχει καμία πληροφορία για την επίλυση του προβλήματος.

1.5 Τεχνικές Σχεδίασης Δικτύων

Σε αυτή την ενότητα θα εξετάσουμε την εκδοχή του προβλήματος όπου αντί η κάθε εγκατάσταση να έχει ένα κόστος, έχουμε τον περιορισμό ότι μπορούμε να ανοίξουμε το πολύ k εγκαταστάσεις. Επίσης θεωρούμε ότι για τις συναρτήσεις καθυστερήσεις ισχύουν οι εξής περιορισμοί. Οι συναρτήσεις $l_e : \Re \mapsto \Re$ είναι φθίνουσες, μη αρνητικές και η συνάρτηση $x \cdot l_e(x)$ είναι αύξουσα και κυρτή. Ονομάζουμε την έκδοση του προβλήματος αυτή Κυρτή Χωροθέτηση Εγκαταστάσεων για Ιδιοτελής Χρήστες (ΚΧΕΙΧ).

Καταρχάς αποδεικνύουμε το θεώρημα 6.4.3 που δηλώνει ότι υπάρχει μια βέλτιστη λύση που δεν περιέχει κύκλους. Άρα μπορούμε να σκεφτούμε την βέλτιστη λύση ως ένα δάσος από δέντρα με ρίζες τους κόμβους των εγκαταστάσεων.

Στην συνέχεια δείχνουμε ότι το πρόβλημα μας γενικεύει το πρόβλημα που αναλύεται στο άρθρο [79] και ως εκ τούτου γενικεύει ένα μεγάλο αριθμό διαφορετικών προβλημάτων Χωροθέτησης Εγκαταστάσεων και Σχεδίασης Δικτύων. Εν συντομία για να αποδείξουμε τον ισχυρισμό αυτό παρατηρούμε ότι αν θέσουμε τις συναρτήσεις καθυστέρησης μας ως:

$$l_e(x) = \begin{cases} \frac{c(e)}{x} + l(e) & \text{αν } x_e > \min_{s \in S}(w_s) \\ \frac{c(e)}{\min_{s \in S}(w_s)} + l(e) & \text{αν } x_e < \min_{s \in S}(w_s) \end{cases}$$

Θα έχουμε συναρτήσεις που ικανοποιούν τους περιορισμούς μας και θα έχουν ίδιο κόστος με το κόστος των ακμών του [79] (όπου $c(e), l(e)$ τα αντίστοιχα κόστη του [79]). Επίσης η εύρεση k εγκαταστάσεων είναι πιο ισχυρή από την εκδοχή που έχουμε έναν σταθερό τελικό προορισμό για όλους τους πελάτες. Η παρατήρηση αυτή μας δίνει κατευθείαν ότι είναι δύσκολο να βρεθεί $\Omega(loglog|S|)$ προσεγγιστικός αλγόριθμος για το πρόβλημα μας ([30]).

Για τους σκοπούς του αλγορίθμου μας ορίζουμε ως $P^w_{u,v}$ το γρηγορότερο (u, v) μονοπάτι για πελάτη βάρους w που κινείται μόνος στο δίκτυο. Επίσης $g(u, v, w) = \sum_{e \in P^w_{u,v}} w \cdot l_e(w)$. Ο αλγόριθμος είναι ο εξής.

1. Αρχικοποιούνται $S_0 = S$, $w_{0,s} = w_s$ και $i = 0$.
2. Όσο $|S_i| > k$:
   (α′) Για κάθε ζευγάρι $u, v \in S_i$ βρες $K_i(u, v) = \min_{z \in V}\{g(u, z, w_{i,u}) + g(v, z, w_{i,v})$
   $+ \frac{w_{i,u}}{w_{i,u}+w_{i,v}} \cdot g(z, u, w_{i,u} + w_{i,v}) + \frac{w_{i,v}}{w_{i,u}+w_{i,v}} \cdot g(z, v, w_{i,u} + w_{i,v})\}$.
   (β′) Κάνε ένα ελάχιστο ταίριασμα στο S_i με κόστη K_i αφήνοντάς k αταίριαστους κόμβους.
   (γ′) Θέσε $S_{i+1} = \{\}$.
   (δ′) Για κάθε ταίριασμα u, v:

i. Στείλε και τους δύο πελάτες στον κόμβο z που ελαχιστοποιεί την έκφραση του 2(α).

ii. Διάλεξε u με πιθανότητα $\frac{w_{i,u}}{w_{i,u}+w_{i,v}}$, αλλιώς διάλεξε v. Χωρίς βλάβη της γενικότητας θεωρούμε ότι διαλέξαμε τον u.

iii. Στείλε τα $w_{i,u} + w_{i,v}$ πίσω στο u, πρόσθεσε u στο S_{i+1} και θέσε $w_{i+1,u} = w_{i,u} + w_{i,v}$.

(ε′) Πρόσθεσε και τους αταίριαστους κόμβους στο S_{i+1}

(ϛ′) Θέσε $i \leftarrow i + 1$

3. Ο αλγόριθμος επιστρέφει ως εγκαταστάσεις τους k κόμβους που ανήκουν στο S_i και ως ροές αυτές που υπαγορεύει η παραπάνω διαδικασία.

Αποδεικνύεται εύκολα ότι ο αλγόριθμος αυτός τερματίζει μετά από $O(log|S|)$ βήματα. Στην συνέχεια στο λήμμα 6.4.6 αποδεικνύουμε ότι το ταίριασμα του 2(α) μπορεί να γίνει πάνω στο δάσος της βέλτιστης λύσης και οι πελάτες χρησιμοποιούν ακμές που χρησιμοποιούσαν στην βέλτιστη λύση. Για την απόδειξη αυτή περιγράφουμε μια διαδικασία έχοντας ως δεδομένη την βέλτιστη λύση που καταλήγει σε ένα ταίριασμα που ικανοποιεί τους περιορισμούς του και άρα έτσι αποδεικνύεται η ύπαρξη του. Πιο συγκεκριμένα παίρνουμε το μονοπάτι κάθε πελάτη προς την εγκατάσταση στο δέντρο του στην βέλτιστη λύση. Όσο πλησιάζουμε προς την ρίζα (εγκατάσταση) θα υπάρχουν κόμβοι στους οποίους συναντιούνται τα μονοπάτια διαφορετικών πελατών. Σε αυτούς τους κόμβους ταιριάζουμε όλους τους πελάτες που μπορούμε και στην χειρότερη μένει ένας αταίριαστος. Τους αταίριαστους πελάτες τους προχωράμε πιο 'πάνω' στα μονοπάτια τους μέχρι να συναντήσουν μονοπάτια άλλων αταίριαστων πελατών και ούτω καθεξής. Τέλος θα καταλήξουμε να συναντιούνται όλα τα μονοπάτια στην ρίζα στην οποία ταιριάζονται οι εναπομείναντες πελάτες και μένει το πολύ ένας αταίριαστος από κάθε δέντρο. Γι' αυτό αφήνουμε k αταίριαστους πελάτες (εκτός από όταν έχουμε $k + 1$ πελάτες που θα κάνουμε μόνο ένα ταίριασμα για να τελειώσει ο αλγόριθμος).

Στο λήμμα 6.4.7 δείχνουμε ότι στο πρώτο ταίριασμα το κόστος δρομολόγησης των πελατών στα σημεία συνάντησης είναι λιγότερο από το κόστος της βέλτιστης λύσης. Το λήμμα αυτό το γενικεύουμε στο λήμμα 6.4.9 που δηλώνουμε ότι το αναμενόμενο κόστος της δρομολόγησης των πελατών στα σημεία συνάντησης είναι λιγότερο από το κόστος της βέλτιστης λύσης. Για να το αποδείξουμε αυτό κάνουμε χρήση της κυρτότητας του κόστους, της ανισότητα Γένσεν και των περιορισμών που έχουμε λόγο του ταιριάσματος που αποδείξαμε ότι υπάρχει.

Τέλος στο λήμμα 6.4.8 δείχνουμε ότι το αναμενόμενο κόστος της δρομολόγησης των ταιριασμένων πελατών πίσω είναι μικρότερο από το κόστος της δρομολόγησης τους στο σημείο συνάντησης και άρα λιγότερο συνολικά από το κόστος της βέλτιστης λύσης. Για να το αποδείξουμε αυτό χρησιμοποιούμε ότι οι συναρτήσεις καθυστέρησης είναι φθίνουσες.

Συνδυάζοντάς όλα τα παραπάνω λήμματα καταλήγουμε στο θεώρημα 6.4.11 που δηλώνει ότι ο παραπάνω αλγόριθμος είναι ένας $O(log|S|)$ προσεγγιστικός αλγόριθμος

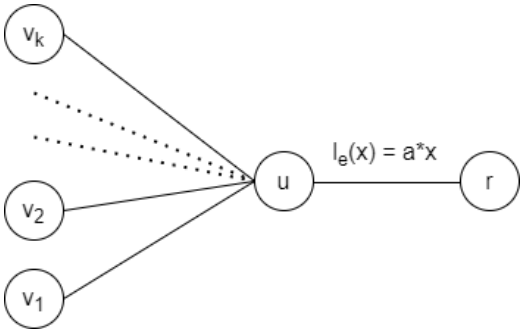
για το ΚΧΕΙΧ πρόβλημα.

1.6 Θεωρία Παιγνίων

Για τους σκοπούς του κεφαλαίου αυτού δεν αλλάζουμε το πρόβλημα αλλά αλλάζουμε τον τρόπο που το αντιμετωπίζουμε. Θεωρούμε ότι έχουμε έναν έξτρα κόμβο t που είναι ο τελικός προορισμός όλων των πελατών και συνδέουμε όλους τους κόμβους του G με ακμές καθυστέρησης $l_e(x) = \frac{B}{x}$ με τον t. Επίσης θεωρούμε τις καθυστερήσεις στις υπόλοιπες ακμές γραμμικές με θετικούς συντελεστές. Είναι εύκολο να συνειδητοποιήσουμε ότι το πρόβλημα είναι ισοδύναμο με αυτό που συζητήσαμε στην ενότητα 1.4 με την διαφορά ότι οι παίκτες μοιράζονται το κόστος της εγκατάστασης.

Είναι εύκολο να δείξουμε ότι αν όλοι οι παίκτες 'ανοίξουν' μια εγκατάσταση στον κόμβο τους έχουμε μια $(\frac{w_{\max}}{w_{\min}} + 1)$-προσεγγιστική ισορροπία Νας για την περίπτωση όπου οι πελάτες δεν μπορούν να σπάσουν το βάρος τους και $(\frac{w_{\max}}{w_{\min}})$ -προσεγγιστική ισορροπία Νας για την περίπτωση όπου οι πελάτες μπορούν να σπάσουν το βάρος τους. Άρα αν όλοι οι πελάτες έχουν το ίδιο βάρος έχουμε μια 2-προσεγγιστική και μία ακριβής ισορροπία Νας.

Επίσης στο λήμμα 6.5.1 αποδεικνύουμε ότι το Τίμημα της Αναρχίας για το παίγνιο αυτό είναι κάτω φραγμένο από το $O(n)$. Για να το δούμε αυτό αρκεί να αναλογιστούμε το παράδειγμα του γραφήματος 1.2. Όλες οι ακμές έχουμε μηδενική καθυστέρηση εκτός από την (u, r) που έχει $l_e(x) = a \cdot x$ με $a = \frac{k-1}{k^2} \cdot B - \epsilon$ όπου $\epsilon > 0$. Τέλος οι κόμβοι v_i έχουν μοναδιαίο βάρος. Η βέλτιστη λύση είναι να ανοίξει η εγκατάσταση στο u με συνολικό κόστος B. Όμως αν είναι ανοιχτή μόνο η εγκατάσταση του r έχουμε ισορροπία Νας με κόστος $a \cdot k^2 + B = (k-1) \cdot B + B = k * B$ που αποδεικνύει τον ισχυρισμό.



Σχήμα 1.2: Ένα δίκτυο με $O(n)$ Τίμημα της Αναρχίας

Από την άλλη δεν είναι δύσκολο να αποδείξουμε ότι το Τίμημα της Αναρχίας είναι άνω φραγμένο από το $O(n)$. Αυτό το αποδεικνύουμε στο λήμμα 6.5.2. Εν συντομία αυτό που λέμε είναι ότι στην βέλτιστη λύση ανοίγει τουλάχιστον μια εγκατάσταση άρα το κόστος της είναι τουλάχιστον B. Από την άλλη σε μία ισορροπία Νας κάθε κόμβος θα έχει το πολύ μια εγκατάσταση ανοιγμένη πάνω του και αν έχει πελάτη

το κόστος δρομολόγησης του θα είναι λιγότερο από το B καθώς ο πελάτης μπορεί πάντα να χρησιμοποιήσει την εγκατάσταση του κόμβου του με μέγιστο κόστος B. Άρα το κόστος οποιασδήποτε ισορροπίας είναι το πολύ $O(n) \cdot B$ που αποδεικνύει τον ισχυρισμό.

1.7 Τεχνικές Αραίωσης

Στην ενότητα αυτή ασχολούμαστε με μία απλούστερη εκδοχή του προβλήματος μας. Θεωρούμε ότι υπάρχει μόνο ένα κόμβος s στον οποίο είναι μαζεμένοι άπειροι παίκτες συνολικού βάρους w_s. Επίσης το κόστος της κάθε εγκατάστασης διαφέρει από κόμβο σε κόμβο. Ο σκοπός μας είναι να ανοίξουμε τις εγκαταστάσεις αυτές που το κόστος τους και το κόστος της ροής Νας που δημιουργείται να είναι το μικρότερο δυνατό.

1.7.1 Το Λήμμα του Αλτχόφερ

Με χρήση του λήμματος του Αλτχόφερ μπορούμε να δείξουμε ότι για οποιαδήποτε ροή f υπάρχει μία ροή $\hat{f}$ η οποία χρησιμοποιεί το πολύ πολύ $\lfloor \frac{\log(2 \cdot m)}{2 \cdot \epsilon^2} \rfloor + 1$ μονοπάτια και ισχύει ότι $|\hat{f}_e - f_e| \leq \epsilon$ για κάθε $e \in E$. Σημαντική είναι η παρατήρηση ότι η ροή $\hat{f}$ δεν χρησιμοποιεί ακμές που έχουν μηδενική ροή στο f. Άρα η $\hat{f}$ ανοίγει μόνο εγκαταστάσεις που έχει ανοίξει και η f. Άρα το κόστος των εγκαταστάσεων της $\hat{f}$ είναι άνω φραγμένο από το κόστος των εγκαταστάσεων της f.

Επίσης παίρνουμε σαν έξτρα περιορισμούς ότι $w_s = 1$, $l_e(x) = a_e \cdot x + b_e$ με $a = \max_{e \in E}\{a_e\}$ και ότι $|P| \leq m^{d_1}$ και $|p| \leq \log^{d_2} m$, όπου P το σύνολο των διαφορετικών απλών s, t μονοπατιών και p οποιοδήποτε από τα μονοπάτια αυτά.

Στην συνέχεια δείχνουμε ότι μπορούμε με εξαντλητική αναζήτηση σε όλους τους διαφορετικούς $m^{O(d_1 \cdot a^2 \cdot \log^{2 \cdot d_2 + 1}(2 \cdot m)/\epsilon^2)}$ συνδυασμούς μονοπατιών να βρούμε ροή $\hat{f}$ για την οποία να ισχύει ότι $l_p(\hat{f}_p) \leq L(f^*) + \epsilon$ όπου $L(f^*)$ το κοινό κόστος όλων των μονοπατιών της βέλτιστης ισορροπίας Νας. Συνδυάζοντάς τα παραπάνω καταλήγουμε ότι για το κόστος της ροής μας ($\hat{C}$) σε σχέση με το κόστος της βέλτιστης λύσης (C^*) ισχύει ότι $\hat{C} \leq C^* + \epsilon$ και ότι ροή $\hat{f}$ είναι μια προσεγγιστική ισορροπία Νας στον γράφο G.

Σε ένα παρόμοιο συμπέρασμα μπορούμε να καταλήξουμε και με τη χρήση του προσεγγιστικού θεωρήματος του Καραθεοδωρή.

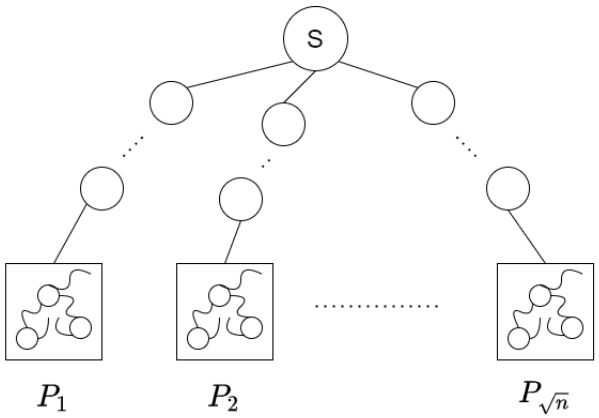
1.7.2 Ένα Δύσκολο Παράδειγμα

Είναι φυσικό να αναρωτηθούμε τι μπορούμε να κάνουμε με τις τεχνικές αυτές στην περίπτωση όπου έχουμε πολλαπλούς κόμβους με πελάτες. Δεν μπορούμε να εφαρμόσουμε τις τεχνικές αυτές για κάθε παίκτη ξεχωριστά γιατί τότε θα είχαμε επαναλήψεις εκθετικές στο πλήθος των παικτών κάτι που μπορεί αν είναι υπολογιστικά δυσβάσταχτο. Άρα πρέπει να εξετάσουμε την ροή συνολικά. Όμως πελάτες με βάρος

μικρότερο του ϵ μπορεί να αγνοηθούν πλήρως από τη λύση και άρα η ροή να μην είναι εφικτή λύση του προβλήματος.

Στο παράδειγμα το σχήματος 1.3, για οποιοδήποτε n, κατασκευάζουμε $\sqrt{n}$ διαφορετικά δύσκολα παραδείγματα $(P_1, P_2, ..., P_{\sqrt{n}})$ που το κάθε ένα έχει βάρος πελατών $\frac{1}{\sqrt{n}}$. Οι προσεγγιστικοί αλγόριθμοι που έχουμε στην διάθεση μας έχουν λόγο προσέγγισης $O(n)$. Η ροή που θα ψάξουμε με εξαντλητική αναζήτηση θα έχει το πολύ k μονοπάτια. Για να καταλήξουμε σε ένα αλγόριθμο μη εκθετικό θα πρέπει $k = poly(log(n))$, δηλαδή το k να είναι της μορφής $\sum_i \alpha_i \cdot log^{\beta_i} n$. Όμως αυτό σημαίνει ότι ένα πολύ μεγάλο κομμάτι του βάρους των ροών $(1 - \frac{\sqrt{n}}{poly(log(n))} \to 1, n \to \infty)$ θα αγνοηθεί.

Αν δεν ανοίξουμε καινούργιες εγκαταστάσεις τότε το συνολικό κόστος θα είναι μεγαλύτερο από το άνοιγμα αποκλειστικά μία εγκατάσταση στο S. Από την άλλη αν λύσουμε με τον $O(n)$ προσεγγιστικό αλγόριθμο όλα τα $\sqrt{n}-k$ προβλήματα που έχουν αγνοηθεί ο λόγος προσέγγισης της λύσης μας θα είναι κάτω φραγμένος από την τιμή $n^{\frac{1}{2}-\lambda}$ για οποιοδήποτε $\lambda > 0$. Επίσης αυτή την προσέγγιση δεν είναι προφανές πως θα την καταφέρναμε καθώς δεν είναι προφανές πως θα ξεχωρίζαμε τα διαφορετικά P_i προβλήματα μεταξύ τους σε μία λίγο πιο περίπλοκη περίπτωση. Αν όμως μπορούσαμε να τα ξεχωρίσουμε, τότε η απευθείας χρήση των $O(n)$ προσεγγιστικών αλγορίθμων σε κάθε ένα από αυτά ξεχωριστά θα μας έδινε μια $O(n^{\frac{1}{2}})$ προσέγγιση. Άρα η χρήση των τεχνικών αραίωσης δεν προσέφερε απολύτως κανένα πλεονέκτημα.



Σχήμα 1.3: Ένα παράδειγμα στο οποίο η χρήση τεχνικών αραίωσης είναι ανούσια.

Chapter 2

Introduction

2.1 Motivation

One of the most heavily researched areas in theoretical computer science literature have been facility location problems. In such problems the goal usually is to open a set of facilities in a way that minimizes the opening cost of those facilities and the routing cost of clients to those facilities. Depending on the cost metrics and any extra restrictions that we might impose on the optimal solution we can come up with many different variants of the problem. For instance, if the cost of each facility depends on the amount of demand it serves we are left with the Universal Facility Location problem, or if we impose to each facility a minimum amount of demand it must serve when opened, then we are left with the Load Balanced Facility Location problem. We may view facility location problems as a sub-field of Network Design problems that we will also examine in this thesis. To solve these problems a variety of techniques have been employed, from Linear Programming to Local Search algorithms.

Another really interesting field of research in computer science is Algorithmic Game Theory. Game Theory is a well known field of research with early works dating back to the 17th century. It started taking its modern form around the 1950s when it received heavy attention from the scientific community. On the other hand the combination of Game Theory with computer science has gained quite recently a lot of attention. Some very important concepts of the field such as the Price of Anarchy [74] had not been formulated up until 1999. Since then extensive work has been conducted on Algorithmic Game Theory with many different branches of the field emerging. The underlying characteristic of these problems is that we have many players that act based on their personal interests rather than the collective welfare. So, for instance, in the sub-field of mechanism design we might have bidders in an auction that do not report truthfully their valuation for the auctioned product. We might also have agents in congestion games that use strategies that minimize their total cost but impose a heavy cost on their fellow "commuters".

We are specifically interested in the sub-field of Algorithmic Game Theory called Congestion Games. In this kind of games we have a set of resources, a set of players and a set of strategies each player has available. Each strategy consists of a set of resources. Each resource has a cost which depends on the amount of players using that resource (i.e. it is susceptible to congestion). In the more interesting case there is a network where each players is at a node and wants to reach a destination. The edges of the network can be used from players in order to construct different paths to reach their destination.

The main problem studied in this thesis is to the best of our knowledge the first to combine the fields of Facility Location and Congestion Games. We call our problem Facility Location for Selfish Commuters (or FLSC for short). In this problem we are given a network and a specific cost for opening facilities in each node of the network. We are also given a set of weighted players that want to be routed from their original node to a facility. The latency on the edges of the network depends on the total weight of players using it (i.e. is susceptible to congestion). This problem and its variants have a lot of motivation both from an optimization and a game theoretic point of view.

2.2 Our Contribution

We examine how some of the main existing algorithmic techniques available can be used to approximate our problem and its variants. We examine local search algorithms and we provide counter examples and compelling arguments on why most likely those techniques can not provide valuable insight to our main problem. We then examine some special cases of the problem. We prove some theorems regarding the structure of the optimal solution of the problem however we also show that some linear program formulations have restrictive integrality gap. We also focus on instances where the edge cost is a decreasing function for which instances we were able to provide a $log|S|$ approximation algorithm for the k-median variant of the problem. We were also preoccupied with the game theoretic point of view of the problem. We were mainly interested in instances where players pay for the facility cost through cost sharing mechanisms. We were able to provide some matching upper and lower bounds on the Price of Anarchy and compute approximate Nash Equilibria. Finally using sparsification techniques we were able to solve a special case of the FLSC problem where there is only one demand point. For the multi commodity version of the problem we provide a counter example that captures the difficult essence of the problem and show why sparsification techniques fail.

2.3 Organization of Thesis

There are four different main chapters. In the first chapter we examine different techniques used to solve different variants of the facility location problem. In the second chapter we examine techniques used to solve network design problems. In the third chapter we examine congestion games and the notion of selfish routing. Finally, in the penultimate chapter we formalize our problem and try to see how the techniques we have analyzed in the previous chapters can be used to solve our problem. In this chapter we combine the already existing techniques with some novel ideas in order to come up with algorithms and valuable insight concerning the Facility Location for Selfish Commuters problem.

Chapter 3

Facility Location Problems

Facility location is a well studied research area. It mainly addresses problems in which a network is given as input and a set of facilities needs to be opened that minimize the opening cost and the routing cost of demands to those facilities. The different variants of this problem arise from the different restrictions those facilities have to comply with and the different ways the total cost of a solution is calculated. Although intuitively it would make more sense to begin this thesis with Network Design Problems, because Facility Location problems can be seen as a special case of Network Design, we are going to analyze some very interesting problems that provide valuable tools that will help to tackle more intricate Network Design Problems.

In this section we will examine some very interesting techniques and approaches. Since our main problem is, in its essence, a Facility Location problem with latencies susceptible to congestion, it seems appropriate to examine the existing bibliography regarding Facility location problems and try to adapt the existing techniques to our problem. That is why, in this section, we will analyze Facility Location problems that have been solved with various different techniques.

We will examine Linear Programming and the primal-dual method which is a powerful tool for approximating NP-Hard problems. We will also briefly refer to Lagrangian relaxation, a technique that helps us use pre-existing tools in problems with additional constraints. We will see how problems can be transformed to simpler ones while incurring a small blow-up in cost. Also we will see how complicated problems can be solved using as black-boxes the algorithms of simple variants of the problem. Additionally, it will be really important to examine problems with multiple costs and how probabilistic algorithms can help us handle those cases. Finally, we will examine the use of local search in facility location problems and more specifically for solving the Universal Facility Location problem.

3.1 Metric Uncapacitated Facility Location

It would be impossible to talk about Facility Location problem without mentioning the Metric Uncapacitated Facility Location problem. It is the most basic and well studied facility location problem that one can think of. For this problem we will examine a Linear Programming algorithm that uses the primal-dual method. Readers unfamiliar with those methods should refer to [93] for further analysis. The primal-dual technique was first introduced by Dantzig et al. [32] and it has been used to solve many combinatorial optimization problems and more specifically network design problems ([4], [3], [50], [46], [52], [53], [70], [81]).

In this problem we are trying to find the optimal way to open a set of facilities and route all of the given demand to a facility location. More formally, in an instance of the Metric Uncapacitated Facility Location we are given a bipartite graph G for which $F \subset G$ is the set of possible facility nodes and $C \subset G$ is the set of clients. We denote as f_j the cost of opening facility $j \in F$. We also have a cost of connecting each client $i \in C$ to a facility $j \in F$ which we will denote as c_{ij}. A feasible solution to this problem consists of a subset $I \subseteq F$ of facilities to open and an assignment of clients to those facilities I.

The first constant factor algorithm for this problem was provided by Shmoys et al. [88]. They provided a guarantee of 3.16 times the optimal cost. Chudak et al. [29] found a $(1 + 2/e)$-approximate algorithm and finally Li [75] provided a 1.488-approximation algorithm for the problem. As far as inapproximability results are concerned, the work of Guha et al. [55] prove that it is NP-Hard to approximate this problem with a factor better than 1.463.

The following is the natural LP formulation of the problem. The variable y_i becomes 1 when we open facility i otherwise is 0. Also variable x_{ij} becomes one when client i is assigned to facility j and 0 otherwise.

$$\begin{aligned}
\min \quad & \sum_{i \in F, j \in C} c_{ij} \cdot x_{ij} \quad + \quad \sum_{i \in F} f_i \cdot y_i & & & (3.1)\\
\text{s.t.} \quad & \sum_{i \in F} x_{ij} \quad \geq \quad 1 & & \forall j \in C & (3.2)\\
& y_i \quad \geq \quad x_{ij} & & \forall i \in F, j \in C & (3.3)\\
& x_{ij} \quad \in \quad \{0,1\} & & \forall i \in F, j \in C & (3.4)\\
& y_i \quad \in \quad \{0,1\} & & \forall i \in F & (3.5)
\end{aligned}$$

Vazirani et al. [63] use a Primal-Dual technique to achieve a 3-approximate algorithm of the problem at hand.

After relaxing the integral constraint we can formulate the dual program which is the following.

$$\max \quad \sum_{j \in C} a_j \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j \in C} b_{ij} \leq f_i \qquad \forall i \in F \tag{3.2}$$

$$a_j - b_{ij} \leq c_{ij} \qquad \forall i \in F, j \in C \tag{3.3}$$

$$b_{ij} \geq 0 \qquad \forall i \in F, j \in C \tag{3.4}$$

$$a_j \geq 0 \qquad \forall j \in C \tag{3.5}$$

The algorithm of Vazirani et al. finds a dual feasible solution through which it determines a set of tight edges and a set of facilities. Then they choose a subset of those facilities and map clients to those open facilities.

They also use their algorithm to solve the k-median problem using Lagrangian Relaxation. First of all, the k-median problem is similar to the Uncapacitated facility location. In this variant we do not have a facility cost however we are obliged to open exactly k facilities. The natural LP formulation of the problem is as follows:

$$\min \quad \sum_{i \in F, j \in C} c_{ij} \cdot x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i \in F} x_{ij} \geq 1 \qquad \forall j \in C \tag{3.2}$$

$$\sum_{i \in F} y_i \leq k \tag{3.3}$$

$$y_i \geq x_{ij} \qquad \forall i \in F, j \in C \tag{3.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in F, j \in C \tag{3.5}$$

$$y_i \in \{0,1\} \qquad \forall i \in F \tag{3.6}$$

Charikar et al. [22] were able to provide a $O(log(n) \cdot loglog(n))$-approximate algorithm which was obtained by derandomizing and refining of an algorithm proposed by Bartal ([14], [15]). Tardos et al. [23] were able to provide the first constant approximation algorithm for the k-median problem with a guarantee of $6\frac{2}{3}$. Finally, Arya et al. [9] provided a $3+\epsilon$ approximation algorithm. We can also derive a $(1+\frac{2}{e}-\epsilon)$ inapproximability result by adapting the proof of hardness for the facility location problem provided by Guha et al [55] (this was observed by Jain et al [62]).

What they basically do is solve the Uncapacitated Facility location problem by putting a cost F to those facilities. Then they perform binary search on the

different values of F until they find a solution that opens exactly k facilities. This technique yields a 6-approximate algorithm for the k-median variant of the problem.

3.2 Hierarchical Caching

Hierarchical Caching is a very interesting problem because it generalized some interesting facility location problems such as the Multi-Level facility location problem. In this section we will analyze the algorithm proposed by Guha et all [54] and use the same definitions and notations as them. In order to solve the General problem which we will call General-Placement problem we will first solve two easier special cases of the problem, the Simple-Placement and the Multi-Level facility location.

While solving the Simple-Placement problem we will examine how a problem can be transformed to a instance of the same problem with additional constraints which in turn render the problem easy to solve with pre-existing tools and algorithms. We will also see, in the section regarding the Multi-Level facility location problem, how a complex facility location problem can be solved using a more extensively studied simpler facility location problem. The key to this technique is finding the correct costs for the simpler facility location problem which will yield a solution to the original problem that is within a constant factor from the optimal solution. Finally, regarding the General-Placement problem, we will see how solving special cases of a problem can help us come up with techniques that solve the more general problem.

3.2.1 Preliminaries

In this problem we are given a network $G = (V, E)$ with a distance metric on its edges, a set of demands and a set of possible location of caches. There are k different cache types with σ_i miss rate for each cash type i. A cache is fully specified by a tuple $\langle u, y, i \rangle$ where u is the location, y the capacity and i the type of the cache. The cost of each cache is given by $f_{u,i}^{y}$. In a feasible solution of the problem each demand is served by a cache of level 1. Each cache of level $1 \leq i < k$ will receive an amount of demand W and will have to route $\sigma_i \cdot W$ demand to a cache of level $i + 1$. We assume that $\sigma_k = 0$. The cost of a solution is the sum of the routing/service cost and the total cost of the open caches.

3.2.2 Simple Placement

In this version of the problem we assume that the cost of a cache is independent of its location. Also caches have no capacity in terms of the amount of the demand they can accommodate.

3.2.2.1 The Algorithm

We first create a new set of caches with the following procedure. We find the first j_1 such that $\prod_{i=1}^{j_1} \sigma_i < a$ for a given constant $0 < a < 1$. We create a new cache by combining all caches of type from 1 to j_1. This means that we open caches of type 1 through j_1 which virtually creates a new cache with miss rate $\sigma'_1 = \prod_{i=1}^{j_1} \sigma_i$. In the same manner we find j_2 such that $\prod_{i=j_1+1}^{j_2} \sigma_i < a$ and create a cache with miss rate σ'_2 and so on.

We will only use the set of new caches we have created. We will repeatedly solve an uncapacitated facility location problem for each of the levels. In each level i facility location problem we use as facility costs the costs of caches of type i, we scale the distance by a factor of a^{i-1} and leave the demands in their original location. Lets call P_i the set of facilities opened at level i. Each demand is assigned to the facility it uses in P_i. In our solution we buy caches of type i in each one of the P_i locations and route each demand first to its corresponding cache in P_1, then P_2, until it reaches P_k.

3.2.2.2 The Analysis

Theorem 3.2.1. *Given an instance of the Simple-Placement problem there exists a solution that only uses caches of capacity σ'_i with at most a $1/a$ blow up in routing costs.*

Consider the optimal solution of the Simple-Placements problem. We can open caches of miss rate σ'_k on the nodes where caches of type j_k were opened in the optimal solution and discard all caches opened in the optimal solution. We let the demands follow the same route they used in the optimal solution. Because $\prod_{i=j_i+1}^{j_{i+1}-1} \sigma_i > a$ we have that each flow is at most $1/a$ times the corresponding one of the optimal solution. Thus the theorem is proved.

Lets call X_i the routing cost of the facility location problem solved by our algorithm in level i. Also let F_i be the total cost of caches of type i in our solution and let S_i be the the routing cost from caches of type $i-1$ to type i in our solution. Also let F_i^*, S_i^* be the relative metrics of the optimal solution. Also suppose we have in our possession an r-approximate algorithm for the Uncapacitated Facility Location problem.

Lemma 3.2.2. $S_i \leq X_i + a \cdot X_{i-1}$

This lemma holds because we can always send a demand from a cache back to its original point and then from there to the next level cache.

Lemma 3.2.3. $X_i + F_i \leq r \cdot \left(\sum_{j=0}^{i-1} a^j \cdot S_{i-j}^* + F_i^* \right)$

This lemma holds because one feasible solution of the facility location problem solved at level i is $P_i = F_i^*$. A feasible routing is the path that the optimal solution uses which will have cost at most $\sum_{j=0}^{i-1} a^j \cdot S_{i-j}^*$.

Theorem 3.2.4. $S + F \leq r \cdot \left(\frac{1+a}{1-a} \cdot S^* + F^*\right)$

This theorem is an immediate outcome of lemma 3.2.3 and concludes the analysis of this algorithm.

3.2.3 Multi-Level Facility Location

In this version of the problem we assume that $\sigma_i = 1$ for all $1 \leq i < k$. Again $\sigma_k = 0$. However in this version the cost of a cache depends on its location. Let f_{ji} be the cost of placing a facility of level i at j.

A 3-approximate algorithm for this problem was obtained by Aardal et al. [1] however the algorithm is not combinatorial. Although Guha's et al. [54] has a worst approximation guarantee, it provides an interesting insight to the problem that helps in solving the General-Placement problem.

3.2.3.1 The Algorithm

We will create an instance of capacitated facility location with soft capacities (i.e. we are able to open multiple facilities at one location). In this new problem we perform the following procedure to determine the cost of a facility $\langle u, y \rangle$ where u is the node of the facility and y is the capacity.

- We construct a directed graph Q which is comprised of $k + 1$ copies of our original graph G. Lets call $G_0, G_1, ...G_k$ those copies. From a node u in G_{i-1} we add an arc to w of G_i with cost $y \cdot c_{uw} + f_{wi}$ for $1 \leq i \leq k$.
- For each node $u \in G_0$ we find its shortest path to its copy in G_k and we denote the distance of this path as R_u^y.
- We set the cost of $\langle u, y \rangle$ equal to R_u^y.

We solve the capacitated facility location problem we constructed, using the best r-approximate algorithm we know. Suppose this new problem returns a set F' of facilities. Then for each facility $f \in F'$ we open facilities in our original problem in the positions dictated by the round-trip of node f in Q. We also send each demand served by facility f to that node and then through the specific round-trip which passes through facilities of all levels.

3.2.3.2 The Analysis

Recall that we still do not have capacities in the problem we want to solve. Suppose we have an optimal solution of the Multi-Level facility location problem. We will show that we can transform this solution to a solution of the capacitated instance we created without increasing to much the cost. Suppose we have a node u that has a facility of level k and the demand p that sends its demand through the

fastest route to u in the optimal solution. Also lets call S the set of all demands that end up in u in the optimal solution. In our new capacitated facility problem we can send all demands of S to u and open at u a facility with capacity $y = \sum_{s \in S} d_s$ where d_s is the amount of demand of s. Sending all demands S to u has a routing cost that is at most the routing cost of the optimal solution. We need to bound R_u^y. A feasible round-trip of R_u^y could be to go to p directly and then from p go to u through the rout and the facilities that p passes in the optimal solution. Thus the facility cost of R_u^y is a cost that we get from a subset of facilities that S use and the routing cost is at most two times the routing cost of demands in S in the optimal solution because as we stated earlier p is the demand that reaches u the quickest in the optimal solution. Thus our total cost is at most three time the cost of the optimal solution. However since we are not able to compute all possible R_u^x round-trips in polynomial time we demand capacities to be a power of $1 + \epsilon$ thus loosing an $1 + \epsilon$ factor in the approximation. Adding the fact that we use an r-approximate algorithm for the capacitated facility location problem we end up with the following theorem.

Theorem 3.2.5. *The algorithm described is an $3 \cdot (1+\epsilon) \cdot r$ approximate algorithm for the Multi-Level facility location problem.*

3.2.4 General Placement

To solve the general placement problem we just combine the ideas analyzed for the Simple-Placement and the Multi-Level facility location problem. That combination of ideas yields the following theorem.

Theorem 3.2.6. *Given an instance of the General-Placement problem with miss rates $\sigma_1, \sigma_2, ..., \sigma_k$ we can reduce the problem to a new one with $k' \leq k$ types of caches with miss rates $\sigma'_1, ..., \sigma'_k$ such that $\sigma'_i < a$ where $0 < a < 1$, in time polynomial in input size and $\frac{1}{\epsilon}$ and at most a $\frac{3 \cdot (1+\epsilon)}{a}$ blowup in cost.*

For a detailed proof consult [54]. We then use the same algorithm as in 3.2.2 for small miss rates which leads to an $O(1)$ approximation algorithm.

3.3 Load Balanced Facility Location

In this section we will use the definitions and notations of Guha et al. [54]. Also the procedures and ideas we will analyze were also proposed in that paper. We will see how this problem can be solved by using an algorithm of the well studied uncapacitated facility location problem as a black-box. Instead of restraining the amount of demand each facility must get, we add to the cost of each facility an amount that is related to its distance from its closest demands. This will, in essence, ensure that each facility that is opened has a lot of demand "near by"

and thus will receive a satisfactory amount of demand in the outcome. In that sense we can draw an analogy between this technique and Lagrangian Relaxation, discussed earlier, since both techniques remove a constraint and introduce an extra cost that enforces that constraint. Then both techniques solve the already known problem without the constraints and end up with an approximation for the original problem.

3.3.1 Preliminaries

Suppose we have a network $G = (V, E)$. For each pair $i, j \in V$ we have a distance metric c_{ij}. Also at each node of the graph $i \in V$ we can open a facility with cost f_i. Additionally there exists a set of nodes that have demands on them of weight d_j. Finally each facility, when opened, must serve at least L_i amount of demand. The natural LP formulation of the problem is the following.

$$\min \quad \sum_i \sum_j d_j \cdot c_{ij} \cdot x_{ij} + \sum_i f_i \cdot y_i \tag{3.1}$$

$$\text{s.t.} \quad \sum_i x_{ij} \geq 1 \qquad \forall j \tag{3.2}$$

$$x_{ij} \leq y_i \qquad \forall i, j \tag{3.3}$$

$$\sum_j d_j \cdot x_{ij} \geq L_i \cdot y_i \qquad \forall i \tag{3.4}$$

$$x_{ij}, y_i \in \{0, 1\} \qquad \forall i, j \tag{3.5}$$

We will call an (a, b) approximation of this problem a solution that has at most a times the cost of the optimal solution and every opened facility serves at least $\frac{L_i}{b}$ demand.

3.3.2 The Algorithm

The algorithm is extremely simple. We create an instance of the traditional facility location problem with the same distances as in the original problem. Each facility in the new problem has a cost f'_i which is the cost f_i plus the minimum cost of routing L_i demand to this facility. We solve the corresponding uncapacitated facility location which opens a set of facilities F'. We close all facilities $i \in F'$ that receive less than $\frac{L_i}{3}$ demand and reroute the demands that are left without a facility to their closest facility.

3.3.3 The Analysis

Suppose we have an r-approximate algorithm for the uncapacitated facility location problem.

Theorem 3.3.1. *The above procedure describes an algorithm that returns a* $(2 \cdot r, 3)$ *approximate solution for the load balance facility location problem where each demand is served by its closest open facility.*

First of all we will show that any feasible fractional solution of the load balanced facility location with cost C has a corresponding feasible solution in new problem we constructed with cost at most $2 * C$. This proof is fairly simple since at any feasible solution of the load balanced facility location the open facilities serve at least L_i demand thus the total extra cost we have added on the facilities is less than the routing cost of the solution to the original problem which in turn is less than C. Thus the total cost of the exact same routing in the new problem is at most $2 \cdot C$.

We will also show that in the flow dictated by the solution of the new problem if we remove any facility i that serves less than $\frac{L_i}{3}$ demand the total cost will not be augmented. Suppose we have such a facility i and we have the closest demand point j which does not send demand to i. We will call the distance between those two nodes $c_{ij} = D$. If j is served by i' that means that $c_{i'j} \leq D$ since uncapacitated facility location solved had the same distances with our original problem. Obviously it holds that $f'_i \geq \frac{2 \cdot L_i}{3} D$. This follows from the fact that f'_i entails the cost of serving L_i demand and the distance of at least $\frac{2 \cdot L_i}{3}$ amount of that demand is greater than D. So we can close facility i move its demand to j and then to i' covering an extra distance of at most $2 \cdot D$. So the extra routing cost is at most $\frac{2 \cdot L_i}{3}$ which is at most the amount we saved by closing facility i.

This procedure can be generalized to an $(\frac{1+a}{1-a} \cdot r, \frac{1}{a})$-approximate algorithm for this problem with $a < 1$.

3.4 Connected Facility Location

In this section we will analyze our first Probabilistic algorithm. Randomness is a powerful tool in our algorithmic toolbox simplifying the analysis of many problems. Also, this problem entails two different costs (the Steiner cost and the connection cost) and thus it is of great importance for our main problem. Those two costs can be related through a parameter M. The bigger M is, the more costly the Steiner cost becomes. This relationship is captured in the algorithm. M serves as a hyper-parameter in the random procedures and the bigger its value, less facilities are open on expectation, diminishing this way the Steiner Cost and giving more emphases to the Connection cost.

The first to provide a constant factor approximation were Karger et al. [68]. Their guarantees were improved by Gupta et al. [57] who used an LP-rounding

technique. Swamy et al. [90] using a primal-dual technique decreased the approximation ratio down to $3 + p_{st}$ where p_{st} is the best approximation ratio for the Steiner tree problem. Using the algorithm of Byrka et al. [17] this yields a 4.39-approximation algorithm.

3.4.1 Preliminaries

We follow the definitions and notations of Gupta et al. [58] who were the first to propose the ideas analyzed in this section. In the Connected Facility Location Problem we are given as an input an undirected graph G with a set of vertices V and a set of edges E. We are also given for each $e \in E$ costs $c_e \geq 0$, a set of demands $D \subseteq V$ where each demand $j \in D$ is associated with a weight $d_j \geq 0$ and a parameter $M > 1$. A feasible solution to this problem consists of a set of facilities $F \subseteq V$, a matching of each demand j to an open facility and a tree T spanning over F. If we denote with $i(j) \in F$ the facility that demand j is assigned then an optimal solution minimizes the following expression:

$$\sum_{j \in D} d_j \cdot l(j, i(j)) + M \cdot c(T)$$

where l(.,.) denotes the shortest path distance with respect to the costs c_e and $c(T) = \sum_{e \in T} c_e$. We will call the first part of the cost, the connection cost and the second, the Steiner cost. We will make also the assumption that we know a vertex $r \in V$ that is used as a facility from the optimal solution and that $d_j = 1$ for each $j \in D$. The derived algorithm can easily be generalized for instances of the problem where those assumption do not hold. Finally, let C^*, S^* be the connection cost and the Steiner cost of the optimal solution OPT, $Z^* = C^* + S^*$ and F^* be the facilities of OPT and T^* be the Steiner Tree of OPT.

We will now analyze the algorithm provided by Gupta et al. [58] that provides a simple $2 + p_{ST}$-approximation for the Connected Facility Location Problem.

3.4.2 The Algorithm

The following algorithm selects randomly a subset of demands to be the facilities and then proceeds to create a Steiner tree on them and connect the rest demands. More formally:

1. We initialize a set $D' = \varnothing$.
2. For each demand $j \in D$ we add j to D' with probability $\frac{1}{M}$ and we mark it as a facility.
3. We construct a p_{ST}-approximate Steiner tree on $F = D' \cup \{r\}$.
4. We finally assign each demand to its closest facility in F.

The algorithm returns F as the facilities, the Steiner tree computed in step 3 as T and the assignment of step 4.

3.4.3 The Analysis

For the purposes of this thesis we are only going to provide the outline of the proofs. For the complete proofs the reader can consult [58].

Lemma 3.4.1. *The expected cost of step 3 is at most* $p_{ST} \cdot Z^*$.

We can construct a Steiner tree by using the optimal Steiner Tree T^* and the shortest paths from the facilities in F to T^*. Since each facility is chosen with probability $\frac{1}{M}$ the expected cost of this Steiner Tree must be at most $S^*+C^* = Z^*$, so since there is a Steiner tree with expected cost at most Z^* then the expected cost of the optimal Steiner tree must be at most Z^* which proves the Lemma.

Lemma 3.4.2. *The expected cost of step 4 is at most* $2 \cdot Z^*$.

To prove this Lemma we view the random facility selection problem from an equivalent but slightly different angle. It is important to observe the fact that the cost of the Steiner Tree is independent from the connection cost when the set of facilities F is selected. The above gives us the freedom to analyze the connection cost with a different algorithm for computing the Steiner tree. We examine the following theoretical course of actions:

1. We initialize sets $A_1 = B_1 = \{r\}$ and $t = 1$.
2. For each $j \in D$:
   (a) We set $t \leftarrow t+1$, $A_t = A_{t-1} \cup j$ and with probability $\frac{1}{M}$ we mark j and set $B_t = B_{t-1} \cup j$ and with probability $\frac{M-1}{M}$ we set $B_t = B_{t-1}$.
   (b) If j is marked then we connect it to the nearest vertex of B_{t-1} through the shortest path with cost $l(j, B_{t-1})$.

The set B_t represents the set of facilities selected until step t. The Steiner tree that is created in an equivalent manner that the MST heuristic would create that has a known 2-approximation guarantee. So the Steiner tree created has an expected cost of at most $2 \cdot Z^*$. We then consider the random variable X_t which is the connection cost minus the Steiner cost incurred at step t. We have that $E[X_t] = (1 - \frac{1}{M}) \cdot l(j, B_t) - \frac{1}{M} \cdot M \cdot l(j, B_t) = -\frac{1}{M} \cdot l(j, B_t) \leq 0$. And by linearity of expectations we have that $E\left[\sum_i X_i\right] \leq 0$. Thus the connection cost is less than the Steiner cost in expectation which concludes the proof.

Theorem 3.4.3. *The process analyzed is a* $(2+p_{ST})$*-approximation algorithm for the Connected Facility Location problem*

Where p_{ST} is the approximation guarantee of the best know Steiner Tree algorithm ([17] provides a 1.39 approximation).

3.5 Universal Facility Location

The Universal Facility Location was one of the first problems that we tried to analyze its techniques because of its similarities with our own problem. In this problem we have constant routing costs and facility costs that depend on the amount of demand they accommodate. In our problem we have the inverse relation, constant facility costs and routing costs that are susceptible to congestion.

In this section we will analyze local search algorithms. In those kind of algorithms we provide a set of local steps. The algorithm begins from a feasible state of the problem and applies local steps in order to arrive at a feasible solution of a smaller cost. When no local steps can be applied to reduce the cost we say that we have arrived to a local minimum. The main analysis of these kind of algorithms is concentrated on showing that a local minimum is not much greater than the global minimum. In order to do so we use the fact that no local step can improve the cost of a local minimum state. Applying this observation in a series of instances that implicate both the optimal solution and the local minimum state we end up with a series of inequalities that in turn provide our approximation guarantees.

In this section we will provide an 8-, a 7- and a 5- approximation algorithm for this problem. Vygen [92] was able to provide a 6.702-approximation algorithm for the problem and Angel et al. [7] a 5.83. Local search algorithms have been extensively examined in the field of Facility Location problems. This line of work can be traced back to Korupolu et al. [73].

3.5.1 Preliminaries

We follow the definitions and notations of Mahdian et al. [77]. In an instance of the University Facility Location Problem we are given as an input a graph G with two sets of nodes. The demands D and the facilities F. Each demand has a weight d_j for $j \in D$. The edges on G have a distance metric c associated with them. A feasible solution of this problems consists of a set of capacities u_i for each facility $i \in F$ and an assignment of demands to facilities in a way that does not violate those capacities. A solution S of this problem has two different costs. The facility cost $c_f(S)$ and the service cost $c_s(S)$. The facility cost is the cost paid for allocating u_i capacity at a certain facility and the service cost is the cost of routing each demand to a facility. More formally suppose a solution S consist of the tuple (u, x) where u is a vector that indicates the capacities allocated at each facility and x is a vector that indicates the amount of weight that each demand sends to each facility. The equivalent non-linear optimization problem is the following:

$$\min \quad \sum_{i \in F} f_i(u_i) \quad + \quad \sum_{i \in F, j \in D} c_{ij} \cdot x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i \in F} x_{ij} \quad = \quad d_j \qquad \forall j \in D \tag{3.2}$$

$$\sum_{j \in D} x_{ij} \quad \leq \quad u_i \qquad \forall i \in F \tag{3.3}$$

$$u_i, x_{ij} \quad \geq \quad 0 \qquad \forall i \in F, \forall j \in D \tag{3.4}$$

We will also assume that functions f_i are non-decreasing and left continuous mapping from non-negatives to non negatives. This assumption is necessary in order to assume that an optimal solution exists.

3.5.2 Locality Gap 8

The Algorithm analyzed in this section is a local search algorithm that was introduced by Mahdian et al. [77]. This means that we provide a set of improvement moves. The algorithm begins form an arbitrary feasible solution and then applies a series of improvement moves. When we reach a state that its cost can not be significantly dropped by an improvement move then he have reached the end of the process and we output the final state as the solution provided by the algorithm. There are two major steps in proving the efficiency of such algorithms. The first is to prove that the algorithm will end in a polynomial amount of steps. The second is to prove that when no significant improvement move exists (i.e. we have reached an approximate local optimal state) then we have reached a feasible solution that is within a multiplicative factor from the optimal solution. This is exactly what we are going to do in the following subsections.

3.5.2.1 The Algorithm

Suppose S is the state of a given step of the local search algorithm. We will require that the next step must improve cost $c(S) = c_f(S) + c_s(S)$ by at least $\frac{\epsilon}{5 \cdot n} \cdot c(S)$ where $\epsilon > 0$ is a predetermined constant. Since for any S, $c(S) \geq c(S^*)$ (where S^* is the optimal solution), our algorithm will terminate at most after $\frac{5 \cdot n}{\epsilon} \cdot \ln\left(\frac{c(S)}{c(S^*)}\right)$ steps.

Although finding the best local operation is NP-hard we will find at each step a local operation that is at most an additive factor of $\frac{\epsilon}{10 \cdot n} \cdot c(S)$ from the optimal step.

We have two types of local transformations (i.e. improvement steps):

- **add**(s, δ). During this operation we set $u_s \leftarrow u_s + \delta$ and find the new minimum assignment of demands to facilities. The total cost will change by

$f_s(u_s + \delta) - f_s(u_s) + c_s(S') - c_s(S)$, where S' is the new state reached due to this operation and S was the previous state.

- **pivot**(s, Δ). During this operation we adjust the capacities of each facility i by Δ_i and redistribute those flows through a pivot node s. More formally, each facility i with $\Delta_i < 0$ sends its excess demand to s. Then we distribute the gathered demand to facilities with $\Delta_i > 0$. It is obvious that $\sum_i \Delta_i = 0$ during this operation. The total cost will change by $\sum_{i \in F}[f_i(u_i + \Delta_i) - f_i(u_i) + c_{si} \cdot |\Delta_i|]$ where c_{si} is the shortest path from s to i with respect to the distance metric c.

To prove that we can compute an admissible local step (i.e. a step with cost at most $-\frac{\epsilon}{5 \cdot n} \cdot c(S)$) we will discretize the problem by assuming that d_j are small integers and that the cost functions f_i are step functions with steps occurring at integer points. These assumptions do not come without loss of generality. Those assumptions are made for convenience and to simplify the underlying processes. At the end of this section we are going to analyze how this restrictions can be waived.

Finding the best **add** operation is straightforward. There are $|d| = \sum_j d_j$ possibilities for δ and $|F|$ possibilities for s. Thus we can try all different possibilities.

Lemma 3.5.1. *Given a solution (u, x) and a point s we can find the minimum cost* **pivot** *operation in time polynomial in the number of facilities $n = |F|$ and the total demand $|d|$.*

The proof of this Lemma is a dynamic programming algorithm that computes the optimal **pivot** operation. We first assume that in our solution (u, x) the capacities are exactly equal to the amount of flow they serve. Let $g_i(\delta) = c_{si} \cdot \delta + f_i(u_i + \delta) - f_i(u_i)$. g_i denotes the contribution of facility i to the cost of the **pivot** operation given that we adjust u_i by δ (note δ can be either positive or negative). We build the table $a(i, w)$, which can be interpreted as the minimum cost of having w excess units of demands on s after considering facilities 1 through i:

$$a(i, w) = \begin{cases} g_1(-w) & i = 1 \\ \min_\delta\{g_i(\delta) + a(i-1, w+\delta)\} & i = 2, ..., n \end{cases}$$

The best **pivot** operation can be found at $a(n, 0)$. We repeat this operation for every possible pivot s.

Finally, instead of discretizing over demands, we can create a new metric $B_i = f_i(u_i)$ which we will call the budget of facility i. We can now discretize over the budgets by assuming they are multiples of $\frac{\epsilon}{n} \cdot c(S)$. This adds an extra factor of ϵ in the approximation guarantee.

3.5.2.2 The Analysis

For the rest of the analysis we shall assume that using the aforementioned process we have reached an approximate local optimal solution $S = (u, x)$ and we will try to bound its cost with respect to the cost of the optimal solution S^*.

First of all it is fairly easy to bound the service cost.

Lemma 3.5.2. *The service cost of a locally optimal solution S is at most the total cost of the optimal solution S^**

We can prove this lemma by contradiction. Suppose $c_s(S) > c(S^*)$. If we increase the capacity of each facility i to $\max(u_i, u_i^*)$ (paying at most an extra amount of $c_f(S^*)$ on facility costs), the assignment vector of the optimal solution becomes feasible. Thus the new cost shall be $c(S) - c_s(S) + c_s(S^*) + c_f(S^*) = c(S) - c_s(S) + c(S^*) < c(S)$. Using the submodularity of the **add** operation there exists an add operation that decreases the total cost of S, thus S is not a local optimal solution which concludes the contradiction.

The most difficult task in this analysis is bounding the facility cost. The way we are going to bound this cost is the following. We are going to state a series of pivot operations. Since we are at a locally optimal solution S the cost of each pivot operation must be non negative because otherwise we would have a step that would decrease the total cost of S contradicting this way the local optimality of that state.

We are now going to state **"The Transshipment Problem"**, the solution of which will serve as a bases for our proof. Suppose we have a locally optimal solution $S = (x, u)$ and an optimal solution $S^* = (x, u)$. Without loss of generality we will assume that $u_i = \sum_j x_{ji}$ and $u_i^* = \sum_j x_{ji}^*$. Also let $\delta_i = u_i - u_i^*$. We define the set of sources $U = \{i \in F | \delta_i > 0\}$ and sinks $U^* = \{j \in F | \delta_j < 0\}$. The transshipment problem is simply finding the optimal flows (with respect to delays c) from sources to sinks where each source $i \in U$ is emanating δ_i flow and each sink j is absorbing δ_j flow.

We can now state the following (obvious) Lemma.

Lemma 3.5.3. *The Transshipment problem has a cost of at most $c_s(S) + c_s(S^*)$.*

Obviously a feasible solution to the transshipment problem is routing all demands back to their original sources with cost at most $c_s(S)$ and then to the sinks with cost at most $c_s(S^*)$.

Let y be the optimal flow for the transshipment problem. We will call the subgraph produced by y **"The Exchange Forest"**. Note that if adding flow at one direction of any circle in y increases the cost then adding flow on the other direction would decrease it, contradicting the optimality of y. Thus we are able for any circle to add flow in any direction until the flow on one edge disappears breaking this way the circle. So without loss of generality y has no circles. Also there are no direct flows from source to source or from sink to sink. From triangle

inequality we can always connect sources and sinks directly. The above lead us to the following conclusion: y is a forest of bipartite trees which consist of alternating U and U^* vertices.

We are now ready to state a list of pivot operations on the exchange graph. For simplicity we will say that we close a source $i \in U$ when we decrease its capacity to u_i^* and that we open a sink $j \in U^*$ when we increase its capacity to u_j^*. It is important to remember that we are still analyzing the case where we are at a local optimal solution S.

Let r be the root of a tree T of the exchange forest. For every sink $t \in U^*$, let T_t be the sub-tree of the exchange graph that is rooted at t and consists of t children and grandchildren. Obviously T_t has a maximum height of 2. Also let for any vertex $x \in T_t$, $C(x)$ be the children of x in T_t.

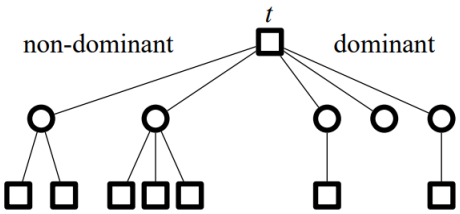


Figure 3.1: The sub-tree T_t where circles denote sources and squares sources (from [77], p. 9)

We are going to divide vertexes of $C(t)$ into two distinct groups, the Dominant and the Non-Dominant. Dominant is a sources s that sends at least half of its flow at t in y (i.e. $y_{st} \geq \frac{1}{2} \cdot \sum_{t' \in U} y_{st'}$). The rest of the sources are Non-Dominant.

For the case of the Dominant sources we have the following pivot operation. We set as our pivot node t, we open t and all the children of Dominant nodes and we close all Dominant nodes. Note that during this pivot operation each edge is used at most 3 times (i.e. by at most 3 times the amount of flow sent at y through that edge).

For the Non-Dominant nodes the pivots are a little bit more complicated. Suppose we order non dominant nodes as $s_1, s_2, ..., s_k$ where $y_{s_1 t} \leq y_{s_2 t} \leq ... \leq y_{s_k t}$. For every $i \in \{1, 2, ..., k-1\}$ we will close s_i and open $C(s_i) \cup C(s_{i+1})$. For s_k we close that node and open t and $C(s_k)$. Note that due to the ordering of Non-Dominant nodes each edge is used at most once (i.e. receives no more flow than the flow received at y).

Since we are at a local optimal state S it should hold that for every pivot operation mentioned earlier it should hold that:

$$\sum_{i \in A} (f_i(u_i^*) - f_i(u_i)) + \sum_e c_e \cdot \delta_e \geq 0$$

Where A is the set of opened and closed facilities.

We now sum over all these inequalities. Remember that through the pivot operations described:

- Each source U is closed exactly once.
- Each sink U^* is opened at most by 4 operations. Once through its Dominant children, once by its Non-Dominant children, and at the subtree of its grandparent either one time as part of $C(Dom)$ either at most two times as part of $C(Non - Dom)$.
- For each edge e the total amount of demand shipped is at most 3 time the amount of demand at y.

Thus redistributing the summed inequalities and using the fact that the cost of y is at most $c_s(S) + c_s(S^*)$ we get that $4 \cdot c_f(S^*) - c_f(S) + 3 \cdot (c_s(S) + c_s(S^*)) \geq 0 \Rightarrow c_f(S) \leq 4 \cdot c_f(S^*) + 3 \cdot (c_s(S) + c_s(S^*))$.

Theorem 3.5.4. *Let S^* be an optimal solution and S be a locally optimal solution produced by the operations described by the algorithm. It holds that $c_s(S) \leq c_s(S^*) + c_f(S^*)$ and $c_f(S) \leq 6 \cdot c_s(S^*) + 7 \cdot c_f(S^*)$*

3.5.3 Locality Gap 7

Pandit [80] noticed that a simple improvement of the proposed by Mahdian et al. can decrease the approximation guarantee to 7.

On top of the possible operations **add** and **pivot** he adds a new operation called **Doublepivot**$(s_1, s_2, \Delta^1, \Delta^2)$ which is basically two pivots **pivot**(s_1, Δ^1) and **pivot**(s_2, Δ^2) done simultaneously.

Then he uses the exact same analysis as described in 3.5.2.2. The main difference is that instead of closing s_k of Non-Dominant nodes separately, he uses the **Doublepivot** operation to close together with the Dominant nodes and thus opening t one time less. Thus when summing over all pivot inequalities he ends up with the following bound, $c_f(S) \leq 3 \cdot c_f(S^*) + 3 \cdot (c_s(S) + c_s(S^*))$ which in turn yields the following Theorem:

Theorem 3.5.5. *Let S^* be an optimal solution and S be a locally optimal solution produced by the operations described in this section. It holds that $c_s(S) \leq c_s(S^*) + c_f(S^*)$ and $c_f(S) \leq 6 \cdot c_s(S^*) + 6 \cdot c_f(S^*)$*

3.5.4 Locality Gap 5

Using a different set of local search operations Bansal et al. [12] were able the approximation down to a factor of 5. The outline of the analysis is analogous to the one stated in 3.5.2. A set of local procedures is listed and then using the

Exchange Graph and due to the local optimality of the last state we can introduce a set of inequalities that lead as to the approximation guarantee of the algorithm.

Bansal et al [12] introduce three local search operations:

- **add**(s, δ): In this operation we increase the capacity of facility s by an amount $\delta > 0$ and then a mincost flow problem is solved to find the new optimal demand allocations.

- **open**(t, δ_1, δ_2): This operation can be described as two concurrent operations. In the one operation we increase the capacity allocated at t by δ_2 and at the same time the total capacity of a set T of facilities is decreased by the same amount. The optimal set T is computed by the operation itself. The other operations is **add**$(t, \delta_1 - \delta_2)$.

- **close**(t, δ_1, t^*): This operation can be also described as two concurrent operations. The one operation is **add**$(t^*, \delta_2 - \delta_1)$ where $\delta_2 > \delta_1$ is a constant to be computed by the operation. In the other operation we decrease the capacity allocated at t by δ_1 and at the same time the total capacity of a set T (where $t^* \in T$) of facilities is increased by the same amount.

The above three operations are computable in polynomial time. Again using the Exchange Graph we can prove that $c(S) < 5 \cdot c(S^*)$. Since the techniques used by Bansal et al [12] do not differ that much from the underlying logic of the proofs stated in 3.5.2 we will not present them here.

3.5.5 Reductions

The Universal Facility Location problem is really important because it generalizes many different facility location problems.

3.5.5.1 Uncapacitated Facility Location

In this problem each facility has a facility cost irrelevant from the amount of demand it receives. Using $f_i(u) = f'_i \cdot [u > 0]$ where f'_i is the facility cost and $[u > 0]$ is the step function, we can model this problem.

3.5.5.2 Capacitated Facility Location with soft capacities

In this facility location problem each facility has a maximum capacity u'_i however we can open multiple copies of a facility in one location. It is obvious that using $f_i(u) = f'_i \cdot \lceil \frac{u}{u'_i} \rceil$ models this problem.

3.5.5.3 Capacitated Facility Location with hard capacities

In this problem again each facility has a maximum capacity u'_i however now we can open only one facility in each location. If we use $f_i(u) = f'_i[u > 0] + \infty \cdot [u > u'_i]$ it becomes evident that we can also model this problem.

Chapter 4

Network Design Problems

In the past few decades many different networks such as the Internet and Communication Networks have increased in size and complexity, creating many new challenging problems that, if solved optimally, can make a significant impact in the efficiency of these systems. In this section we will analyze such interesting and challenging problems. The class of Network Design Problems usually takes as input a network (commonly modeled with graphs) and given a set of costs and constraints, an algorithm will try to find a sub-network or a flow that satisfies the constraints of the problem and minimizes the relative cost. Given that the problems we are going to analyze are NP-hard it would be incredibly difficult (probably impossible) to compute an optimal solution. Thus we are concerned with finding approximation algorithms with good approximation guaranties.

In this section we will examine algorithms that use a variety of techniques. Most of these algorithms use randomness for their analysis. We will see how building some big infrastructure on the problem with a cost relative to the one of the optimal solution can help simplify the rest of the algorithm. We will also examine techniques that step by step gather demands together in order to make use of the economies of scale provided by the problem. Finally, we will see how matching and randomized routing can help us tackle problems with multiple unrelated costs.

4.1 Virtual Private Network Design Problem

In this section we will examine an algorithm that builds some infrastructure that can accommodate all demand as the backbone of its solution. Then connecting all demands to that powerful structure helps move all demands around and send them to their final destination. This algorithm also uses probabilistic techniques that render the analysis of the problem more tractable.

The problem was introduced by Fingerhut et al. [39] and later independently by Duffield et al. [33]. Gupta et al. [57] analyzed the problem from an approximation point of view. However all constant solutions before Gupta et al. [58] were for

special cases of the problem.

4.1.1 Preliminaries

We follow the definitions and notations of Gupta et al. [58]. In the Virtual Private Network Design Problem we are given as an input an undirected network with a set of vertices V and a set of edges E. We are also given costs $c_e \geq 0$ for each $e \in E$, a set of demands $D \subseteq V$ and two thresholds $b_{in}(j), b_{out}(j)$ for each $j \in D$. A flow can be represented by a $D \times D$ matrix where d_{ij} denotes the amount of demand flowing from i to j. A flow is feasible if it obeys the given constraints (i.e. $\sum_i d_{ij} \leq b_{in}(j)$ and $\sum_j d_{ij} \leq b_{out}(i)$ for each j, i respectively). A feasible solution consists of paths P_{ij} for each demand pair i, j and capacities u_e for each $e \in E$ such that the capacities can handle any feasible flow that uses paths P_{ij} (i.e. for any feasible flow $D \times D$, $\sum_{ij:e \in P_{ij}} d_{ij} \leq u_e$). The objective is to minimize the cost of the solution that is given from the following expression:

$$\sum_e c_e \cdot u_e$$

Also we will assume that there are only two types of demands, the senders and the receivers. We denote the set of senders as S and we have that $b_{in}(j) = 0$ and $b_{out}(j) = 1$ for each $j \in S$. We also denote the set of receivers as R and we have that $b_{in}(j) = 1$ and $b_{out}(j) = 0$ for each $j \in R$. Finally we will assume that the amount of receivers is less or equal the amount of senders (i.e. $|R| \leq |S|$, note that one set can have a smaller cardinality than the other since b_{in} and b_{out} are thresholds and not actual demands). We will denote the amount of senders $|S|$ as M. The above assumptions simplify the algorithm and can easily be generalized.

We are now going to analyze the algorithm provided by Gupta et al. [58] that provides a simple 5.55-approximation for the Virtual Private Network Design. This is the first algorithm to provide a constant approximation for the problem.

4.1.2 The Algorithms

The algorithm creates a central high-bandwidth path and connects senders and receivers with this path. More formally the algorithm has the following steps:

1. We initialize $u_e = 0$ for each $e \in E$.
2. We pick a sender $s \in S$ uniformly at random from the set of senders.
3. We create a new set $R' = \varnothing$.
4. We add each $j \in R$ independently with probability $\frac{1}{M}$ to R'.
5. We then construct a p_{ST}-approximate Steiner tree T_s on $F = R' \cup \{s\}$.

6. We set $u_e = M$ for each $e \in T_s$.

7. Finally for each senders and receivers j where $j \notin T_s$ we increase the capacity of the shortest path from j to T_s by one.

4.1.3 The Analysis

We denote as T the set of edges that was assigned a non zero capacity from the algorithm. Note that T is a tree because every circle in T can be divided into two segments with the same cost. Thus circles can be avoided with a consistent tie-breaking mechanism. Also it is not difficult to see that T with the corresponding capacities is a feasible solution.

In order to prove the desired approximation guarantee three Lemmas are used to bound the costs incurred by different steps of the aforementioned algorithms. For the purpose of this thesis we will only analyze the high level rationale of the proofs. Complete proofs are provided in [58].

Lemma 4.1.1. *The expected cost of Step 5 is at most $p_{ST} \cdot Z^*$, where Z^* is the cost of the optimal solution OPT.*

First of all we can see the random process of steps 2 through 4 can be seen as an equivalent different random process. We initialize $D_s = \varnothing$ for each $s \in S$. Each receiver $r \in R$ chooses a sender s uniformly at random and we update $D_s = D_s \cup \{r\}$. We have M senders in total and so each sender is chosen with probability $\frac{1}{M}$.

Since the optimal solution specifies paths between every pair of sender receiver r, s we can isolate the appropriate paths and come up with (possibly not optimal) Steiner Trees for any set D_s. The above observation, when combined with the fact that when an edge e is part of k distinct r, s paths it must have capacity $u_e \geq k$ in order to accommodate all feasible flows, leads us to the following expression:

$$Z^* \geq \sum_s c(T_s^*)$$

where Z^* is the cost of OPT, T_s^* is the optimal Steiner Tree of D_s and $c(T) = \sum_{e \in T} c_e$. From the above expression we can see that there must be a set D_s with a Steiner Tree of cost $\frac{Z^*}{M}$ and thus when we install M capacity on this tree we incur a cost of Z^*. Using a p_{ST}-approximate algorithm for computing a Steiner Tree we end up with the guarantee that is provided from the Lemma.

Lemma 4.1.2. *The expected cost from connecting receivers to the central core is at most $2 \cdot Z^*$.*

This Lemma can be proved using the same arguments that proved Lemma 3.4.2

Lemma 4.1.3. *The expected cost from connecting senders to the central core is at most* $2 \cdot Z^*$.

For this proof we fix an arbitrary set $R' \subseteq R$ where $|R'| = M$. Any perfect matching $\mathcal{M}$ over R' and S produces a valid $D \times D$ flow. Thus we can show that $\sum_{(r,s)\in\mathcal{M}} l(r,s) \leq Z^*$. Averaging over all matchings we get that $E_{s\in S}\left[\sum_{r\in R'} l(r,s)\right] \leq Z^*$. This inequality combined with the fact that $\sum_{s'\in S} l(s,s') \leq \sum_{r\in R'} l(r,s) + sum_{(r,s')\in\mathcal{M}} l(r,s') \leq \sum_{r\in R'} l(r,s) + Z^*$ concludes the proof.

Theorem 4.1.4. *The process analyzed is an* $(4 + p_{ST})$*-approximate algorithm for the Virtual Private Network Design Problem.*

Where p_{ST} is the approximation guarantee of the best know Steiner Tree algorithm.

4.2 Access Network Design Problem

For this problem we are going to analyze the algorithm proposed by Guha et al. [54] and we will use the same notations and definitions as in that paper.

In this section we will see an algorithm that uses techniques developed for facility location problems in order to gather demands together and take advantage of the economies of scale of the problem at hand. It will also serve as a point of reference for understanding the underlying structure of the more general Single-Sink Buy at Bulk problem.

4.2.1 Preliminaries

This problem is a simpler version of the Single Sink Buy at Bulk problem that we will analyze later in this thesis. More specifically in this problem we are given a network $G = (V, E)$ with a length function on the edges and a sink node $s \in S$. There are k types of pipes that have a fixed ϕ_k cost per edge length and an incremental cost δ_k per unit of demand per unit of length. Thus the cost per unit length of a pipe type k that is being used by d demand is $\phi_k + d \cdot \delta_k$. Also without loss of generality we assume that $\phi_1 < \phi_2 < ... < \phi_k$ and $\delta_1 > \delta_2 > ... > \phi_k$. If this constraint does not hold, then it must be the case that some pipe types are always better than other pipe types giving us the possibility to discard those inefficient pipe types and end up with an instance where this condition holds.

The problem that we have described until now is an equivalent problem to the Single Sink Buy at Bulk within a factor of two. However for the Access Network Design Problem we actually have three more restrictions that make the problem a special case of the Single Sink Buy at Bulk. Those restrictions are the following:

1. For $2 \leq k \leq K$ we have that if $d < \frac{\phi_k}{\delta_k}$ then $d \cdot \delta_{k-1} + \phi_{k-1} < d \cdot \delta_k + \phi_k$

2. $d \cdot \delta_1 > \phi_1$

3. $\sum_{i<k} \phi_i = O(\phi_k)$

4.2.2 The Main Idea

Andrews et al. [6] proved that there exists a near optimal solution which has the following properties:

- It is a tree.
- Each demand is routed through demands of consecutive type.
- Any pipe of type k has at least $u_k = \frac{\phi_k}{\delta_k}$ of flow.

Through this observation it becomes more apparent that there is a strong connection between this problem and the Simple Placement problem that we analyzed in section 3.2.2. The main difference is that in this problem a pipe of type k becomes profitable when u_k of demand is using it. This is where the ideas, analyzed in section 3.3 about the Load Balanced Facility Location problem, come into play.

First of all we can prove an analogous to 3.2.1 theorem which states that there exists a solution to our problem that uses only pipes where $\sigma_i = \frac{\delta_i}{\delta_{i-1}} \leq a$ where a pipe of type i has always at least u_i amount of flow and has a maximum $1/a$ blow up in cost.

The algorithm is straight forward and is a combination of ideas presented in 3.2.2 and 3.3. For each type of pipe i we solve a Load Balanced Facility Location problem. In this new problem we have the demands in their original position, the cost along an edge is δ_{i-1} times the length of that edge. The cost for each facility is zero (i.e. $f_i = 0$) and we have that for the sink s, $L_s = 0$ and for all other nodes $L_i = u_i$. We will call C_i the cost of the approximate solution to this new problem which is obtained by using an (r, γ)-approximate algorithm. Also let C^* be the cost of the optimal solution to our original problem. Then we can prove that $\sum_i C_i \leq r \cdot \frac{1}{1-a} \cdot C^*$ and that the total fixed cost of our solution is bounded by $\gamma \cdot (\sum_i C_i) \cdot (1 + a)$ and the total incremental cost is bounded by $\frac{1-a}{1-2 \cdot a} \cdot \sum_i C_i$. Putting all this together we end up with a 94.5 approximate algorithm for the Access Network Design problem.

4.3 Single Sink Buy-at-Bulk Problem

In this section we will see another Probabilistic algorithm. The main technique used here is due to a redistribution lemma that in each step of the algorithm gathers demands in order to make use of the economies of scale that govern the costs of the different pipe types.

This problems has received significant attention from the scientific community. Awerbuch et al [10] were the first to come up with a $O(log^2(n))$-approximate algorithm for the general case of the problem. Using the tree embeddings of Fakcharoenphol et al. [36] one can come up with a $O(log(n))$-approximate algorithm for the problem. Garg et al. [47] used LP-rounding techniques in order to come up with a $O(K)$-approximation algorithm for the problem. The first constant approximation algorithm was due to Guha et al. [56]. The approximation ratio was decreased down to 216 by Talwar [91].

4.3.1 Preliminaries

We follow the definitions and notations of Gupta et al. [58]. As an input for this problem we are given an undirected network with a set of vertices V and a set of edges E. We are also given a root vertex r, a set of demands D where each demand $j \in D$ wants to send d_j amount of flow to the root vertex r. Each edge $e \in E$ is associated with a length c_e. Finally, there are K types of cables ($\{1, 2, 3, .., K\}$) that can be installed in each edge with cable i having capacity u_i and cost σ_i per unit length. We will also use the auxiliary variable $\delta_i = \frac{\sigma_i}{u_i}$.

We assume that δ_i is a power of 2. This can be generalized by loosing a factor of 4 in the approximation since we just need to round each capacity u_i down to its closest power of 2, and round each cost σ_i up to its closest power of 2. Without loss of generality we can assume that $u_i < u_j$ and $\sigma_i < \sigma_j$ for each $i < j$. Note that if there is a cable i' for which there is a cable i with $u_{i'} \leq u_i$ and $\sigma_{i'} \geq \sigma_i$ we can eliminate cable i' from consideration. We can also assume that $u_1 = \sigma_1 = 1$. The cables must obey economies of scale (i.e. $\delta_k < \delta_j$ for each $j < k$) since otherwise we could eliminate cable type k. Since we have assumed that δ_i is a power of 2 that means that $\delta_{i+1} \leq \frac{\delta_i}{2}$. We define $g_k = \frac{\sigma_{k+1}}{\sigma_k} \cdot u_k$. We can easily see that $u_k < g_k < u_{k+1}$. Finally, we denote the cost of the optimal solution as $C^* = \sum_j C^*(j)$ where $C^*(j)$ is the cost of cables of type j in the optimal solution.

4.3.2 A 72.8-approximation ?

This section analyzes the algorithm provided by Gupta et al. [58]. In order to proceed with the algorithm we will first need to prove the following Redistribution Lemma:

Lemma 4.3.1 (Redistribution Lemma)**.** *Let T be a tree with edges of capacity U. Also, for each $j \in V(T)$ we have that $w(j) < U$ represents the weight located at node j with $\sum_j w(j)$ a multiple of U. Then there exists an efficiently computable (random) flow, which respects the capacities in T and redistributes the weights such that node j ends up with either U weight with probability $\frac{w(j)}{U}$ or 0 weight with probability $1 - \frac{w(j)}{U}$.*

The proof is straight forward and in short it chooses a random variable and wile it traverses the Euler tour of the augmented tree T it accumulates demand on certain nodes in order to satisfy the restraints of the Lemma.

More formally, the following process produces a redistribution with the intended outcome:

- We augment tree T by replacing each edge with two oppositely directed arcs which produces an Euler tour C on T.
- We uniformly at random select Y from $(0, U]$.
- We initialize $Q = 0$
- For each vertex j on the Euler tour beginning from the root r:
  - We set $Q \leftarrow Q + w(j)$.
  - Suppose right before reaching j Q was Q_{old} and right after it becomes Q_{new}.
  - If for some integer x, $x \cdot U + Y \in (Q_{old}, Q_{new}]$:
    - We mark j.
    - We ask $jtosend$ $Q_{new} - (x \cdot U + Y)$ demand to the next marked vertex in the Euler tour.
  - Else we ask j to send all of its demand to the next marked vertex in the Euler tour.
- Using flow cancellation we remove the directed arcs with one single edge.

Using the Redistribution Lemma we can build a p_{ST}-approximate Steiner on D using cables of capacity $u_1 = 1$ and end up with integral demands paying a cost of at most $p_{ST} \cdot \sum_j \frac{C^*(j)}{\sigma_j}$ since the optimal solution is a candidate Steiner tree. We can now assume that $d_j = 1$ and make the number of Demands $|D|$ a power of 2 by placing dummy demands on r with 0 cost.

Unfortunately, as observed by Jothi et al. [65], when using this redistribution Lemma we have no guarantee that the demands wont split into multiple parts that are eventually going to be routed through different paths to the sink. So this algorithm actually solves a variation of the Single Sink Buy at Bulk problem called Divisible Single Sink Buy at Bulk.

4.3.2.1 The Algorithm

1. We initialize $D_1 = D$ and $t = 0$.
2. While there is demand that has not been routed to r:
   (a) We set $t \leftarrow t + 1$.

(b) We mark each demand in D_t with probability $p_t = \frac{u_t}{g_t}$ and we create a new set D'_t that contains those marked demands.

(c) We construct a p_{ST}-approximate Steiner tree on the set $F_t = D'_t \cup \{r\}$ with cables of type $t+1$.

(d) Each vertex $j \in D_t$ send its demand to the nearest vertex of F_t, lets assume i, which accumulates at this point a total of $w_t(i)$ demand.

(e) Since at step t all demands are either 0 or u_t, $i \in F_t$ has received demand from $\frac{w_t(i)}{u_t}$ vertices of D_t. We divide those vertices into groups of $\frac{u_{t+1}}{u_t}$ vertices leaving $b_i = \frac{w_t(i)}{u_t} \mod \frac{u_{t+1}}{u_t}$ residual vertices without a group.

   i. We send u_{t+1} demand to a random member of the group using a cable of type $t+1$.

   ii. We use the Redistribution Lemma with $T = T_t, w_t(i) = b_i \cdot u_t$ and $U = u_{t+1}$

   iii. For every vertex $i \in F_t$ that ends up with u_{t+1} due to the redistribution we randomly send back that demand to one of i's residual vertices with cables of type $t+1$

Note that this algorithm will end since at step K, $p_K = 0$ and thus $T_K = r$ and all demands will be rooted to r through their shortest path.

4.3.2.2 The Analysis

Although at first site the algorithm seem complicated its guarantees can be explained through a series of easy Lemmas.

Lemma 4.3.2. *For every non-root vertex $j \in D$ it holds that:*

$$\mathbf{Pr}[j \in D_t] = \frac{1}{u_t}$$

This Lemma can easily be proved using Inductive reasoning.

Corollary 4.3.2.1. *For a non-root vertex $j \in D$ it holds that:*

$$\mathbf{Pr}[j \in F_t] = p_t \cdot \frac{1}{u_t} = \frac{1}{g_t}$$

Lemma 4.3.3. *Let T_t^* be the optimal Steiner tree on F_t, then:*

$$\mathbf{E}\left[\sum_{e \in T_t^*} c_e\right] \leq \sum_{s>t} \frac{1}{\sigma_s} \cdot C^*(s) + \sum_{s \leq t} \frac{1}{\delta_s \cdot g_t} \cdot C^*(s)$$

This expression can easily be shown by proving that there exists a Steiner tree spanning F_t with that cost. That Steiner tree is created by combining two sets of edges. The first one is the set of edges that have a cable type of $t+1$ or higher. This set of edges has an expected total cost of $\sum_{s>t} \frac{1}{\sigma_s} \cdot C^*(s)$. The next set of edges can be derived by examining each vertex $i \in F_t - \{r\}$. We add one of the paths that i uses in the optimal solution with probability proportional to the fraction of i's weight that this path carries. This yields an expected cost of $\sum_{s \leq t} \frac{1}{\delta_s \cdot g_t} \cdot C^*(s)$. The above sets of edges produce a Steiner tree with the desired restrictions.

Lemma 4.3.4. *The expected cost of step t is at most $(3 + p_{ST}) \cdot \sigma_{t+1} \cdot \mathbf{E}[c(T_t^*)]$, where T_t^* is the optimal Steiner tree on F_t.*

The proof of this Lemma is straight forward if we consider the fact that we can easily establish upper bounds for the costs of each step of the algorithms:

- The cost of the Steiner tree of step 2.c. is at most $p_{ST} \cdot \sigma_{t+1} \cdot c(T_t^*)$
- The cost of step 2.d. is at most $2 \cdot \sigma_{t+1} \cdot c(T_t^*)$. This can be shown using an argument analogous to Lemma 3.4.2.
- We can easily show that the cost of steps 2.e. are at most $\sigma_{t+1} \cdot c(T_t^*)$.

Theorem 4.3.5. *The process described is a 72.8-approximation algorithm for the Divisible Single Sink Buy at Bulk Problem.*

By adding the initial rounding costs and the cost incurred by the algorithm we come up with a guarantee of $16 \cdot (3 + p_{ST})$

4.3.3 A 153.6-approximation

We now analyze the algorithm provided by Jothi et al. [65]. The core idea of the algorithm is the same as the one presented by Gupta et al. [58]. The most important contribution is a revisal of the Redistribution Lemma 4.3.1 in order to ensure that demands can not be split and routed through multiple paths to the sink. Then using the revised redistribution Lemma with some appropriate hyper parameter tuning they conclude to improved results for both the Divisible and the non-divisible Single Sink Buy at Bulk Problem.

In the algorithm analyzed by Jothi et al. [65] instead of demanding δ_i to be a power of 2 (as in [58]), we demand that it is a power of $1 + \epsilon$. This simple observation lets us significantly decrease the cost of the algorithm.

4.3.3.1 The Revised Redistribution Lemma

First of all, we are going to need two auxiliary Lemmas in order to prove the Revised Redistribution Lemma.

Lemma 4.3.6. *Either there exists at least one edge with zero flow in the directed tour constructed by Lemma 4.3.1, or there exists a redistribution with zero flow on one edge of the tour that end up with the exact same assignment of weights.*

Suppose there is no edge with zero flow in the tour. We can simply subtract form each edge the smallest amount of flow that is being redistributed. Then we end up with the desired result.

Lemma 4.3.7. *For any Y chosen uniformly at random from $(0, U)$ during the procedures of Lemma 4.3.1, there exists a redistribution using the same Lemma but with $Y = U$ that ends up with the same weights.*

The key observation in order to prove this Lemma is that Lemma 4.3.1 always starts from a fixed vertex, however this dose not need to be the case. Suppose a redistribution with a random Y. From Lemma 4.3.6 we know that there exists (or we can create) a vertex q in the tour that receives zero flow. It is easy to show that if we begin the procedure of Lemma 4.3.1 from q with $Y = U$ we end up with the same weights.

We are now ready to prove the Revised Redistribution Lemma.

Lemma 4.3.8 (Revised Redistribution Lemma)**.** *Let T be a tree with edges of capacity U, which is a power of two . Also, for each $j \in V(T)$ we have that $w(j) < U$ represents the weight located at node j which is also a power of two. Then there exists an efficiently computable flow, which respects the capacities in T and without splitting the flow of a vertex that redistributes the weights such that node j ends up with either U weight with probability $\frac{w(j)}{U}$ or 0 weight with probability $1 - \frac{w(j)}{U}$.*

There are two key parts into proving this Lemma. First, we begin the procedure of the Redistribution Lemma by the vertex we found with Lemma 4.3.7 and with $Y = U$. Then whenever we mark an edge and need to send $Q_{new} - x \cdot U$ demand to the next marked vertex we can show that since all demands are a power of two then there exist a subset W of vertexes that have not been assigned to a marked vertex yet and that $Q_{new} - \sum_{i \in W} w(i) = x \cdot U$. This observation shows that whenever we mark a vertex we do not need to split any demands. The second key part is to prove that we do not need to split any demands when we replace the directed arcs of the Euler tour with the original edges of T. We can do this by applying the following procedure recursively on all leafs of T until we are left with no vertices. If a leaf is not marked then we send its demand to its parent in T and discard it. If the leaf is marked then we just send the amount of demand needed to the leaf from its parent and then discard the leaf. Notice that in both situations we use only one of the two arcs corresponding to each edge of T. Recursively this procedure gives us the desired flow.

Once we have this revised redistribution lemma we can use the same procedure introduced by Roughgarden et al [58] to derive an algorithm that solves the unsplittable version of the Single Sink Buy at Bulk problem.

4.4 Two Metric Network Design

In this section we will analyze the $O(\log k)$-approximation algorithm for the Cost-Distance problem presented by Meyerson et al [79]. This was one of the first problems that occupied our attention because of its similarities with our own main problem. In both instances we have to cope with two different type of costs. One that is related to the distance that the demands need to travel and a second that is a constant cost, payed when we "buy" an edge in order to be able to use it. The main differences of the two problems are that in our main problem that we need to buy facilities, not edges, and that distances in our instance are not constant but are related to the amount of congestion on each edge. In the algorithm of the following section, we will analyze how we can use matchings and randomized redistribution of demands in order to decrease by half the amount of demand points. This leads to a logarithmic amount of steps that in turn yields the logarithmic approximation guarantee.

This problem is of significant importance because, as it will be discussed later, it generalizes many important network design problems. In the case where the two metrics (cost and distance) are related, there are constant factor approximations due to the works of Awerbuch et al. [11] and Khuller et al. [69]. The algorithm presented borrows ideas from a similar version of the problem examined by Marthe et al. [78] and Kortsarz et al. [72], in which the goal is to find a minimun cost tree (based on a metric c), whose diameter is less than L (based on a different metric l). It was shown from Chuzhoy et al. [30] that this problem is $\Omega(loglog|S|)$ hard to approximate.

4.4.1 Preliminaries

We follow the definitions and notations of Meyerson et al. [79]. In the Cost-Distance Problem we are given a graph G with a set of vertices V and a set of edges E. We are also given a set $S \subset V$ of demands that need to be routed to a single sink vertex $t \in V$. For each demand $s \in S$ we have a weight $w(s)$. For each edge e we have a cost $c(e)$ and a distance $l(e)$ metric. A feasible solution consists of a subgraph $G' = (E', V') \subset G$ for which $V' \subseteq S \cup \{r\}$. An optimal solution minimizes the sum of costs of edges of E' and the distances of the demands to the sink. More formally an optimal solution minimizes the following expression:

$$\sum_{e \in E'} c(e) + \sum_{s \in S} w(s) \cdot L'(s,t)$$

where L'(.,.) is the total distance of the two vertices with respect to the distance metric l on graph G'. We will address the total cost as the total value in order to avoid ambiguity in the analysis since we already have a cost metric.

4.4.2 The Algorithms

This algorithm describes a randomized procedure that pairs demands until one demand is left which is then routed to the sink. More formally we have the following algorithms:

1. We initialize $S_0 = S \cup \{t\}, w_0 = w, E' = \varnothing$ and $i = -1$
2. While $S_i \neq \{t\}$.
   (a) $i \leftarrow i + 1$.
   (b) For every pair of vertices $(u, v) \in S_i/\{r\}$:
      i. We find the shortest $u - v$ path P_{uv} with respect to the distance metric $M_{uv}(e) = c(e) + \frac{2 \cdot w_i(u) \cdot w_i(v)}{w_i(u) + w_i(v)} \cdot l(e)$.
      ii. Let $K_i(u, v) = \sum_{e \in P_{uv}} M_{uv}(e)$.
   (c) For every node $u \in S_i/\{r\}$:
      i. We find the shortest $u - r$ path P_{ur} with respect to the distance metric $M_{ut}(e) = c(e) + w_i(u) \cdot l(e)$.
      ii. Let $K_i(u, t) = \sum_{e \in P_{ut}} M_{ut}(e)$.
   (d) We compute a matching between nodes in S_i such that:
      i. The number of unmatched nodes plus half the amount of matched nodes (i.e. the number of matches) is at most $\frac{S_i}{\alpha}$.
      ii. The value of $\sum_{(u,v)_{matched}} K_i(u, v)$ is at most β times the minimum K_i-value of a perfect matching.
   (e) We set $S_{i+1} = \varnothing$.
   (f) For every matched pair (u, v):
      i. We add all edges of P_{uv} into E'.
      ii. Choose u to be the center with probability $\frac{w_i(u)}{w_i(u)+w_i(v)}$ and otherwise v (i.e. with probability $\frac{w_i(v)}{w_i(u)+w_i(v)}$).
      iii. We add the chosen vertex to S_{i+1}
      iv. We set $w_{i+1}(center) = w_i(u) + w_i(v)$
   (g) For all unmatched nodes $u \in S_i$:
      i. $S_{i+1} \leftarrow S_{i+1} \cup \{u\}$
      ii. We set $w_{i+1}(u) = w_i(u)$
   (h) $S_{i+1} \leftarrow S_{i+1} \cup \{r\}$
3. We return $G' = (E', V')$ where E' are the edges selected by the algorithm and V' are the adjacent nodes to edges in E'.

4.4.3 The Analysis

In step 2.d. a minimum value perfect matching is polynomialy computable and would yield $\alpha = 2$ and $\beta = 1$, however the algorithms available for this task are highly impractical. Other approximate matchings can be more easily computed and they result in the same approximation guaranties.

The total running time, if Dijkstra's algorithm is used for computing shortest paths, can easily be show that is $O\left(|S|^2 \cdot (m + n \cdot \log(n))\right)$.

The first important observation concerning the analysis of the algorithm is that the optimal solution is a tree T^*. If the optimal solution G^* is not a tree then we can take shortest paths from r to all nodes in S and the resulting tree subgraph would have at most the same value as G^*.

Let $C^* = \sum_{e \in T^*} c(e)$, $L^*(u)$ be the shortest (u, r) path in T^* and $D^* = \sum_{u \in S} w(u) \cdot L^*(u)$. Finally, let $D_i^* = \sum_{u \in S_i} w_i(u) \cdot L^*(u)$.

The hole analysis focuses on bounding the expected value of each step i of the algorithm. We will show that the value incurred in each step is within a constant factor from the optimal total value (i.e. $C^* + D^*$) and because the algorithm ends in at most after $O(\log(S))$ steps the desired result will be proven.

This result is proven through three main Lemmas.

Lemma 4.4.1. *For every step i, $\mathbf{E}[D_i^*] \leq D^*$*

This Lemma can be easily proven through induction. For the first step it is obvious that $D_0^* = D^*$. Suppose that $\mathbf{E}[D_{i-1}^*] \leq D^*$. For every pair $u, v \in S_{i-1}$ that was matched at step $i - 1$, their contributions in D_{i-1}^* where $w_{i-1}(u) \cdot L^*(u)$ and $w_{i-1}(v) \cdot L^*(v)$ respectively. Since those demands were matched their expected distance from the source is the sum of the distance from center u times the probability we chose that center plus the distance from center v time the respective probability. From the definition of the algorithm we have that the expected distance is going to be $\frac{w_{i-1}(u) \cdot L^*(u) + w_{i-1}(v) \cdot L^*(v)}{w_{i-1}(u) + w_{i-1}(v)}$. The cumulative weight of the new center is going to be $w_{i-1}(u) + w_{i-1}(v)$. Thus it is easy to see that the expected contribution of u and v will not change in step i. It is also obvious that the contribution of unmatched nodes does not change and the contribution of the vertex that is matched with r will drop to zero. Summing over all nodes the lemma is proven.

Lemma 4.4.2. *Given a tree $T = (E, V)$ and a set $S \subseteq V$, there exist a perfect matching of nodes in S such that uses each edge in E at most once.*

This Lemma is fairly easy to prove through induction on the number of nodes in T. If there is a leaf that is not in S then we can remove that leaf. If all leaves are in S then we have two possibilities. We chose a leaf arbitrary. If the leafs parent is in S we match those nodes, remove the leaf from T and remove its parent from S. If the leafs parent is not in S we remove the leaf, add its parent in S, solve this instance and then match the leaf with the same node its parent was matched.

Lemma 4.4.3. *The expected value of step i is at most $\beta \cdot (2 \cdot D^* + C^*)$.*

On every step i there exists a perfect matching on S_i using only edges of the optimal tree solution T^*. Thus there exist a perfect matching that has cost of at most $2 \cdot D^* + C^*$ (the factor of 2 in front of D^* comes from the fact that in the cost of the matching we use $\frac{2 \cdot w_i(u) \cdot w_i(v)}{w_i(u)+w_i(v)}$ which is at most two times the smallest demand). Since we are using a β approximate matching and the expected value of the step is equal to the value of the matching the Lemma follows.

Theorem 4.4.4. *The process described is an $O(\log |S|)$-approximate algorithm for the Cost-Distance problem.*

Using Lemma 4.4.3 we have the expected cost of each step bounded by $\beta \cdot (2 \cdot D^* + C^*)$. Also we can have at most $\log_\alpha |S|$ steps. Thus the total expected value of the algorithm is at most $\beta \cdot (\log_\alpha |S|) \cdot (2 \cdot D^* + C^*) \rightarrow O(\log |S|)$.

4.4.4 Reductions

In this sections we will analyze reductions of this problem to some really important network design problems showing the importance of the Cost-Distance problem.

4.4.4.1 Facility Location

We are given as input a network $G = (V, E)$ a distance metric on each edge c_e, a set of demands $D \subseteq V$ with each demand having a weight d_i associated with it and a set of possible facilities $F \subseteq V$ where each facility has an opening cost f_i. We create a graph $G' = (V', E')$ where $V' = V \cup \{r\}$ where r will act as the sink. For E' we take all edges in E with zero cost and c_e distance and also add edges that connect each facility with the sink with zero distance and cost f_i. It is obvious that the solving the Cost-Distance problem on G' produces a solution of Facility Location on G with the same approximation guarantee.

4.4.4.2 Capacitated Facility Location

In this version of the Facility Location each facility has a maximum capacity of u_i that can be served. If we want to route more than u_i demand at facility i we need to open more copies of that facility paying f_i for each copy of the facility. We follow the same reduction as 4.4.4.1 with a key difference. The cost of the edges that connect facility nodes to the sink have cost metric of f_i and a distance metric $\frac{f_i}{u_i}$. With this reduction we lose a factor of two in the worst case scenario.

4.4.4.3 Single Sink Buy-at-Bulk

Suppose we have the Single Sink Buy-at-bulk problem analyzed in 4.3. Let G be the network from the Single Sink Buy-at-Bulk instance. We replace each edge of $e \in G$ with K parallel edges where each edge $i \in K$ has costs $c_e \cdot \sigma_i$ and distance $c_e \cdot \frac{\sigma_i}{u_i}$. Using this reduction the analysis is analogous to 4.4.4.2. We lose for the same reasons at most a factor of two in the reduction.

4.4.4.4 Multi-level Facility Location

Suppose we have an instance of the Multi-level Facility Location Problem analyzed in section 3.2.3. Let G be the network of that instance. Also suppose that we have k-level problem. We first create k copies of G, $(G_1, G_2, ..., G_k)$. Each edge of copy G_i has cost zero and distance c_e (i.e. the distance of edge e in G). We connect node u in G_i with its copy on graph G_{i+1} with zero distance and f_{ui} cost (recall f_{ui} is the cost of opening a level i facility on vertex u). Last but not least we create a sink r and we connect each node $u \in G_k$ with r through an edge of zero distance and f_{uk} cost. Using the above described graph the approximation preserving reduction between the two problems is obvious.

4.4.4.5 Concave Functions

Suppose we have the problem where a set of demands S needs to be routed to a sink r where the cost of each edge is described by the concave function $f_e(d) = \min_{i=1}^{r_e}(a_{ie} + b_{ie} \cdot d)$ where d is the amount of demand that uses edge e. It is obvious that if we replace each edge by r_e copies where each copy i has cost a_{ie} and distance b_{ie} we have a approximation preserving reduction to the Cost Distance problem. As we will show in section 6 a more generalized version of the Cost-Distance algorithm can solve this problem for a much wider class of concave functions.

Chapter 5

Congestion Games

In this section we will analyze the game theoretic aspect of network problems. In Game Theory in general we are preoccupied with problems in which a set of selfish players want to minimize their own cost, disregarding the total cost of all players in the game. This creates instances where equilibria have far greater cost than the optimal solution. This concept of inefficiency due to selfish behavior is also tightly linked with paradoxes that can be observed. In this section we will analyze the impact of those selfish behaviors in problems where players want to reach a destination in a network as fast as possible also known as Congestion Games.

This section is really important for our main problem since it entails many game theoretic aspects that we analyze in our work. From this section it is important to understand why selfish routing diverges from the optimal solution. This characteristic is perfectly illustrated by Braess paradox. We will see techniques on how to compute the Price of Anarchy and the Price of Stability of a problem, and how to compute exact (when its possible) and approximate Nash Equilibria through best response dynamics. We will examine some very useful tools such as the potential function method. We will examine how the existing bibliography handles games where cost sharing costs exist which might help us handle the facility cost of our problem. Finally, we will examine why it is difficult to detect instances where Braess like paradoxes exist and how we can circumvent those inapproximability results with the use of sparsification techniques.

5.1 Introduction to Congestion Games

In the field of Algorithmic Game Theory a well established and studied field of research are Congestion Games. In an instance of a Congestion Game we usually have a network in which selfish (noncooperative) demands want to reach a destination inside the network. However the latency of each edge of the network depends on the amount of demand using it. A form of congestion is created and

the selfish users deviate from the overall best routing in order to minimize their own latency. We assume that users reach a Nash Equilibrium, however, it is well known that Nash Equilibrium can be highly inefficient.

To formalize this kind of games we will use the definitions and notations used by Roughgarden [83]. Suppose we have a network $G = (V, E)$ where V is the set of nodes and E the set of edges that connect those nodes. For now we will consider the single commodity instance where we only have one source vertex $s \in V$and a sink vertex $t \in V$. All players are gathered in s and want to reach t via the shortest possible path. We will denote with $P \neq \varnothing$ the set of simple $s-t$ paths. A flow f is feasible if $\sum_{p \in P} f_p = r$ where r is the traffic rate that wants to be routed from s to t. We also define $f_e = \sum_{p:e \in P} f_p$. Finally suppose that for each edge e we have a latency function l_e and that $l_p(f) = \sum_{e \in p} l_e(f_e)$ is the total latency of a path $p \in P$. The network G, the traffic rate r, and the latency functions l create an instance of a congestion game.

We will say that a flow f is at Nash Equilibrium if and only if for every $p_1, p_2 \in P$ with $f_{p_1} > 0$, $l_{p_1}(f) \leq l_{p_2}(f)$. This definition states that a flow is at Nash Equilibrium if and only if no amount of demand can deviate from the flow and end up with a lower amount of latency. If we consider two paths p_1, p_2 that both receive non zero demands at a Nash flow f, using the aforementioned definition we have that $l_{p_1}(f) \leq l_{p_2}(f)$ and $l_{p_1}(f) \geq l_{p_2}(f)$ thus $l_{p_1}(f) = l_{p_2}(f)$. Thus all different paths used by a Nash flow receive a common total latency which we will define as $L(G, r, l)$.

5.1.1 Atomic vs Non-Atomic

We call Atomic congestion games the ones where there is a bounded amount of players each one having a nonnegligible weight. Non-atomic games on the other hand are the ones where there is an infinite amount of players each one having infinitesimal weight. Obviously the formalization done earlier is for Non-Atomic congestion game. Similar analysis can be done for Atomic congestion games.

5.1.2 The Price of Anarchy

A really important metric through which we will evaluate the efficiency of a flow is its total cost. We define the total cost of a flow f as:

$$C(f) = \sum_{p \in P} l_p(f) \cdot f_p$$

From the observations we have already made about Nash flows f, we can see that $C(f) = r \cdot L(G, r, l)$ at a Nash Equilibrium.

Now we are ready to state the Price of Anarchy which was proposed first by Koutsoupias et al [74]. The Price of Anarchy is nothing more than the ratio of the maximum possible cost at a Nash Equilibrium to the cost of the optimal flow f^*.

$$PoA = \frac{\max_{f \in Equil} C(f)}{C(f^*)}$$

5.1.3 The Price of Stability

The Price of Stability is a metric very similar to the price of anarchy. It is essentially, the ratio between the lowest cost Nash Equilibrium to the Optimal Cost. In essence it shows us how much of cost we have to sacrifice in order to have stability (i.e. Nash Equilibrium).

$$PoS = \frac{\min_{f \in Equil} C(f)}{C(f^*)}$$

5.2 Braess Paradox

A well known phenomenon that occurs is selfish routing networks is the Braess Paradox that was first stated by Braess in 1968. This paradox states that if we remove an edge from a network its total performance at a Nash Equilibrium (with respect to the total cost) might improve.

(a) The Network (b) The optimal Subnetwork

Figure 5.1: Braess Paradox.

Consider the network depicted in 5.1a. We have a source node s that wants to send $r = 1$ traffic rate to the sink t. It can do so through the following three paths, $s \to v \to t$, $s \to w \to t$ and $s \to v \to w \to t$. The latency of the edges are the ones depicted on the figure where x is the amount of traffic that uses a specific edge. Since the traffic rate is 1 the costs of edges $s - v$ and $w - t$ are always less or equal to 1. Thus the only Nash Equilibrium at this state is if all demand is routed through the path $s \to v \to w \to t$ which will have a total cost of $C(f) = 2$. However if we consider the optimal Subgraph of this network depicted in 5.1b we can see that the only Nash Equilibrium in this network is achieved when half of the total demand is routed through path $s \to v \to t$ and half through $s \to w \to t$

with a total cost of $C(f') = \frac{3}{2}$. Thus the Price of Anarchy in this instance is $PoA = \frac{2}{\frac{3}{2}} = \frac{4}{3}$. Tardos et al. [86] proved that for linear latencies $\frac{4}{3}$ is an upper bound on the price of anarchy. Generalizing their techniques Roughgarden [85] proved that for polynomial latencies with bounded maximum degree of p the price of anarchy is $\left[1 - p \cdot (k+1)^{-\frac{k+1}{k}}\right]^{-1}$ which is asymptoticaly $\Theta\left(\frac{p}{lnp}\right)$ as $p \to \infty$. In this paper he also proved that the Price of Anarchy is independent of the graphs topology.

5.2.1 Inapproximability Results

A natural network design problem that arises from the above observation is that given an instance (G, r, l) of a Congestion Game we would like to find a subgraph $H \subseteq G$ such that $L(H, r, l)$ becomes minimized. It turns out that we can not. More formally Roughgarden [83] proved that for linear latencies there is no $(\frac{4}{3} - \epsilon)$-approximation algorithm for this network design problem and for general increasing non negative latencies there is no $(\frac{n}{2} - \epsilon)$-approximation algorithm where n is the number of vertices in G.

5.2.1.1 Linear Latency

For the inapproximability results we are going to use the 2 Directed Disjoint Paths problem which was proved NP-complete by Fortune et al. [41]. In this problem we are given a network $G = (V, E)$ and $s_1, s_2, t_1, t_2 \in V$. The goal is to find if there are paths P_1 from s_1 to t_1 and P_2 from s_2 to t_2 such that P_1 and P_2 are vertex disjoint.

Suppose we have an instance of the 2 Directed Disjoint Path problem where we have a network $G = (V, E)$. We create a instance of our network design problem as follows. We create a new graph $G' = (V', E')$ where $V' = V \cup \{s, t\}$ and $E' = E \cup \{(s, s_1), (s, s_2), (t_1, t), (t_2, t)\}$. We set for each edge $e \in E$ the latency as $l_e(x) = 0$. We also set for edges $e \in \{(s, s_1), (t_2, t)\}$, $l_e(x) = 1$ and for edges $e \in \{(s, s_2), (t_1, t)\}$, $l_e(x) = x$. Suppose we have an algorithm that solves our network design problem. If there were disjoint paths then our algorithm would choose them yielding a total cost of $\frac{3}{2}$ (recall figure 5.1a). If there were not any disjoint paths then our algorithm would yield a total cost of 2 (recall figure 5.1b). Since the 2 Directed Disjoint Paths problem is NP-complete then there is no $(\frac{4}{3} - \epsilon)$-approximation algorithm for our network design problem (unless $P = NP$).

5.2.1.2 General Latency

For General Latency the proof is much more intricate. First we create a family of graphs that generalize the network of the Braess Paradox. Also it is of great importance that although we want our latencies to be smooth functions we can emulate step functions using a small enough parameter δ. We can have for instance

$l_e(x) = 1$ for $x \leq 1$ and $l_e(x) = M$ for $x \geq 1 + \delta$ where M is a big enough constant to be considered as ∞. For a small enough δ $\left(\text{i.e. } \frac{1}{A \cdot (p+k)}\right)$ and a big enough M $\left(\text{i.e. } \frac{n}{2}\right)$ we can consider edge e to have a capacity of 1. Then using the generalized Braess Graphs and the fact that we can set capacities on edges we can show that we can solve the Partition problem using our network design problem. For a complete proof address [83].

Lin et al. [76] generalized this founding for multi-commodity networks. In those networks instead of a single source sink pair we have k source-destination pairs $(s_1, t_1), ..., (s_k, t_k)$. Lin et al. created a general family of two-commodity instances where removing a single edge can decrease the price of anarchy by $2^{\Omega(n)}$. Again using capacities on edges they showed that finding the best subnetwork of a two-commodity game is equivalent to solving the Partition problem. Thus the Multi-commodity Network Design problem is exponentially inapproximable.

The aforementioned results show that if we want to diminish the Price of Anarchy of a given congestion game we can not do so efficiently by removing edges. The best thing we can do in this direction is find a subnetwork with an approximate Nash Equilibrium that is close to the Nash flow of the optimal subnetwork using sparsification techniques that we will analyze later on. Other techniques have also been adopted such as using Stackelberg routing where we can use a small fraction of coordinated traffic in order to decrease the Price of Anarchy of the rest of the traffic ([16], [42], [67], [71], [84]). Also, the idea of adding tolls on resources has been extensively studied ([66], [21], [31], [40], [45]).

5.3 Computing Approximate Equilibrium

In this section we will analyze a series of works that use best response dynamics in order to compute approximate Nash Equilibria in congestion games. It will be of great interest to see if the techniques proposed by those papers can be extended to find approximate Nash Equilibria in a variant of our main problem.

In general finding an exact Nash Equilibrium in congestion games is computationally hard ([2], [35]). In fact in weighted congestion games exact Nash Equilibria may not even exist [51]. Dunkle et al. [34] by extending the work of Fotakis et al. [44] showed that it is even NP-complete to determine if a weighted congestion game has an exact Nash Equilibrium. That is why a well studied problem is that of finding an approximate Nash equilibrium. We call q-approximate Nash equilibrium a state of a congestion game in which no player can change their strategy and decrease their latency by a factor greater than q. Due to the work of Skopalik et al. [89] we know that computing a q-approximate equilibrium in congestion games is PLS-complete for any q that is polynomialy computable. That is why we focus on special cases of congestion games. In this section for instance we will analyze congestion games with polynomial latencies of bounded degree. Besides the papers we will analyze in this section, positive results in computing approxi-

mate Nash Equilibria have been also given by Chien et al. [26] and Feldotto et al. [37].In the following sections we will analyze different algorithms that use similar techniques to find approximate Nash equilibria in Atomic congestion games.

5.3.1 Preliminaries

We are going to use the notations and definitions of Caragiannis et al. [20] as a bases and change or introduce new notations were necessary. Suppose we have a congestion game with a set $N = 1, 2, ..., n$ of players that all have unit demand. There is a set of resources E and each player u using the resources of E has a set of possible strategies Σ_u. When each player chooses a strategy $s_u \in \Sigma_u$ we have a state $S = s_1, s_2, ..., s_n$ of the game. Also we have a polynomial latency function f_e on each resource $e \in E$, which depends on the amount of players that are using that resource, has a bounded maximum degree d and has non negative coefficients. We will denote $n_e(S) = |\{u \in N : e \in s_u\}|$. Thus the total cost of a player u using strategy s_u is $c_u(S) = \sum_{e \in s_u} f_e(n_e(S))$. Given a state $S = s_1, s_2, ..., s_n$ of the game we will depict as (S_{-u}, s'_u) the state in which all players have the same strategy as in S except player u who has deviated to strategy s'_u (i.e. $(S_{-u}, s'_u) = s_1, s_2, ..., s'_u, ..., s_n$). Finally, we will denote as $BR_u(S)$ the best response of player u in state S (i.e. the strategy s'_u for which $c_u(S_{-u}, s'_u)$ is minimized). With an abuse in notation we can say that $BR_u(0)$ is the best response of player u when no other player has chosen a strategy yet.

In order to formalize the notion of the q-approximate equilibrium stated earlier we introduce the notion of a q-move. We call q-move when a player deviates from strategy s_u of a state S to a strategy s'_u and $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{q}$ (i.e. player u ameliorates his cost by a factor q when he deviates to strategy s'_u). We say that we have a q-approximate Nash equilibrium when there are no available q-moves for any player in the game.

It will be useful to consider sequences of moves where only a set of players $F \subseteq N$ are playing while the rest N/F players are frozen to their original strategy. For this reason we introduce the notation $f_e^F(x) = f_e(x + t_e)$ where t_e stands for the number of players in N/F that use resource e. Also $n_e^F(S)$ denotes the number of players in F that use resource e under the state S.

5.3.2 Potential Function

It is now a good time to introduce the notion of Potential Functions. Congestion Games with unit demands are potential games. In other words they admit a potential function Φ. This function has the following very useful property:

$$\Phi(S_{-u}, s'_u) - \Phi(S) = c_u(S_u, s'_u) - c_u(S)$$

It is obvious that local minima of the potential function consist Nash Equilibria of the game at hand. The first potential function for unweighted congestion games

was introduced by Rosenthal [82] and can be written as $\Phi(S) = \sum_{e \in E} \sum_{j=1}^{n_e(S)} f_e(j)$. From this definition of the potential function we can see that $\sum_{e\ inE} f_e(n_e(S)) \leq \Phi(S) \leq \sum_{u \in N} c_u(S)$ holds. Weighted congestion games do not admit potential function except for linear and exponential latencies [60].

5.3.3 Unweighted Players

In this section we will analyze a simple algorithm that finds approximate Nash equilibria when all players have unit demands introduced by Caragiannis et al [20]. In the next sections we will present works that generalize this algorithm for weighted congestion games using two different techniques.

5.3.3.1 The Algorithm

For the sake of this algorithm we will denote as $\theta_d(q)$ the upper bound of the worst case ratio between the potential of of any q-approximate equilibrium and the minimum potential. For an instance $G = (N, E, (\Sigma_i)_{i \in N}, (f_e)_{e \in E})$ of a congestion game and a constant ψ we have the following algorithm.

1. We initialize $q = 1 + n^{-\psi}$, $p = \left(\frac{1}{\theta_d(q) - n^{-\psi}}\right)^{-1}$.
2. For each $u \in N$ we set $l_u = c_u(BR_u(0))$.
3. We initialize $l_{\min} = \min_{u \in N} l_u$, $l_{\max} = \max_{u \in N} l_u$ and $m = 1 + \lceil \log_{2^{d+1} \cdot n^{2\psi+d+1}} \left(\frac{l_{\max}}{l_{\min}}\right) \rceil$.
4. We then partition the players into blocks $B_1, B_2, ..., B_m$ such that $u \in B_i$ if and only if $l_u \in \left(l_{\max} \cdot (2^{d+1} \cdot n^{2\psi+d+1})^{-i}, l_{\max} \cdot (2^{d+1} \cdot n^{2\psi+d+1})^{-i+1}\right]$.
5. We set the initial state of the game $S = (BR_1(0), BR_2(0), ..., BR_n(0))$
6. For phase $i = 1$ to $m - 1$ such that $B_i \neq \varnothing$:
   (a) While there exists a player u in B_i with a p-move or in B_{i+1} with a q-move, make u deviate to his best response strategy (i.e. set $s_u \leftarrow BR_u(S)$)

5.3.3.2 The Analysis

For the purpose of this thesis we will only going to provide the high level ideas of the analysis presented in [20].

We will call S^i the instance of the game after the end of phase i (S^0 will be the initialized state), $b_i = \left(2^{d+1} \cdot n^{2 \cdot \psi + d + 1}\right)^{-i}$ and R_i the set of players that move at least once during phase i. Finally, Φ_{R_i} is the potential of the game where only players in R_i play and all other players are frozen.

The key lemma concerning this algorithm is the following:

Lemma 5.3.1. *For every phase $i \geq 2$, $\Phi_{R_i}(S^{i-1}) \leq \frac{b_i}{2^d \cdot n^\psi}$*

This lemma is proven by contradiction. We assume that $\Phi_{R_i}(S^{i-1}) > \frac{b_i}{2^d \cdot n^\psi}$ then we can show that state S^{i-1} is not a q-approximate equilibrium for players in $R_i \cap B_i$. However this means that there was a player u in B_i at phase $i-1$ who had a q-move which contradicts with the contradicts step 6(a) of the algorithm.

Now we are ready to prove that this algorithm finishes after a polynomialy bounded number of steps. More formally:

Lemma 5.3.2. *The algorithm terminates after at most $O(n^{5\cdot\psi+3\cdot d+3})$ best response moves.*

Using the facts that:

- The maximum number of Blocks and thus phases of the algorithm are n.
- Lemma 5.3.1 (i.e. $\Phi_{R_i}(S^{i-1}) \leq \frac{b_i}{2^d \cdot n^\psi}$).
- Each best response must decrease the potential by at least $(q-1) \cdot b_{i+2}$.
- And that the maximum degree d of the latency is bounded.

We can bound the amount of best response steps by $O(n^{5\cdot\psi+3\cdot d+3})$.

In order to prove the approximation guarantee we need one last important lemma.

Lemma 5.3.3. *Let u be a player of block B_t with $t \leq m-2$. Let s'_u be a different strategy from the one assigned to u at the end of phase t. The for $i \geq t$, it holds that:*

$$c_u(S^i) \leq p \cdot c_u(S^i_{-u}, s'_u) + \frac{p+1}{n^\psi} \cdot \sum_{k=t+1}^{i} b_k$$

In order to prove this lemma we again use key lemma 5.3.1 to prove two crucial inequalities:

$$c_u(S^{i+1}) \leq c_u(S^i) + \frac{b_{i+1}}{n^\psi}$$

and

$$c_u(S^i_{-u}, s'_u) \leq c_u(S^{i+1}_{-u}, s'_u) + \frac{b_{i+1}}{n^\psi}$$

then using induction we prove the claim.

Finally using lemma 5.3.3 we can fairly easily show that:

Theorem 5.3.4. *The state computed by the algorithm is a $p \cdot \left(1 + \frac{4}{n^\psi}\right)$- approximate equilibrium.*

For linear latencies this translates to a $2+O(n^{-\psi})$-approximate equilibrium and for bounded degree d for the latencies we have a $d^{O(d)}$-approximate equilibrium.

5.3.4 Using Ψ-Games

A natural question that occurs from the previous section is what can be done for computation of approximate Nash Equilibria in weighted congestion games. Caragiannis et al [19] answered this question in their work. The basic problem is that weighted congestion games with polynomial latencies with degree $d \geq 2$ there are no polynomialy computable potential functions. Thus it is not obvious how to generalize the work analyzed in section 5.3.3. For this reason Caragiannis et al [19] used the so called Ψ-games. They admit a potential function and thus the techniques of [20] can be generalized and their outcomes can be used to produce an Approximate Nash Equilibrium in the original congestion game.

5.3.4.1 Preliminaries

We will continue with the notation introduced in section 5.3.1. There are only few additions that we have to make because of the introduction of weights. Each player $u \in N$ has a weight w_u associated with him. Also we will call $N_e(S)$ to be the multiset of weights of players that are using resource e in state S (i.e. $N_e(S) = \{w_u : e \in s_u\}$ where $u \in N$). Finally let $L(A)$ denote the sum of elements in multiset A. Thus the total cost incurred by player u in state S is $c_u(S) = w_u \cdot \sum_{e \in s_u} f_e(L(N_e(S)))$.

5.3.4.2 Ψ-games

First of all, we define a function Ψ_k mapping from finite multisets of reals to reals as follows. $\Psi_0(A) = 1$ for any set A. $\Psi_k(\varnothing) = 0$ for $k \geq 1$. For any non empty multiset $A = a_1, a_2, ..., a_l$ and $k \geq 1$:

$$\Psi_k(A) = k! \cdot \sum_{1 \leq d_1 \leq ... \leq d_k \leq l} \prod_{t=1}^{k} a_{d_t}$$

We can see that for instance $\Psi_1(A) = L(A)$. Since latency functions are polynomials with bounded degree we can depict them as $f_e(x) = \sum_{k=0}^{d} a_{e,k} \cdot x^k$ with $a_{e,k} \geq 0$.

Using the above we can define the Ψ-game. A Ψ-game G can be represented with the tuple $(N, E, (w_u)_{u \in N}, (\Sigma_u)_{u \in N}, (a_{e,k})_{e \in E, k=0,1,...,d})$. In other words everything is identical with a normal potential game. The only change is in the cost payed by a player u in state S which is given by the following formula:

$$\hat{c}_u(S) = w_u \cdot \sum_{e \in s_u} \sum_{k=0}^{d} a_{e,k} \cdot \Psi_k(N_e(S))$$

The reason why we are interested in Ψ-games is twofold. On the one hand Ψ-games are potential games since $\Phi(S) = \sum_e \sum_{k=0}^{d} \frac{a_{e,k}}{k+1} \cdot \Psi_{k+1}(N_e(S))$ is a potential

function for a Ψ-game. Thus the algorithm and the analysis presented in [20] can be generalized in order to yield approximate Nash equilibria for Ψ-games.

On the other hand it is not difficult to see that given a weighted congestion game with polynomial latency functions of degree d and its corresponding Ψ-game it holds that $c_u(S) \leq \hat{c}_u(S) \leq d! \cdot c_u(S)$. Using this observation it is not difficult to see that a p-approximate Nash equilibrium in a Ψ-game is a $d! \cdot p$-approximate Nash equilibrium in its corresponding congestion game.

The rest of paper [19] goes on to provide an algorithm to compute $\frac{3+\sqrt{5}}{2} + O(\gamma)$-approximate Nash equilibrium for linear latencies and $d^{d+o(d)}$-approximate equilibrium for general latencies. This algorithm has a polynomial running time in γ^{-1} and the size of the input. Given the aforementioned observations this algorithm yields a $\frac{3+\sqrt{5}}{2} + O(\gamma)$ and a $d^{2 \cdot d+o(d)}$-approximate Nash equilibria for linear and general congestion games respectively.

5.3.5 Using Approximate Potential Function

In their work Giannakopoulos et al [48] they were able to compute $d^{d+o(d)}$-approximate Nash equilibrium for congestion games. Their motivation was that in [19] the algorithm performs the sequence of best responses in the Ψ-game, thus when we translate those moves to the original congestion games there are cases where players would deteriorate their position. Thus the algorithm proposed in [19] is not a natural one. To achieve this instead of using Ψ-games [19] for their potential function they used approximate potential functions. Those approximate potential functions and their properties were developed in parallel by Giannakopoulos et al [49] and we will talk extensively about them in the next section.

First they were able to formulate a matching upper and lower bound for the Price of Anarchy for approximate Nash equilibria. They showed that in a congestion game with polynomial latencies of maximum degree d a p-approximate Nash equilibrium has $\Phi_{d,p}^{d+1}$ Price of Anarchy where $\Phi_{d,p}$ is the unique positive root of the equation $p \cdot (x+1)^d = x^{d+1}$.

Then they formulated the following function for every resource $e \in E$:

$$\phi_e(x) = a_{e,0} \cdot x + \sum_{k=1}^{d} a_{e,k} \cdot \left(x^{k+1} + \frac{k+1}{2} * x^k \right)$$

Where $a_{e,k}$ are the coefficients of the polynomial cost functions introduced in section 5.3.4. Using the above function we can construct the following approximate potential function $\Phi(S) = \sum_{e \in E} \phi_e(L(N_e(S)))$. The following lemma illustrates why this is an approximate potential function:

Lemma 5.3.5. *For any $e \in E$, $x \geq 0$ and $w \geq 1$:*

$$w \cdot f_e(x+w) \leq \phi_e(x+w) - \phi_e(x) \leq (d+1) \cdot w \cdot f_e(x+w)$$

Where f_e is the cost function of resource e. Using this lemma we can show that $\Phi(S) - \Phi(S_{-u}, s'_u) \geq C_u(S) - (d+1) \cdot C_u(S_{-u}, s'_u)$ which constitutes one of the most important properties of this approximate potential function.

In the rest of paper [48] they use this potential function in order to generalize the algorithm of [20].

5.3.6 Unifying Approximate Potential Function

In this section we will see how to compute potential functions for a variety of latency functions including cost sharing games. If we are to generalize the work of the previous section for our problem this will be a very important tool.

Some really interesting work has been done in finding approximate potential functions ([25], [28], [59], [27], [18]). In the work of Giannakopoulos et al [49] they introduced a way to compute approximate potential functions for most conceivable latency functions. In this section we will present the work presented in [49].

First of all, to understand how to compute approximate potential functions we need the following lemma:

Lemma 5.3.6. *For a given congestion game we assume that for each resource e there exist positive reals $a_{1,e}, a_{2,e}, b_{1,e}, b_{2,e}$ and a function ϕ_e that maps the set of demands that use resource e to reals such that $\phi_e(\varnothing) = 0$ and*

$$a_{1,e} \leq \frac{\phi_e(I \cup i) - \phi_e(I)}{w_i \cdot c_e(w_I + w_i)} \leq a_{2,e} \quad \textit{for all } i \in N, I \subseteq N - \{i\}.$$
$$b_{1,e} \leq \frac{\phi_e(I)}{w_I \cdot c_e(w_I)} \leq b_{2,e} \quad \textit{for all } \varnothing \neq I \subseteq N.$$

Then the game has an (a,b)-equilibrium where $a = \max_{e \in E} \frac{a_{2,e}}{a_{1,e}}$ and $b = \frac{\max_{e \in E} b_{2,e}/a_{1,e}}{\max_{e \in E} b_{1,e}/a_{2,e}}$

Where a state S of a congestion game is an (a,b)-equilibrium if it is a a-approximate Nash equilibrium and the social cost paid in S is at most b times the optimal social cost. Also $w_I = L(I)$.

Firstly we let $\Phi(S) = \sum_{e \in E} \frac{1}{a_{1,e}} \cdot \phi_e(N_e(s))$ which serves as approximate potential function for our congestion game.Then the proof uses the inequalities considered as true from the lemma to conclude in the following two inequalities:

$$\Phi(S_{-i}, s'_i) - \Phi(S) \leq w_i \cdot [a \cdot C_i(S_{-i}, s'_i) - C_i(s)]$$

and,

$$\Phi(S') - \Phi(S) \leq \min_{e \in E} \left(\frac{b_{1,e}}{a_{1,e}} \right) \cdot [b \cdot C(S') - C(s)]$$

which in turn show that:

$$\begin{aligned} \Phi(S) \leq \Phi(S_{-i}, s_i') \quad &\Rightarrow \quad C_i(S) \leq a \cdot C_i(S_{-i}, s_i') \\ \Phi(S) \leq \Phi(S') \quad &\Rightarrow \quad C(S) \leq b \cdot C(S') \end{aligned}$$

which proves the claim.

Now let as define good functions and good games. A function f of a congestion game G is (a_1, a_2, b_1, b_2)-good if there exists $\xi \geq 0$ such that for all $x \in \{0\} \cup [w_{\min}, W]$ and $w \in [w_{\min}, w_{\max}]$ it holds that:

$$a_1 \cdot f(x+w) - \xi \cdot f(w) \leq \frac{1}{w} \cdot \int_x^{x+w} f(t)dt \leq a_2 \cdot f(x+w) - \xi \cdot f(w)$$

and for all $x \in [w_{\min}, W]$:

$$b_1 \cdot f(x) - \xi \cdot f_{\min}(x) \leq \frac{1}{w} \cdot \int_0^{x} f(t)dt \leq b_2 \cdot f(x) - \xi \cdot f_{\max}(x)$$

where $w_{\min}$ is the smallest weight of a player in the congestion game, $w_{\max}$ the maximum weight, W the sum of all weights,$f_{\min}(x) = \min_{y \in [w_{\min}, x]} f(y)$ and $f_{\max}(x) = \max_{y \in [w_{\min}, x]} f(y)$.

A congestion game is $\{(a_{1,j}, a_{2,j}, b_{1,j}, b_{2,j})\}_{j \in J}$-good if for any resource $e \in E$ there exists a non empty set $J_e \in J$ and for each $j \in J_e$ there exists $\lambda_{e,j} \geq 0$ such that:

$$f_e(t) = \sum_{j \in J_e} \lambda_{e,j} \cdot f_j(t)$$

where $f_j(t)$ is a $(a_{1,j}, a_{2,j}, b_{1,j}, b_{2,j})$-good function.

We can now see that function:

$$\phi_e(I) = \int_0^{w_I} f_e(t) \cdot dt + \xi_e \cdot \sum_{i \in I} w_i \cdot f_e(w_i)$$

where ξ_e is a constant that complies with the definition of a good function, is an equation that satisfies lemma 5.3.6. Thus a $\{(a_{1,j}, a_{2,j}, b_{1,j}, b_{2,j})\}_{j \in J}$-good game has an (a, b)-equilibrium where $a = \max_{j \in J} \frac{a_{2,j}}{a_{1,j}}$ and $b = \frac{\max_{j \in J} b_{2,j}/a_{1,j}}{\min_{j \in J} b_{1,j}/a_{1,j}}$.

Also the approximate potential function of this game is the following:

$$\Phi(S) = \sum_{e \in E} \frac{1}{a_{1,e}} \cdot \phi_e(N_e(s))$$

5.3.6.1 Fair Cost Sharing

A really interesting application for us when we have cost sharing latencies. Suppose a congestion game where $w_{\min} = 1$ (can be achieved with scaling if necessary) and latency functions $f_e(x) = \frac{F_e}{x}$ where $F_e \geq 0$. We can easily see that those functions can be described as linear combinations of the function $f(x) = \frac{1}{x}$ so we will focus on that and the outcomes will be transferable to the more general case.

However in order to compute an approximate potential function for this problem we need to integrate the cost function which can not be done in this case. However since the minimum amount of demand that can pass through a resource is $w_{\min} = 1$ it is equivalent to consider the following latency:

$$f(x) = \begin{cases} \frac{1}{x} & x \geq 1 \\ \lambda & 0 \leq x \leq 1 \end{cases}$$

This latency function is (a_1, a_2, b_1, b_2)- good for:

$$\begin{aligned} a_1 &= 1, & a_2 &= \max\left(\left(1 + \frac{1}{w_{\max}}\right) \cdot \ln(1 + w_{\max}), \ln(w_{\max}) + \lambda\right), \\ b_1 &= \lambda, & b_2 &= \ln(W) + \lambda. \end{aligned}$$

In short to prove this we can chose $\xi = 0$ which leads us to $\phi_e(x) = \ln(x)$ which in turn proves the necessary bounds.

Using the above it is easy to see using lemma 5.3.6 that this congestion game has an (a, b)-equilibrium where:

$$\begin{aligned} a &= \max\left(\left(1 + \frac{1}{w_{\max}}\right) \cdot \ln(1 + w_{\max}), \ln(w_{\max}) + \lambda\right) \\ b &= 1 + \frac{\ln(W)}{\lambda} \end{aligned}$$

The constant λ is one we can chose depending on the problem at hand and serves as a trade-of between the approximation guarantee of the equilibrium and the impact of the equilibrium on the social welfare.

5.4 Cost Sharing Games

Cost Sharing games is a subcategory of congestion games. In those games the cost of using an edge is evenly distributed among all players. So in the instances we will encounter each edge has a cost c_e and if x amount of demand is using that edge then its cost for each unit of demand is $\frac{c_e}{x}$. This cost sharing mechanism is intuitive and abides to Shapleys value [87].

5.4.1 Price of Stability

First we will try to find an upper and lower bound for the Price of Stability for the unweighted case. This bound was first introduced by Anshelevich et al. [8] and we will analyze their examples and proofs. Although their proofs are based on directed graphs, work has also been done for undirected graphs [38].

5.4.1.1 Lower Bound

Consider the instance depicted in figure 5.2. We have k unweighted demands on s that want to be routed to the destinations $t_1, t_2, ..., t_k$. The optimal solution would be for every one to use the edge of cost $1+\epsilon$. In this instance every player would experience a cost of $\frac{1+\epsilon}{k}$. However one player will find it more profitable to deviate and use the edge of cost $\frac{1}{k}$. It becomes evident that the players will deviate one by one until they all use an edge of the form s, t_i. This is the only Nash Equilibrium in this example. The cost that is paid in the equilibrium is $H(k) = \sum_{i=1}^{k} \frac{1}{i} = \Theta(log(k))$. While the best possible solution paid a cost of $1+\epsilon$. For $\epsilon \to 0$ we get an $H(k)$ Price of Stability.

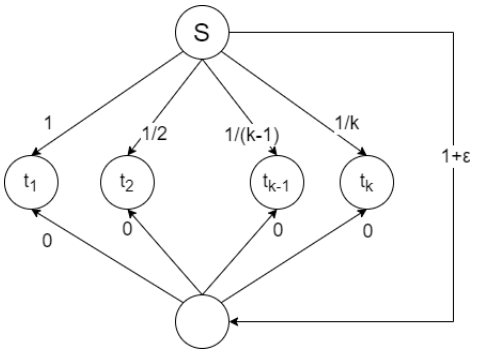


Figure 5.2: A Cost Sharing Instance with PoS $O(log(n))$

5.4.1.2 Upper Bound

This game since it is unweighted has a potential function which is $\Phi(S) = \sum_{e \in E} \sum_{x=1}^{x_e} f_e(x)$ where $f_e(x) = \frac{c_e}{x}$. It is obvious that for any flow S with cost C_S we have $C_S = \sum_{e:e \in S} c_e \leq \sum_{e \in S} c_e + \sum_{e \in E} \sum_{x=2}^{x_e} \frac{c_e}{x} = \sum_{e \in E} \sum_{x=1}^{x_e} \frac{c_e}{x}$. Thus $C_S \leq \Phi(S)$. Also we have that $\Phi(S) = \sum_{e \in E} \sum_{x=1}^{x_e} \frac{c_e}{x} = \sum_{e \in E} c_e \sum_{x=1}^{x_e} \frac{1}{x} = \sum_{e \in E} c_e \cdot H(x_e) \leq H(k) \cdot \sum_{e \in S} c_e = H(k) \cdot C_S$ where k is the number of players in the game and $H(k) = \sum_{i=1}^{k} \frac{1}{i}$. So we end up with the following inequality:

$$C_S \leq \Phi(S) \leq C_S \cdot H(k)$$

Suppose we have an optimal flow to the problem S^*. We know that the flow for which Φ takes its minimum value is a Nash Equilibrium. Lets call S that flow that

minimizes the potential function. We have the following $C_S \leq \Phi(S) \leq \Phi(S^*) \leq H(k) \cdot C_{S^*}$. Thus we have proven a matching upper bound on the price of anarchy for any possible Cost Sharing game.

5.4.2 Computing Minimum Potential Nash Equilibria

In their work Chekuri et al. [24] provide a Linear Programming alogorithm that computes a Nash Equilibrium in cost sharing games. However, there is a key difference to the cost sharing game we have seen so far. The cost that each player pays for using an edge differs in this setting than the ones we have examined earlier. Also here demands are able to split their flow in order to achieve smaller latency.

We will first consider the unweighted case. In their setting they suppose that player i sends a fraction $f_{e,i}$ of his demand through edge e. Without loss of generality we can assume that $f_{e,1} \leq f_{e,2} \leq ... \leq f_{e,n_e} \leq 1$ where n_e is the total amount of players routing non-zero flows through e. Also we assume that $f_{e,0} = 0$. Then player i pays $c_e^i = c_e \cdot \sum_{k=1}^{i} \frac{f_{e,k} - f_{e,k-1}}{n_e - k + 1}$ cost for using edge e. Notice that the total cost paid for e is $f_{e,n_e} \cdot c_e$. This means that the whole cost of edge e is not paid. This subtle difference allows them to come up with a potential function for this game which is the following:

$$\Phi(S) = \sum_{e \in S} \left(\sum_{j=1}^{n_e(S)} \sum_{i=1}^{n_e(S)+1-j} c_e \cdot \frac{f_{e,j} - f_{e,j-1}}{i} \right)$$

Now from their original graph $G = (V, E)$ they create a new graph $G' = (V, E')$ in which they take each edge $e \in E$ and create n copies $e_1, e_2, ..., e_n$ where n is the total amount of demands. The cost of edge e_i is $\frac{c_e}{i}$. We also denote f_p^i the amount of flow that demand i send through path p. Then we can construct the following LP formulation:

$$\min \sum_{e \in E} \sum_{j=1}^{n} \left(\sum_{i=1}^{j} \frac{c_e \cdot x_{e_j}}{i} \right) \tag{5.1}$$

$$\text{s.t.} \quad \sum_{p} f_p^i \geq 1 \qquad \forall \text{ commodities i} \tag{5.2}$$

$$\sum_{e_j \in p} f_p^i \leq x_{e_j} \qquad \forall \text{ edges e, copies j, commodities i} \tag{5.3}$$

$$\sum_{i=1}^{n} \sum_{e_j \in p} f_p^i = j \cdot x_{e_j} \qquad \forall \text{ edges e, copies j} \tag{5.4}$$

$$0 \leq x_{e_j} \leq 1, \quad f_p^i \geq 0 \tag{5.5}$$

Note that there is an exponential lumber of variables, however it can be solved in polynomial time via the dual program using the Ellipsoid algorithm as stated in [24]. Afterwards they show that flow in G' produced by the linear program can be transformed to a solution of minimum potential in G. Later they also present a generalization of the algorithm for weighted games.

5.5 Edge Sparsification Techniques

As we have already seen in 5.2.1 detecting and efficiently eliminating instances of the Braess's Paradox in networks is NP-hard. Is there a way to escape Braess's Paradox under mild assumptions? The answer is yes and this can be achieved through sparsification techniques. Essentially what we are going to show in this section is that in any network under specific assumptions there is at least one subnetwork that emulates the best subnetwork with great accuracy and the total amount of paths used is small enough that we can perform exhaustive search in order to find it. This will become more evident once we analyze the two techniques one of which is based on Althöfer's Lemma and the other one is based on an approximate version of Caratheodory's theorem.

5.5.1 Preliminaries

We are going to use in this section the definition and notations of Fotakis et al [43]. We have a selfish routing instance that consists of a network $G = (V, E)$, a source and a sink vertex s and t respectively, a continuous, differentiable and convex latency function l_e for each edge $e \in E$ and a traffic rate r that wants to be routed from source s to the sink t. Let $n = |V|$, $m = |E|$ and P denote the set of simple $s - t$ paths in G. We will call any subgraph $H(V, E')$ obtained from G by deleting edges, a subnetwork.

A flow f specifies an amount of demand f_p that is routed through each different simple path in P. A flow is feasible if $\sum_{p \in P} f_p = r$. Also we will call $f_e = \sum_{p:e \in p} f_p$ the amount of flow that passes through each edge $e \in E$. The latency of a path p under flow f is $l_p(f) = \sum_{e \in p} l_e \cdot (f_e)$. For a flow f let $E_f = e \in E : f_e > 0$ be the set of edges that are being used by flow f and $G_f = (v, E_f)$ the subnetwork of G corresponding to f.

We will assume that there is an infinite amount of players in source s that selfishly want to route their infinitesimal demand to the sink t. A flow f will be at Nash equilibrium if no infinitesimal player can deviate from the flow and end up with a smaller latency (i.e. for every path p where $f_p > 0$ it holds that $l_p(f) \leq l_{p'}(f)$ where p' is an arbitrary path). Therefore all players incur a common latency in a Nash flow f which we will call $L(f)$.

Finally, we will call an ϵ-Nash flow, a flow f such that for every path p with $f_p > 0$ it holds that $l_p(f) \leq l_{p'}(f) + \epsilon$ where p' is an arbitrary path.

In the best subnetwork problem we will be given an instance of selfish routing and we will have to find a subnetwork H^B such that $L(H^B) \leq L(H)$ for any other subnetwork H. Where $L(G)$ is the common latency at a Nash equilibrium in G.

5.5.2 Althöfer's Lemma

For this technique Fotakis et al [43] used a Probabilistic Method that was based on the proof of Althöfer's lemma [5]. The following lemma will help us find an approximate solution for the best subnetwork problem.

Lemma 5.5.1. *Suppose we have an instance of a graph $G = (V, E)$ and a feasible flow f. Then for any $\epsilon > 0$, there exists a feasible flow $\hat{f}$ that assigns positive traffic to at most $\lfloor \frac{\log(2 \cdot m)}{2 \cdot \epsilon^2} \rfloor + 1$ paths, such that for any $e \in E$, $|\hat{f}_e - f_e| \leq \epsilon$.*

The proof employs the following idea. We view flow f as a probability distribution over the paths. We perform $k > \frac{\log(2 \cdot m)}{2 \cdot \epsilon^2}$ rounds where we pick a path at random using this probability distribution. We assign flow at each path proportional to the amount of times that we have selected that path during the k rounds. We will show that there is a non negative probability that we end up with a flow that satisfies the guarantees of the lemma. Then such flow exists.

More formally let $k = \lfloor \frac{\log(2 \cdot m)}{2 \cdot \epsilon^2} \rfloor + 1$. We can assume that traffic rate is 1 (this can be achieved with scaling if necessary).Let $\mu = |P|$ where P is the set of paths. We have $Q_1, Q_2, ..., Q_k$ which are independent random variables where for each $j \in [\mu]$ $\mathbf{P}[Q_i = j] = f_j$. We have $F_{e,i} = (1$ if $e \in Q_i$ else $0)$. Let $F_e = \frac{1}{k} \cdot \sum_{i=1}^{k} F_{e,i}$. We can see that $\mathbf{E}[F_e] = f_e$.

We can now apply Chernoff-Hoeffding bound ([61]) which leads us to the following inequality:

$$\mathbf{P}[|F_e - f_e| > \epsilon] \leq 2 \cdot e^{-2 \cdot \epsilon^2 \cdot k} < \frac{1}{m}$$

Where the second inequality holds because of the value of k we have already chosen. Finally by applying union bound over all edges we conclude that $\mathbf{P}[\exists e : |F_e - f_e| > \epsilon] < m \cdot \frac{1}{m} \Rightarrow \mathbf{P}[\forall e : |F_e - f_e| \geq \epsilon] > 0$. Thus a feasible flow with the characteristics of the lemma exists.

Now using this lemma it is easy to prove the following theorem.

Theorem 5.5.2. *Suppose we have an instance of selfish routing on a network $G = (V, E)$ where $l_e(x) = a_e \cdot x + b_e$ and $r = 1$. Let $a = \max_{e \in E}\{a_e\}$ and H^B be the best subnetwork in G. Also let $d_1, d_2 > 0$ be constants such that $|P| \leq m^{d_1}$ and $|p| \leq \log^{d_2} m$ for all $p \in P$. Then for any $\epsilon > 0$ we can compute in time $m^{O(d_1 \cdot a^2 \cdot \log^{2 \cdot d_2 + 1}(2 \cdot m)/\epsilon^2)}$ a flow $\hat{f}$ that is an ϵ-Nash flow on $G_{\hat{f}}$ and $l_p(\hat{f}) \leq L(H^B) + \frac{\epsilon}{2}$ for all paths p in $G_{\hat{f}}$.*

To prove this theorem we set $\epsilon_1 = \frac{\epsilon}{2 \cdot a \cdot \log^{d_2}(2 \cdot m)}$. We then use lemma 5.5.1 to state that there exists a flow $\hat{f}$ on H^B such that $|f_e - \hat{f}_e| \leq \epsilon_1$ that uses at most $k = \lfloor \frac{\log(2 \cdot m)}{2 \cdot \epsilon_1^2} \rfloor + 1$ paths, where f is the flow of H^B. Also $\hat{f}_e = 0$ for any edge not in H^B. Since $|f_e - \hat{f}_e| \leq \epsilon_1 \Rightarrow |l_e(\hat{f}) - l_e(f)| \leq a \cdot \epsilon_1$. Summing this inequality over all edges of a path p we have that $|l_p(\hat{f}) - l_p(f)| \leq a \cdot \log^{d_2}(2 \cdot m) \cdot \epsilon_1 = \frac{\epsilon}{2}$. Finding this path can be done through exhaustive search over the $m^{O(d_1 \cdot k)}$ different possible combinations.

5.5.3 Approximate Caratheodory's Theorem

We can achieve similar results with the one analyzed in 5.5.2 by using an approximate version of Caratheodory's Theorem first stated by Barman [13].

The Caratheodory's theorem in short states that if a point x lies in the convex hull of a set P then x can be written as the convex combination of $d+1$ elements of P where d is the number of dimensions in which set P exists.

The approximate Caratheodory's Theorem introduced by Barman [13] is the following:

Theorem 5.5.3. *Let X be a set of vectors $X = x_1, ..., x_n \subset \mathbb{R}^d$ and $\epsilon > 0$. For every $\mu \in conv(X)$ and $2 \leq p \leq \infty$ there exists an $O(\frac{p \cdot \gamma^2}{\epsilon^2})$-uniform vector $\mu' \in conv(X)$ such that $\|\mu - \mu'\|_p \leq \epsilon$, where $\gamma = \max_{x \in X} \|x\|_p$ and a vector is k-uniform when it can be expressed as an average of k vectors of X with replacements allowed.*

We will use this theorem in the equivalent way we used lemma 5.5.1 to produce an sparse ϵ-Nash flow.

Theorem 5.5.4. *Suppose we have an instance of selfish routing on a network $G = (V, E)$ where $l_e(x)$ are a-Lipschitz. Let H^B be the best subnetwork in G. Also let $|p| \leq M$ for all $p \in P$. Then for any $\epsilon > 0$ there exists a flow $\hat{f}$ that uses $O(\frac{a^2 \cdot M^3 \cdot r^2}{\epsilon^2})$, is an ϵ-Nash flow on $G_{\hat{f}}$ and $l_p(\hat{f}) \leq L(H^B) + \frac{\epsilon}{2}$ for all paths p in $G_{\hat{f}}$.*

To prove this we are going to use theorem 5.5.3. First of all for each path $p \in P$ we create a vector x_p of $|E|$ dimension (i.e. $x_p = (x_{p,1}, x_{p,2}, ..., x_{p,|E|})$). We set $x_{p,e} = r$ if $e \in p$ otherwise we set it to 0. Let $X = \{x_1, x_2, ..., x_{|P|}\}$ be the the set of vector in theorem 5.5.3. It is obvious that any feasible flow lies in the convex hull of X. We can also see that $\gamma = \max_{x \in X} \|x\|_2 = \sqrt{r^2 \cdot \max_{p \in P} |p|} \leq \sqrt{r^2 \cdot M}$.

Let $o \in conv(X)$ be the equilibrium flow of H^B. Without loss of generality we can assume that $o_e > 0$ for all $e \in E(H^B)$. Using theorem 5.5.3 we can see that there exist $k = O(\frac{M \cdot r^2}{\epsilon_1^2})$ paths of H^B $(i_1, i_2, ..., i_k)$ such that for flow $f = \sum_{j=1}^{k} \frac{x_{i_j}}{k}$ it holds that $\|o - f\|_2 \leq \epsilon_1 \Rightarrow |o_e - f_e| \leq \epsilon_1 \Rightarrow |l_e(o) - l_e(f)| \leq a \cdot \epsilon_1 \Rightarrow |l_p(o) - l_p(f)| \leq a \cdot M \cdot \epsilon_1$. Choosing $\epsilon_1 = \frac{\epsilon}{2 \cdot a \cdot M}$ we end up with $l_p(f) \leq L(H^B) + \frac{\epsilon}{2}$.

Using the above we can see that with exhaustive search we can find subnetwork H and an ϵ-Nash flow g on H such that $L(g) \leq L(H^B) + \epsilon$. The number of steps during this exhaustive search is $|P|^{O\left(\frac{a^2 \cdot M^3 \cdot r^2}{\epsilon^2}\right)}$.

Chapter 6

Facility Location for Selfish Commuters

In this section we will analyze the work that we have done regarding Facility Location problems with latencies that depend on the amount of demand using each edge. This twist to the existing facility location problems creates a plethora of interesting side problems that we are going to address one way or another in this section. Some of these questions are: can we find the optimal solution for this problem? Is there some kind of structure that the optimal solution follows? If players pay for the facilities with a cost sharing mechanism, can we compute a Nash equilibrium and can we bound the Price of Anarchy?

We begin our analysis by examining Local Search algorithms and provide an example that shows that the local ratio is really big for our problem. We continue by examining Linear Programming techniques. We first prove a theorem concerning the structure of the optimal solution of our problem that could be used as a restriction in the LP formulation, however we show that even with this added restriction the integrality gap is really big for our problem. We continue by using probabilistic techniques analyzed in section 4 and we provide a $O(log|S|)$ approximation algorithm for instances with decreasing latency functions. We continue by analyzing the problem from a game theoretic point of view. We compute approximate Nash Equilibria, we provide matching upper and lower bounds for the Price of Anarchy and we show how we can use sparsification techniques to solve a simpler version of our problem. Finally, we provide a counter example that captures the difficult essence of the more general problem and we show through this example why sparsification techniques fail in the more general instance.

6.1 Formulation of the Problem

We are now going to formally state the Facility Location for Selfish Commuters problem. We are given a graph $G = (V, E)$ along with a set of source vertices

$S \subseteq V$. We are given for every edge $e \in E$ a latency function $l_e : \Re \mapsto \Re$. We are also given a weight function $w : S \mapsto \Re$ on every source. Finally, for each node $i \in V$ we are given a facility cost f_i.

Our objective is to open a set of facilities F and route the demands to those facilities while minimizing the following expression:

$$\sum_{e \in E} x_e \cdot l_e(x_e) + \sum_{j \in F} f_j$$

Where x_e is the amount of demand that is being transferred through e.

6.2 Why Local Search does not Work in our Setting

One natural idea would be to try and solve this problem using local search algorithms. Recall in section 3.5 we analyzed local search algorithms where we had constants as the edge latencies and a general family of functions that represented the facility cost. Now we want constant costs regarding the facilities and polynomial latencies on the edges. However it is not obvious how those ideas can be used in our setting.

Consider the following example for instances where the latency of an edge is a polynomial of bounded degree d. Since in our problem we have constant facility costs we are going to examine local steps that do not take into consideration the amount of demand each facility accommodates. We are going to use the **open** local step where one facility is opened and the demand is rerouted optimally, the **close** local step where an open facility is closed and the demand is rerouted optimally and the **swap** where an opened facility is closed while a closed facility is opened simultaneously and the demand is the rerouted optimally.

In the example illustrated in figure 6.1 we have nodes $c_1, c_2, ..., c_k$ all having unit demand. The cost of opening a facility on o_i is ϵ and the latency of edges (c_i, o_i) are constant and equal to 1. We also have another possible facility node S with cost of opening a facility $(k-1)^{d+1}$. The latency of edges $e = (c_i, S)$ is $l_e(x) = x^d$. The optimal solution is opening all o_i facilities with cost $k \cdot \epsilon$ and route each demand c_j to its corresponding facility o_j. Now consider the following locally optimal solution where only S is open. Obviously we can not close S. We can not open o_i because it would not change the routing cost and just add an extra ϵ to the facility cost. Also, we can not swap S with o_i. If we did this then the facility cost would be just ϵ but the routing cost would be $3 \cdot (k-1) + 1 + (k-1) \cdot (k-1)^d$ since all demand need to be routed at o_i. Thus having S as our only facility is a locally optimal solution with a local ratio of $O(k^d)$.

Consider also the following argument for any set of local steps in this setting. Recall that in local search algorithms, to prove the approximation guarantees

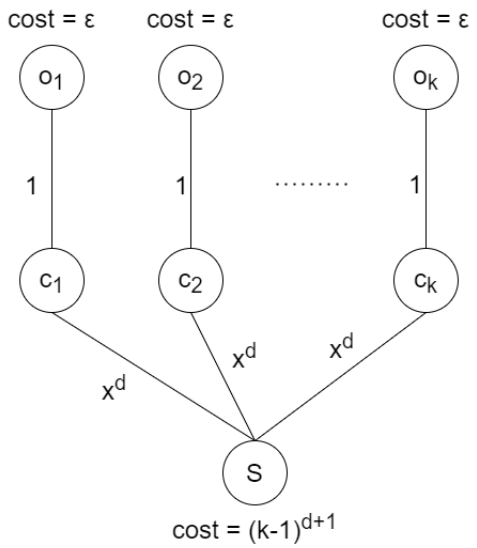


Figure 6.1: A counter example for the use of Local Search

we usually employ the following analysis. We first assume that we have found a locally optimal state. This means that any local step will increase the total cost. Thus we state one by one local steps that involve the optimal solution. We end up with inequalities that have both costs concerning the locally optimal solution, and costs of the universally optimal solution. Summing all of those inequalities and rearranging the terms yield a bound of the local optimal state as some multiplicative factor of the universally optimal solution. However it is really difficult to use this reasoning with our problem. Suppose we state a list of local steps and we want to find out information concerning a locally optimal solution. We will have to use the fact that no local step can improve the cost of the locally optimal state in order to bound its cost. However we need good estimations concerning the new routing costs once a local step is made. Since the edge latencies depend on the amount of traffic they receive, then we can not come up with an estimate concerning the routing cost while taking into consideration only demands that are being affected due to the local step. In other words, in order to estimate the increase or decrease in the total latency cost we need to take into account all of the demand that have passed through each edge up until we reached our final state. This fact increases the complexity and new techniques must be created in order to tackle this problem.

6.3 Why Linear Programming does not Work in our Setting

A second natural idea would be to find some underlying structure of the optimal solution in order to come up with an LP-rounding or primal-dual algorithm. To do so in this section we will try and simplify our problem. We will assume that the facility cost is the same for all nodes and equal to B (i.e. $f_j = B, \forall j \in V$). We will also assume that our latency functions are linear functions with no negative coefficients (i.e. $l_e(x) = a \cdot x + b$ where $a, b \geq 0, \forall e \in E$).

6.3.1 Proving a Lower Bound on the Demand of each Facility

We conjecture the following. For linear latencies and constant facility costs, in an optimal solution each open facility serves at least $\min_{s \in S} w_s$ demand (it is easy to show that for the more general case this conjecture does not hold). Although we were not able to prove or disprove this conjecture we were able to come up with the following lemma that approximates this conjecture.

Lemma 6.3.1. *There exists a flow for the FLSC problem with cost at most* $4*OPT$ *where an open facility serves at least* $\min_{s \in S} \frac{w_s}{2}$ *demand. Where* OPT *is the cost of the optimal flow.*

Proof. Suppose we have the optimal solution with cost OPT. Lets call F^* the set of open facilities and $D \subseteq F^*$ the set of open facilities that receive less than $\min_{s \in S} \frac{w_s}{2}$ flow. Lets also call P_i^s the amount of undivided flow that demand s sends through path i in the optimal solution. We will also call $End(i)$ the facility at which path i terminates. Obviously if $End(i) \in D$ then $P_i^s < \frac{w_s}{2}$. For every d where $\sum_{i:End(i) \in D} P_i^s < \frac{w_s}{2}$ we can re-route those flows towards facilities that are not in D and at most double each flow P_i^s where $End(i) \notin D$.

Now lets call $S' \subseteq S$ the demands where $\sum_{i:End(i) \in D} P_i^s \geq \frac{w_s}{2}$ and lets call $D' \subseteq D$ the facilities in D that receive flow from demands in S'. Note that $|S'| < |D'|$. That is because if $|S'| \geq |D'|$ the total amount of flow that D' receives from S' is $\sum_{s \in S'} \sum_{i:End(i) \in D} P_i^s > \sum_{s \in S'} \frac{w_s}{2} \geq |S'| * \min_{s \in S} \frac{w_s}{2} \geq |D'| * \min_{s \in S} \frac{w_s}{2}$ which means that at least one facility in D' receives flow more than $\min_{s \in S} \frac{w_s}{2}$ which is a contradiction to our original hypothesis. Thus it holds that $|S'| < |D'|$.

So we perform the following process. For every demands $s \in S$ where $\sum_{i:End(i) \in S} P_i^s < \frac{w_s}{2}$ we route all of its flow to facilities $\notin D$ augmenting those P_i^s where $End(i) \notin D$ at most two times. This in turn augments the flow on each edge at most two times. After this move any facility that receives no demand is closed. Thus the only facilities in D that remain are the one that are also in D'. Those facilities receive demand only from demands in S'. However we know that $|D'| > |S'|$.

So we can close all facilities in D' and open one facility on each demand in S' without augmenting the facility cost.

The only thing left to answer is how will we route the demands of $s \in S'$ after a facility opens on it. A naive answer would be that we could send all of s demand to its facility. However there might has been a $P_i^s > 0$ with $End(i) \notin D$ which when removed can make the amount of demand received by that facility drop to an amount less than $\min_{s \in S} \frac{w_s}{2}$ compromising the goal of this process. What we actually do for $s \in S'$, is route $\sum_{i:End(i) \in D'} P_i^s$ (which is greater than $\frac{w_s}{2} \geq \min_{s \in S} \frac{w_s}{2}$) to the facility opened on s and leave the rest of its flow untouched.

To recapitulate, after this process the facilities $\notin D$ receive more or equal to the amount of demand they received before our process begun and thus obey the restrictions of the lemma. All of facilities $\in D$ have closed. Facilities on S' have opened that receive more than half of the demand of each $s \in S'$. The total amount of facilities have not been augmented and the flow on each edge has at most doubled. Thus the resulted state obeys the restrictions of the lemma and has a routing cost at most 4 times the routing cost of the optimal solution. □

6.3.2 The Integrality Gap

Consider the following natural formulation of the problem:

$$min \sum_{e \in E} (a_e \cdot x_e^2 + b_e \cdot x_e) + \sum_{e \in F} z_e \cdot B$$

$$\begin{array}{lrcll} s.t. & \sum_{e \in \delta^-(v)} x_e & = & \sum_{e \in \delta^+(v)} x_e, & \forall v \in V \text{ (flow conservation)} \\ & x_e & \leq & z_e \cdot \sum_{s \in S} w_s, & \forall e \in F \text{ (use only built infrastructure)} \\ & x_e & = & w_s & \forall e \in S \\ & z_e & \in & \{0.1\} & \forall e \in F \end{array}$$

In the above formulation we treat demands as edges that lead into the graph and carry already w_s demand, and facilities as edges that leave from the graph. The variable z_e becomes 1 when the corresponding facility is opened and 0 otherwise. Thus the second inequality of our restrictions ensures that only open facilities are being used. This linear program has an integer restriction and thus can not be solved in polynomial time. However this restriction is essential to the formulation. Consider the relaxation where we just demand that $0 \leq z_e \leq 1$. This problem can be solved in polynomial time however its optimal solution offer no valuable information because of the following observation.

When relaxing the integrality constraint on z_e, since we have a minimization problem z_e will receive the smallest possible value and thus we will have $x_e = z_e * \sum_{s \in S} w_s \Rightarrow z_e = \frac{x_e}{\sum_{s \in S} w_s} \Rightarrow \sum_{e \in F} z_e * B = \sum_{e \in F} \frac{x_e}{\sum_{s \in S} w_s} * B = \frac{B}{\sum_{s \in S} w_s} \cdot \sum_{e \in F} x_e$. However all demands must be served by a facility so $\sum_{e \in F} x_e = \sum_{s \in S} w_s \Rightarrow \frac{B}{\sum_{s \in S} w_s} \cdot \sum_{e \in F} x_e = B$. This tells us that no mater how many facilities are opened

and no matter how the demand will be arranged to them, the facility cost will always be equal to B. So in order to minimize the routing cost a fractional facility will be opened at every demand bringing the routing cost down to zero. Since the fractional problem returns always the same solution with the same cost, it is of no use for a better approximation than the naive. In other words the intergrality gap of this non-linear program is $O(n)$. Our problem in general becomes really easy when intergrality constraints are remove and thus LP based solutions have big integrality gaps and thus the inherently cannot provide valuable information.

6.4 Network Design Techniques

In this section we will examine how probabilistic techniques analyzed in previous sections can be used to solve our problem. For this reason we will examine the k-median variant of our problem (i.e. there is no facility cost and we can open at most k facilities) and we will demand our latency functions l_e to be good. We call a function $l : \Re \mapsto \Re$ good if it is non-negative, decreasing and also $x \cdot l(x)$ is an increasing concave function. We will call this variant of our problem the Concave FLSC problem or CFLSC for short.

Meyerson et al [79] as stated in section 4.4.4 have shown that their problem can generalize instances of concave cost functions that can be seen as many parallel linear cost functions. Our algorithm is based on the work of [79] however their algorithm is heavily based on the notion that their cost functions are linear. We introduce a novel matching technique that works for much more general latency functions.

6.4.1 The Optimal Solution is a Forest

Let o be the flow in the optimal solution. A key property of the optimal solution on which we base our algorithm is that its flows form a forest on the graph. In other words the optimal solution that opens k facilities consists of flows that do not form circles thus they can be seen as k trees rooted in each one of the opened facilities. This property is ensured by the fact that our latency functions are good.

Lemma 6.4.1. *There exists an optimal flow to the CFLSC problem where there are no circles in which all demand flows in one directions.*

Proof. Since $x \cdot l_e(x)$ is increasing, we could simply decrease the flow on this circle leading in a feasible solution and the cost payed for each edge can not be increased. □

Lemma 6.4.2. *There exists an optimal flow to the CFLSC problem where there are no circles in which there are edges with clockwise flow and edges with counter-clockwise flow.*

Proof. This proof is a little bit more involved. We start with the observation that all we need to prove is that once flows meet they will not split until they reach a facility. If we prove this then circles where demands flow in both directions are not possible. However to show this we need a way to talk about the cost of demand separately from the total cost. This is where game theory comes into play.

It is a well known fact that congestion games for the case where demand is infinitesimally small are potential games. The potential function of any such game is the following:

$$\Phi(f) = \sum_{e \in E} \int_0^{f_e} l_e(t)dt$$

Where f is as feasible flow and f_e is the corresponding flow on edge e. This function has the well known property that its local minima are Nash Equilibria. Thus if we create a game on the same graph with new latency functions $\hat{l_e(x)} = (x \cdot l_e(x))'$ the optimal solution of our original game is going to be a Nash Equilibrium in the new one. Where $f'(x)$ is the first derivative of $f(x)$. This holds because the potential function of the new game will be $\Phi'(f) = \sum_{e \in E} \int_0^{f_e} (t \cdot l_e(t))'dt = \sum_{e \in E} f_e \cdot l_e(f_e)$ and the global minimum of this function is a Nash equilibrium for the new game but also minimizes the cost function of our original game.

Now using this knowledge we can examine a Nash equilibrium in a game with $\hat{l_e(x)} = (x \cdot l_e(x))'$ as its latencies. Since $x \cdot l_e(x)$ is concave $\hat{l_e(x)}$ is decreasing. Lets suppose that we have some flow that splits to paths P_1 and P_2 in a Nash Equilibrium. Obviously the two paths must have equal cost under that flow otherwise it would not be a Nash Equilibrium. However since latencies are decreasing the total cost of demand choosing P_1 would not increase if we moved them to path P_2 and thus the potential function would not increase. Lets call f^* the flow that minimizes $\Phi'(x)$. From the above we can conclude that there exists a flow f where $\Phi'(f^*) = \Phi(f)$ and in f demands do not split once they have met. It is important to note that although our problem treats demands as unsplittable we have proven the lemma for splittable flows. However any unsplittable flow is a feasible splittable flow, and since we have proven that splittable flows actually remain unsplittable then the proof works for our instance also. □

From lemmas 6.4.1 and 6.4.2 it follows that:

Theorem 6.4.3. *There exists an optimal flow to the CFLSC problem that has no circles (i.e. is a forest).*

6.4.2 It Generalizes the Cost-Distance Problem

Theorem 6.4.4. *The CFLSC problem is a generalization of the Cost-Distance problem [79].*

Proof. We need to prove two things. One, that our latencies generalize the latencies of the Cost-Distance problem. Two, that the k-median problem we solve is more general that having one sink.

First of all it is easy to show that our latencies generalize those of the Cost-Distance problem. Recall that in the cost distance problem we have a cost metric $c(e)$ and a distance metric $l(e)$ for each edge e. The total cost incurred by one edge on a flow f is $c(e)+l(e)\cdot\sum_{s:e\in P_s} w_s$ where P_s is the route chosen for demand $s\in S$ in f. We can achieve the same cost by using the following latency function:

$$l_e(x)=\begin{cases}\frac{c(e)}{x}+l(e) & \text{if } x_e>\min_{s\in S}(w_s)\\ \frac{c(e)}{\min_{s\in S}(w_s)}+l(e) & \text{if } x_e<\min_{s\in S}(w_s)\end{cases}$$

Note that for $x>\min_{s\in S}(w_s)$ this function has all the properties we want. For $x<\min_{s\in S}(w_s)$ we do not really care because no flow less than $\min_{s\in S}(w_s)$ can exist and we just want $l_e(0)$ to be bounded so as to pay zero cost when no flow passes through e.

Now assume we have a graph G on which we want to solve the cost distance problem. Lets call t the sink of G. We can solve the k-median problem with $k=1$ and add an extra demand on t of weight equal to the sum of weights of all other demands (i.e. $w_t=\sum_{s\in S} w_s$). Obviously one possible solution is to open the facility on t thus there is a feasible solution that achieves the same cost as the cost distance problem. Now assume that the optimal solution of the 1-median problem opens the facility at an arbitrary vertex v. We can obviously move the facility on t and route all of the demand to v and then using the path that t used send all of the demand together at t without augmenting the cost. Notice that demands that meet the flow of t earlier than v for example on vertex u, they can be sent to u and then follow the rest of the demand to t without augmenting the total cost since $x\cdot l_e(x)$ is increasing.

Recall that it was shown from Chuzhoy et al. [30] that the Cost-Distance problem is $\Omega(loglog|S|)$ hard to approximate. Thus, this hardness also holds for our more general version. □

6.4.3 The Algorithm

The algorithm is influenced by the algorithm proposed in [79]. However their matching mechanism and analysis are tailored made to the fact that the total cost incurred by each edge is linear. We introduce a novel matching mechanism which in turn complicates the resulting analysis. For the algorithm we will need the following distance metric: $g(u,v,w)=\sum_{e\in P^w_{u,v}} w\cdot l_e(w)$ where $P^w_{u,v}$ is the closest path from u to v with respect to the distance metric $l_e(w)$ (i.e. the shortest u,v path if demand w was traveling alone on the network).

1. We initialize $S_0=S$, $w_{0,s}=w_s$ and $i=0$.

2. While $|S_i| > k$:

   (a) For every pair $u, v \in S_i$ find $K_i(u,v) = \min_{z \in V}\{g(u,z,w_{i,u}) + g(v,z,w_{i,v}) + \frac{w_{i,u}}{w_{i,u}+w_{i,v}} \cdot g(z,u,w_{i,u}+w_{i,v}) + \frac{w_{i,v}}{w_{i,u}+w_{i,v}} \cdot g(z,v,w_{i,u}+w_{i,v})\}$.

   (b) Perform a matching on S_i with respect to the costs K_i while at the same time leaving k nodes unmatched (if not possible then $k-1$).

   (c) Set $S_{i+1} = \{\}$.

   (d) For each matching u, v:

      i. Send both demands to the node z that minimized the expression of step 2(a).

      ii. Choose u with probability $\frac{w_{i,u}}{w_{i,u}+w_{i,v}}$, otherwise chose v. Without loss of generality we will assume that we chose u.

      iii. Send the combined $w_{i,u} + w_{i,v}$ demand back to u, add u to S_{i+1} and set $w_{i+1,u} = w_{i,u} + w_{i,v}$.

   (e) Add unmatched nodes to S_{i+1}

   (f) Set $i \leftarrow i + 1$

3. We return as facilities the k nodes that are in S_i and as flows the ones dictated by the above procedure.

6.4.4 The Analysis

For our analysis we introduce the metric C_u^i to be the total cost payed due to the movement of player u in phase i. The total cost payed in phase i is $C^i = \sum_{s \in S_i} C_u^i$. It is not difficult to see the following lemma.

Lemma 6.4.5. *The algorithm presented terminates after $O(log|S|)$ phases.*

If we prove that in each phase of the algorithm the expected cost payed (i.e. $\mathbf{E}[C^i]$ is at most the cost of the optimal solution, the combining this fact with lemma 6.4.5 show that our algorithm is an $O(log|S|)$-approximate algorithm for the CFLSC problem. We will first show that this is true for the first phase and then we will show that the expected cost of each phase is less or equal to the cost of the first phase.

6.4.4.1 Phase 0

Phase 0 is the first phase of our algorithm where no demands have been aggregated yet. We begin with the following lemma which is a generalization of lemma 4.2 [79].

Lemma 6.4.6. *Suppose a set of sources $S' \subseteq S$ that are routed at a facility in node r in the optimal solution. Lets call $T = (E, V)$ the tree that corresponds to the flows of nodes in S' to r. Then there exist a matching like the one done in phase 2(a) of our algorithm where demands only use edges that they use in the optimal flow and at most one source is left unmatched.*

Proof. For every $s \in S'$ we take its path towards r at T until it meets with the path of another source. We will call the vertex in which the two paths merge a level one meeting point. We move all sources at their corresponding level one meeting points. In each meeting point with an even number of demands we match them arbitrarily until no demand is left unmatched. In each meeting point with an odd number of demands we do the same thing however now there will be one demand left out. We continue along the path of those level one meeting points with an unmatched demand until their path merges with the path of another level one meeting point. The vertices in which those level one paths meet we call level two meeting points. We continue along the same line of thought until we reach our final meeting point r where at most one demand will be left unmatched. □

It is obvious that in our algorithm the above matching is possible and we can choose as z in step 2(a) the respective meeting point described by the proof of lemma 6.4.6. Thus to bound the cost of our matching we just need to bound the cost of sending demands to those meeting points and the expected demand of sending them back again. The first half is accomplished with the following lemma.

Lemma 6.4.7. *The cost of sending all demands to their respective meeting points is less than the cost of the optimal solution.*

Proof. From lemma 6.4.6 there is a subtle implication that becomes very useful right now. Each edge is traversed by demands that traverse that edge in the optimal solution. More specifically suppose that an edge $e \in E$ is traversed by as set S'' of demands. In our matching only one of this demands traverses that edge. Thus the amount of flow f'_e traversing edge e in our matching is less or equal than the amount of flow f_e that traverses e in the optimal solution. And since $x \cdot l_e(x)$ is increasing we have that $f'_e \cdot l_e(f'_e) \leq f_e \cdot l_e(f_e)$. Summing over all edges we get that the total cost is less than the optimal cost. □

We will now state a lemma that holds for any phase i of the algorithm.

Lemma 6.4.8. *The expected cost of routing demands back from their meeting point to the selected source is less than or equal to the cost paid for sending them to the meeting point.*

Proof. Suppose we are at phase i. Also lets assume that $a_1, a_2, ..a_m$ are the edges along the path of u to the meeting point and $b_1, b_2, ..b_n$ the ones of v. Then the expected cost $\mathbf{E}[C_{\text{back}}]$ of sending them back to a source is the cost of sending them to u times the probability of choosing u plus the probability of sending them to v

times the cost of sending them there. In other words we have that the $\mathbf{E}[C_{\text{back}}] = (w_{i,u} + w_{i,v}) \cdot \frac{w_{i,u}}{w_{i,u}+w_{i,v}} \cdot (l_{a_1}(w_{i,u} + w_{i,v}) + l_{a_2}(w_{i,u} + w_{i,v}) + ... + l_{a_m}(w_{i,u} + w_{i,v})) + (w_{i,u}+w_{i,v}) \cdot \frac{w_{i,v}}{w_{i,u}+w_{i,v}} \cdot (l_{b_1}(w_{i,u}+w_{i,v}) + l_{b_2}(w_{i,u}+w_{i,v}) + ... + l_{b_m}(w_{i,u}+w_{i,v})) \leq w_{i,u} \cdot (l_{a_1}(w_{i,u}) + l_{a_2}(w_{i,u}) + ... + l_{a_m}(w_{i,u})) + w_{i,v} \cdot (l_{b_1}(w_{i,v}) + l_{b_2}(w_{i,v}) + ... + l_{b_n}(w_{i,v}))$ which is exactly the cost of sending those demands to the meeting point. The inequality holds since the demands are positive and latency functions l_e are decreasing. □

From lemmas 6.4.7 and 6.4.8 we can see that $C^0 \leq 2 \cdot C^*$ where C^* is the total cost of the optimal solution.

6.4.4.2 Phase i

The only thing left to do is to bound the cost of sending demands to their meeting points in phase i of the algorithm. This can be done using the following lemma.

Lemma 6.4.9. *The expected cost of routing demands to their meeting points in phase i is less than the cost of the optimal solution.*

Proof. Suppose at phase i that in a node u a set S''_u of demands has been gathered with a total demand $W_u = \sum_{s \in S''_u} w_s$. The probability of u having this demand is $\frac{w_u}{W_u}$. This claim is easy to see because for u to have that demand it must have been selected in all phases with a total probability of $\frac{w_u}{w_u+w_{v_1}} \cdot \frac{w_u+w_{v_1}}{w_u+w_{v_1}+w_{v_2}} \cdot \frac{w_u+w_{v_1}+w_{v_2}}{w_u+w_{v_1}+w_{v_2}+w_{v_3}}$.... We once again perform the matching described by lemma 6.4.6 with respect to weights $w_{i,u}$.

We are now going to show that the expected cost of the matching described by lemma 6.4.6 is bounded by the cost of the optimal solution. For the sake of the argument lets call $g_e(x) = x \cdot l_e(x)$ the cost payed for the usage edge e (recall $g_e(x)$ is a increasing concave function). Suppose that a specific edge e in the optimal solution is used by demands $w_1, w_2, ...w_k$. Thus the total cost payed for edge e in the optimal solution is $g_e\left(\sum_{j=1}^{k} w_j\right)$. In phase i the expected cost payed by edge e because of demand w_j is the cost payed because a total demand of weight W_{w_j} passes through e times the probability that all of the demand with which w_j has been matched, actually passes through e. That probability can be expressed as the probability of W_{w_j} ending up in w_j (which we have shown earlier that is equal to $\frac{w_j}{W_{w_j}}$), times the probability that demand is not matched earlier in the Tree. We will call the later probability p_{e,w_j}. Thus the expected cost payed by edge e due to demand w_j is $\frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot g_e(W_{w_j})$. Also we will call $p_{e,0}$ the probability that no demand passes through e in our matching either because with probability $\prod_{j=1}^{k}\left(1 - \frac{w_j}{W_{w_j}}\right)$ there are no demands in the subtree underneath e or there was an even number of demands in the subtree underneath e and thus all demands have been matched lower in the Tree. So the total expected cost

of edge e is $p_{e,0} \cdot g_e(0) + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot g_e(W_{w_j})$. However it is obvious that $p_{e,0} + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} = 1$ and $g_e(x)$ is concave, so we can use Jensen's inequality [64] for concave functions which leads to the following expression (Suppose $C_{i,e}$ is the cost payed in the matching of phase i by e):

$$\mathbf{E}[C_{i,e}] \leq p_{e,0} \cdot g_e(0) + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot g_e(W_{w_j}) \leq$$

$$g_e\left(p_{e,0} \cdot 0 + \sum_{j=1}^{k} \frac{w_j}{W_{w_j}} \cdot p_{e,w_j} \cdot W_{w_j}\right) = g_e\left(\sum_{j=1}^{k} w_j \cdot p_{e,w_j}\right) \leq g_e\left(\sum_{j=1}^{k} w_j\right)$$

Where the last inequality holds because $p_{e,w_j} \leq 1$ and $g_e(x)$ is increasing. This argument concludes our proof.

□

Combining the aforementioned lemmas we end up with the following lemma.

Lemma 6.4.10. *The routing cost of each phase is by expectation at most the cost of the optimal solution.*

Combine this lemma with the fact that our algorithm has $O(log|S|)$ phases we get the following theorem.

Theorem 6.4.11. *The algorithm analyzed in this section produces a $O(log|S|)$-approximate solution to the CFLSC problem.*

6.5 Game Theoretic Analysis

In this section we look at the problem from a game theoretic point of view. In the original network G we add an extra sink vertex t and we connect each vertex of G with t with a cost sharing edge (i.e., an edge with latency $l_e(x) = \frac{B}{x}$). We also assume that for all other edges latencies are linear.

6.5.1 Computing Approximate Nash Equilibria

Consider the case where demands are unsplittable. We can easily find an $\frac{w_{\max}}{w_{\min}} + 1$-approximate Nash Equilibrium. If every player uses the facility located at their node then every player pays B where B is the facility cost. Any player can deviate from his strategy and pay a routing cost C_r and a new facility cost C_f. Lets call i the player that deviates and i' the player whose facility player i uses when he deviates. Players i new cost will be $C_r + C_f$ where $0 \leq C_r$ and

$C_f = \frac{w_i}{w_i + w_{i'}} * B \geq \frac{w_{min}}{w_{min} + w_{max}} * B$ thus the maximum amount that a player can benefit from his deviation is $\frac{w_{max} + w_{min}}{w_{min}} = \frac{w_{\max}}{w_{\min}} + 1$. Which means that in the unweighted version of the problem this is a 2-approximate Nash equilibrium. Also in the case where the demand is splittable and each player has an infinitesimal amount of demand this state is a $\frac{w_{\max}}{w_{\min}}$-approximate Nash equilibrium where $w_{\max}$ is the maximum amount of demand gathered at one node and $w_{\min}$ the minimum. So if all nodes have the same amount of demand/traffic this is an exact Nash equilibrium.

6.5.2 Bounding the Price of Anarchy

An interesting question is what is the price of anarchy in such games. We will provide an upper and a lower bound for this metric.

6.5.2.1 Lower Bound

Lemma 6.5.1. *The game described in section 6.5 has Price of Anarchy greater or equal than $O(n)$.*

Proof. Consider the network shown at figure 6.2. There we have k vertices $v_1, v_2, ..., v_k$, a vertex u and a vertex r. All vertices have zero latency except edge (u, r) which has $l_e(x) = a \cdot x$ latency where $a = \frac{k-1}{k^2} \cdot B - \epsilon$, B is the cost of opening a facility and $\epsilon > 0$. Also each vertex v_i has demand $w_{v_i} = 1$.

The optimal solution would be to open a facility on u. The routing cost would be 0 and the facility cost B. We will denote this optimal cost as C^*.

Now consider the case where only one facility is opened at r. Then all players have a latency of $a \cdot k + \frac{B}{k} = \frac{k-1}{k} \cdot B - \epsilon \cdot k + \frac{B}{k} = B - \epsilon \cdot k$. Thus this state is a Nash equilibrium because the only way each player could defect would be to open a facility on his vertex but then he would pay a cost $B > B - \epsilon \cdot k$. The cost of this routing is $C = a \cdot k^2 + B = (k-1) \cdot B + B = k * B$. Thus the price of anarchy of this equilibrium is $PoA = \frac{C}{C^*} = k = n - 2 = O(n)$ where n is the amount of vertecies in the network. □

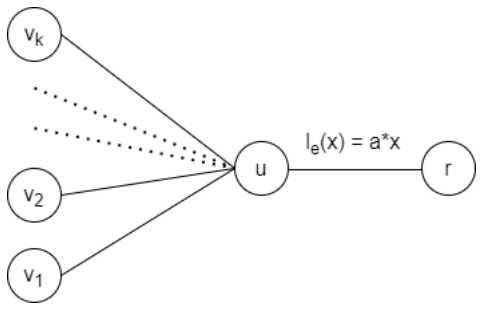


Figure 6.2: A network with $O(n)$ PoA

6.5.2.2 Upper Bound

We will also provide a matching upper bound of $O(n)$ for our game.

Lemma 6.5.2. *The game described in section 6.5 has Price of Anarchy that is at most $O(n)$.*

Proof. Consider a random Nash equilibrium in the game we are considering. The costs that we pay are either routing costs or facility costs. We will consider each node of the network separately and show that it contributes at most $2 \cdot B$ to the total cost of the equilibrium. For each vertex that has no demand on it can either have a facility opened on it or not and thus contributing at most B to the total cost. For vertices that have demand on them they contribute an amount of B cost to the total routing cost. That is, because if one demand pays more than B for its routing cost then it can always open a facility on its node and pay less cost contradicting the assumption that the flow is an equilibrium. Also a demand might or might not have a facility open on it and thus contributing at most B to the facility cost. Note that a demand might have a facility opened on it but still prefer to use another facility because of the cost sharing mechanism. Thus for each node we have bounded its total cost contribution by $2 \cdot B$. Thus the total cost of an equilibrium is at most $2 \cdot B \cdot n$. However any feasible solution opens at least one facility thus the optimal cost is at least B. Thus its holds that $PoA \leq \frac{2 \cdot B \cdot n}{B} = 2 \cdot n = O(n)$. □

From lemmas 6.5.1 and 6.5.2 we have the following theorem.

Theorem 6.5.3. *The game described in section 6.5 has a $O(n)$ Price of Anarchy.*

6.5.3 The use of Sparsification Techniques

In this section we will simplify our problem even more and at the next section we will try to provide some compelling arguments on why this is necessary. We will consider the case where we only have one splittable demand point s and the cost of opening a facility on each vertex of the graph differs. For this version of the problem we will make some extra assumptions for each one of the two sparsification techniques we will analyze. Our task will be to find as set of facilities that when opened the facility cost plus the routing cost of the resulting Nash Equilibrium is minimized. We will call this flow the optimal Nash flow.

6.5.3.1 Althöfers Lemma

Recall lemma 5.5.1. That lemma can be directly applied in our case. An important detail of that lemma will play a crucial role. Recall that in the proof we choose k paths at random and we use the flow that we want to emulate to determine the probabilities. This means that any path that is chosen must have

had a non zero amount of flow. This means that in our case for any $\epsilon > 0$, there exists a feasible flow $\hat{f}$ that assigns positive traffic to at most $\lfloor \frac{\log(2 \cdot m)}{2 \cdot \epsilon^2} \rfloor + 1$ paths, such that for any $e \in E$, $|\hat{f}_e - f_e^*| \leq \epsilon$ where f_e^* is the amount of flow passing through edge e in the optimal Nash flow.

Now suppose in our case that the amount of demand on the source is $w_s = 1$ (this can be achieved through scaling if necessary). Let $l_e(x) = a_e \cdot x + b_e$ be the latency on our network and $a = \max_{e \in E}\{a_e\}$. Lets call f^* the optimal Nash flow. Also let $d_1, d_2 > 0$ be constants such that $|P| \leq m^{d_1}$ and $|p| \leq \log^{d_2} m$ for all $p \in P$.

Using the exact same reasoning as theorem 5.5.2 we can state that we can find in $m^{O(d_1 \cdot a^2 \cdot \log^{2 \cdot d_2 + 1}(2 \cdot m)/\epsilon^2)}$ time a flow $\hat{f}$ such that $l_p(\hat{f}_p) \leq L(f^*) + \epsilon$. Where recall $L(f^*)$ is the common cost incurred by all paths in the Nash equilibrium. Obviously $\hat{f}$ is a ϵ-Nash flow. However now we are also concerned with the cost of that flow compared with the cost of f^*.

Taking inequality $l_p(\hat{f}_p) \leq L(f^*) + \epsilon$ and multiplying both parts by $\hat{f}_p$ (i.e. the amount of flow passing through path p in $\hat{f}$) we get $\hat{f}_p \cdot l_p(\hat{f}_p) \leq \hat{f}_p \cdot (L(f^*) + \epsilon)$. Summing over all p that are being used by $\hat{f}$ we get $\sum_p \hat{f}_p \cdot l_p(\hat{f}_p) \leq \sum_p \hat{f}_p \cdot (L(f^*) + \epsilon) \Rightarrow \hat{C}_r \leq (L(f^*) + \epsilon) \cdot \sum_p \hat{f}_p = L(f^*) + \epsilon = C_r^* + \epsilon$, where $\hat{C}_r$ is the total routing cost of $\hat{f}$ and C_r^* is the routing cost of f^*. As we have already mentioned $\hat{f}$ only uses facilities that f^* uses an thus $\hat{C}_f \leq C_f^*$ where $\hat{C}_f$ is the facility cost of $\hat{f}$ and C_f^* is the facility cost of f^*. Thus for the total costs it holds that $\hat{C} \leq C^* + \epsilon$. So, the following theorem holds.

Theorem 6.5.4. *For a given instance of this sections problem we can find in $m^{O(d_1 \cdot a^2 \cdot \log^{2 \cdot d_2 + 1}(2 \cdot m)/\epsilon^2)}$ time a 2ϵ-Nash flow (and the set of facilities it uses) with a total cost at most $C^* + \epsilon$ where C^* is the optimal Nash cost.*

6.5.3.2 Approximate Caratheodory's Theorem

We can achieve similar results using the Approximate Caratheodory's Theorem. Recall theorem 5.5.3 and the set of vectors X used in the proof of theorem 5.5.4. In our instance we have $x_p = (x_{p,1}, x_{p,2}, ..., x_{p,|E|}, x_{p,|E|+1}, x_{p,|E|+2}, ..., x_{p,|E|+|N|})$ where $x_{p,e} = 1$ if $e \in p$ otherwise we set it to 0 and $x_{p,|E|+v} = 1$ if p ends in facility v otherwise $x_{p,|E|+v} = 0$. Suppose we have an optimal Nash flow f^* that uses a set of paths P'. Obviously that flow lies in the convex hull of $X' = \{x_p : p \in P'\}$. Thus we can use X' and this will guarantee that and edge and a facility will be used by our result only if it gets a non negative amount of flow in f^*.

We will now assume that all latency functions are a-Lipschitz, $\max_{p \in P} |p| \leq M$ and the amount of demand on s is $w_s = 1$. Now using the same arguments as in the proof of theorem 5.5.4 we can state that there exists a flow $\hat{f}$ that uses $O(\frac{a^2 \cdot M^3}{\epsilon^2})$, is an ϵ-Nash flow and $l_p(\hat{f}) \leq L(f^*) + \epsilon$. Then using the exact same argumentation used in the previous section (6.5.3.1) we can show that $\hat{C} \leq C^* + \epsilon$ where $\hat{C}$ is the

total cost of flow $\hat{f}$ and C^* is the total cost of flow f^*. Thus the following theorem holds.

Theorem 6.5.5. *For a given instance of this sections problem we can find in* $|P|^{O\left(\frac{a^2 \cdot M^3}{\epsilon^2}\right)}$ *time a* 2ϵ*-Nash flow (and the set of facilities it uses) with a total cost at most* $C^* + \epsilon$ *where* C^* *is the optimal Nash cost and* P *is the set of all possible simple paths that begin in* s*.*

6.5.3.3 A Difficult Example

Suppose we wanted to solve our problem where we have multiple demand points and the cost of all facilities is B. First of all we can not try to use sparsification techniques for each demand point separately because this would lead us to an exhaustive search that is exponential to the amount of demand points. However its is obvious that the number of facilities is bounded by the number of demands. So if we could perform exhaustive search exponential to the amount of demands we could simple check all possible facility combination in the first place thus rendering the use of sparsification techniques useless.

Thus it becomes evident that we have to treat the flow as a whole in order for sparsification techniques to add value to our analysis. However in doing so the results return by the exhaustive search might not be a feasible solution of the problem. It might also not take into consideration many small demands and close facilities that are crucial to the optimal solution. So once we have used those techniques we then have to find set of facilities that will accommodate large amount of demands which was the exact problem we began with. To better illustrate this argument consider the following example that captures the essence of the difficulty of the problem.

If we do not use sparsification techniques the best algorithm we are aware of is the naive $O(n)$-approximate algorithm. Consider the example depicted in figure 6.3. For any n we can create $\sqrt{n}$ different problems where each one of them has $\sqrt{n}$ nodes and $\frac{r}{\sqrt{n}}$ demand (where r is the total demand of the problem). We will call these problems $P_1, P_2, ..., P_{\sqrt{n}}$. Suppose that each problem P_j has C_j^* cost in its optimal solution. We also create a node S and we connect it with each problem with a long chain of nodes. Due to the restrictions of sparsification techniques those chains can be of length at most $b_1 \cdot log^{b_2}(n)$ where b_1, b_2 are constants. This ensures that the optimal solution of the final graph is to solve optimally each problem separately. So the total number of nodes is $\sqrt{n} \cdot \sqrt{n} + \sqrt{n} \cdot b_1 \cdot log^{b_2}(n) + 1 = O(n)$. If we tried to solve each problem separately the best we could do without the use of sparsification techniques would be to use our naive approximate algorithm and end up with a total cost of $\sqrt{n} * OPT$.

Now we will examine how well do sparsification techniques perform in this general problem. Sparsification techniques will examine at most k paths and thus will open at most k facilities. For the exhaustive search of the sparsification techniques

to end in quasi-polynomial time we want $k = poly(log(n))$. Thus the sparsification technique will yield a result that addresses at most k from the $\sqrt{n}$ problems.

If we do not open any more facilities then we will have to route demand from $\sqrt{n}-k$ facilities to S and then to the k addressed problems. This obviously creates a cost that is greater than that of the naive algorithm of opening just one facility in S and routing all demands there. So it becomes apparent that we have to open new facilities.

The only possible alternative is to solve each of the $\sqrt{n}-k$ unaddressed problems separately using our naive $O(n)$-approximation algorithm. Without loss of generality suppose that $P_1, P_2, ..., P_{\sqrt{n}-k}$ are the unaddressed problems. Solving them naively we get $\sqrt{n}\cdot C_1^* + \sqrt{n}\cdot C_2^* + ... + \sqrt{n}\cdot C_{\sqrt{n}-k}^* = \sqrt{n}\cdot\sum_{j=1}^{\sqrt{n}-k} C_j^*$ cost. We will try to find the approximation ratio of our solution. Obviously we can bound from underneath the approximation ratio if we make the following assumptions:

- The k addressed problems are solved optimally.
- The cost of the k addressed problems is the maximum possible.
- The cost of the $\sqrt{n}-k$ unaddressed problems is the minimum possible.

We can observe that each problem must open at least one facility and opening one facility in each node is a possible solution (recall that B is the cost of opening a facility). Thus $B \leq C_j^* \leq \sqrt{n}\cdot B$ for all j. Thus we get that the approximation ratio of or algorithm is:

$$A = \frac{\sqrt{n}\cdot\sum_{j=1}^{\sqrt{n}-k} C_j^* + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} C_j^*}{\sum_{j=1}^{\sqrt{n}-k} C_j^* + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} C_j^*} \geq \frac{\sqrt{n}\cdot\sum_{j=1}^{\sqrt{n}-k} B + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} \sqrt{n}\cdot B}{\sum_{j=1}^{\sqrt{n}-k} B + \sum_{j=\sqrt{n}-k+1}^{\sqrt{n}} \sqrt{n}\cdot B} \Rightarrow$$

$$A \geq \frac{\sqrt{n}\cdot(\sqrt{n}-k) + \sqrt{n}\cdot k}{\sqrt{n}-k+\sqrt{n}\cdot k} = \frac{\sqrt{n}}{1-k/\sqrt{n}+k} \geq \frac{\sqrt{n}}{1+k}$$

Thus $A \geq \frac{\sqrt{n}}{1+k} \geq O(n^{\frac{1}{2}-\lambda})$ for any $\lambda > 0$. The second inequality holds since $k = poly(log(n))$. So, especially for large n we have not made any significant progress regarding the $O(\sqrt{n})$ naive algorithm we could have implemented in the first place. Also it is not obvious how we will distinguish those different problems in more difficult settings. Additionally with some hyper parameter tuning such as adjusting the total number of different problems and pose some restraints on the cost of the optimal solution of each problem P_j we can achieve a tighter bound on A. Nevertheless this analysis is simple enough to be easily understood but also perfectly illustrates our main point.

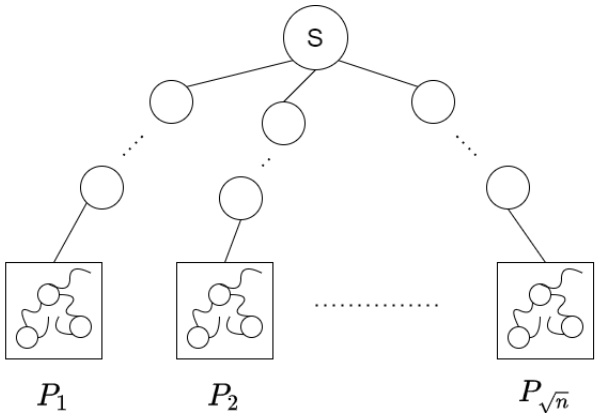


Figure 6.3: An example where sparsification techniques are rendered useless

Chapter 7

Conclusion

To conclude, some very interesting problems and ideas were analyzed in this thesis. However, there is a lot of work to be done in the field of facility location for selfish commuters. The problems we analyzed seem much more difficult than first anticipated and there seems to be no technique available in today's bibliography that can be readily used for this problem. Either new techniques must be developed or strong inapproximability results must exist. Through our experience with this problem we believe that the later case is more likely. A good place to start looking would be the counter example we provided in section 6.5.3.3. If our problem has few facilities in its optimal solution ($poly(log)$) we can find those with exhaustive search. If our problem opens an amount of facilities close to the amount of the demands then the naive algorithm of opening a facility on each demand has good approximation guarantee. It becomes obvious that the difficult cases are those that have demands comparable to the amount of nodes and facilities that are an order of magnitude less than the amount of demands. This is exactly what our counter example captures.

Bibliography

[1] Karen Aardal, Fabián A. Chudak, and David B. Shmoys. A 3-approximation algorithm for the -level uncapacitated facility location problem. *Information Processing Letters*, 72(5-6):161–167, 1999.

[2] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. On the impact of combinatorial structure on congestion games. *J. ACM*, 55(6), December 2008.

[3] Manica Aggarwal and Naveen Garg. A scaling technique for better network design. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '94, page 233–240, USA, 1994. Society for Industrial and Applied Mathematics.

[4] Ajit Agrawal, Philip Klein, and R. Ravi. When trees collide. *Proceedings of the twenty-third annual ACM symposium on Theory of computing - STOC 91*, 1991.

[5] Ingo Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and Applications*, 99:339–355, 1992.

[6] M. Andrews and L. Zhang. The access network design problem. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 40–49, 1998.

[7] Eric Angel, Nguyen Thang, and Damien Regnault. Improved local search for universal facility location. volume 29, 01 2014.

[8] E. Anshelevich, Anirban Dasgupta, Kleinberg JM, Éva Tardos, T. Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. volume 38, pages 295– 304, 11 2004.

[9] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[10] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 542–547, 1997.

[11] Baruch Awerbuch, Alan Baratz, and David Peleg. Cost-sensitive analysis of communication protocols. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, PODC '90, page 177–187, New York, NY, USA, 1990. Association for Computing Machinery.

[12] Manisha Bansal, Naveen Garg, and Neelima Gupta. A 5-approximation for universal facility location. 12 2018.

[13] Siddharth Barman. Approximating carathéodory's theorem and nash equilibria. *CoRR*, abs/1406.2296, 2014.

[14] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Proceedings of 37th Conference on Foundations of Computer Science*, 1996.

[15] Yair Bartal. On approximating arbitrary metrices by tree metrics. *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC 98*, 1998.

[16] Vincenzo Bonifaci, Tobias Harks, and Guido Schäfer. Stackelberg routing in arbitrary networks. *Lecture Notes in Computer Science Internet and Network Economics*, page 239–250, 2008.

[17] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. An improved lp-based approximation for steiner tree. STOC '10, page 583–592, New York, NY, USA, 2010. Association for Computing Machinery.

[18] Ioannis Caragiannis and Angelo Fanelli. On Approximate Pure Nash Equilibria in Weighted Congestion Games with Polynomial Latencies. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 133:1–133:12, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[19] Ioannis Caragiannis, Angelo Fanelli, Nick Gravin, and Alexander Skopalik. Computing approximate pure nash equilibria in weighted congestion games with polynomial latency functions. *CoRR*, abs/1107.2248, 2011.

[20] Ioannis Caragiannis, Angelo Fanelli, Nick Gravin, and Alexander Skopalik. Efficient computation of approximate pure nash equilibria. *CoRR*, abs/1104.2690, 2011.

[21] Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Taxes for linear atomic congestion games. *Lecture Notes in Computer Science Algorithms – ESA 2006*, page 184–195, 2006.

[22] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: Deterministic approximation algorithms for group steiner trees and k-median. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 114–123, New York, NY, USA, 1998. Association for Computing Machinery.

[23] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[24] Chandra Chekuri, Julia Chuzhoy, Liane Lewin-Eytan, Joseph Naor, and Ariel Orda. Non-cooperative multicast and facility location games. *Selected Areas in Communications, IEEE Journal on*, 25:1193 – 1206, 09 2007.

[25] Ho-Lin Chen and Tim Roughgarden. Network design with weighted players. *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures - SPAA 06*, 2006.

[26] Steve Chien and Alistair Sinclair. Convergence to approximate nash equilibria in congestion games. *Games and Economic Behavior*, 71(2):315–327, 2011.

[27] George Christodoulou, Martin Gairing, Yiannis Giannakopoulos, and Paul G. Spirakis. The price of stability of weighted congestion games. *SIAM Journal on Computing*, 48(5):1544–1582, 2019.

[28] George Christodoulou, Elias Koutsoupias, and Paul G. Spirakis. On the performance of approximate equilibria in congestion games. *Algorithmica*, 61(1):116–140, 2010.

[29] Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[30] Julia Chuzhoy, Anupam Gupta, Joseph (Seffi) Naor, and Amitabh Sinha. On the approximability of some network design problems. *ACM Trans. Algorithms*, 4(2), May 2008.

[31] Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. How much can taxes help selfish routing? *Proceedings of the 4th ACM conference on Electronic commerce - EC 03*, 2003.

[32] G. B. Dantzig, L. R. Ford, and D. R. Fulkerson. A primal-dual algorithm for linear programs. *Linear Inequalities and Related Systems. (AM-38)*, page 171–182, 1957.

[33] N. G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merive. A flexible model for resource

management in virtual private networks. *SIGCOMM Comput. Commun. Rev.*, 29(4):95–108, August 1999.

[34] Juliane Dunkel and Andreas S. Schulz. On the complexity of pure-strategy nash equilibria in congestion and local-effect games. *Mathematics of Operations Research*, 33(4):851–868, 2008.

[35] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. STOC '04, page 604–612, New York, NY, USA, 2004. Association for Computing Machinery.

[36] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. Special Issue on STOC 2003.

[37] Matthias Feldotto, Martin Gairing, Grammateia Kotsialou, and Alexander Skopalik. Computing approximate pure nash equilibria in shapley value weighted congestion games. *Web and Internet Economics Lecture Notes in Computer Science*, page 191–204, 2017.

[38] Amos Fiat, Haim Kaplan, Meital Levy, Svetlana Olonetsky, and Ronen Shabo. On the price of stability for designing undirected networks with fair cost allocations. pages 608–618, 07 2006.

[39] J.Andrew Fingerhut, Subhash Suri, and Jonathan S. Turner. Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2):287–309, 1997.

[40] L. Fleischer, K. Jain, and M. Mahdian. Tolls for heterogeneous selfish users in multicommodity networks and generalized congestion games. *45th Annual IEEE Symposium on Foundations of Computer Science*, 2004.

[41] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[42] Dimitris Fotakis. Stackelberg strategies for atomic congestion games. *Algorithms – ESA 2007 Lecture Notes in Computer Science*, page 299–310, 2007.

[43] Dimitris Fotakis, Alexis C. Kaporis, and Paul G. Spirakis. Efficient methods for selfish network design. *Theoretical Computer Science*, 448:9–20, 2012.

[44] Dimitris Fotakis, Spyros Kontogiannis, and Paul Spirakis. Selfish unsplittable flows. *Theoretical Computer Science*, 348(2):226–239, 2005. Automata, Languages and Programming: Algorithms and Complexity (ICALP-A 2004).

[45] Dimitris Fotakis and Paul G. Spirakis. Cost-balancing tolls for atomic network congestion games. *Lecture Notes in Computer Science Internet and Network Economics*, page 179–190, 2007.

[46] Harold N. Gabow, Michel X. Goemans, and David P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming*, 82(1-2):13–40, 1998.

[47] Naveen Garg, Rohit Khandekar, Goran Konjevod, R. Ravi, F. S. Salman, and Amitabh Sinha. On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design problem. page 170–184, Berlin, Heidelberg, 2001. Springer-Verlag.

[48] Yiannis Giannakopoulos, Georgy Noarov, and Andreas S. Schulz. An improved algorithm for computing approximate equilibria in weighted congestion games. *CoRR*, abs/1810.12806, 2018.

[49] Yiannis Giannakopoulos and Diogo Poças. A unifying approximate potential for weighted congestion games. *CoRR*, abs/2005.10101, 2020.

[50] M. Goemans, A.V. Goldberg, Serge Plotkin, David Shmoys, Éva Tardos, and D. Williamson. Improved approximation algorithms for network design problems. *Proceedings of the Annual ACM SIAM Symposium on Discrete Algorithms*, 01 2001.

[51] M. Goemans, Vahab Mirrokni, and A. Vetta. Sink equilibria and convergence. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 142–151, 2005.

[52] Michel X. Goemans and David P. Williamson. .879-approximation algorithms for max cut and max 2sat. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, page 422–431, New York, NY, USA, 1994. Association for Computing Machinery.

[53] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[54] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, page 603, USA, 2000. IEEE Computer Society.

[55] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[56] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation for the single sink edge installation problems. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 383–388, New York, NY, USA, 2001. Association for Computing Machinery.

[57] Anupam Gupta, Jon Kleinberg, Amit Kumar, Rajeev Rastogi, and Bulent Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. pages 389–398, 01 2001.

[58] Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 365–372, New York, NY, USA, 2003. Association for Computing Machinery.

[59] Christoph Hansknecht, Max Klimm, and Alexander Skopalik. Approximate Pure Nash Equilibria in Weighted Congestion Games. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 242–257, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[60] Tobias Harks and Max Klimm. On the existence of pure nash equilibria in weighted congestion games. *Automata, Languages and Programming Lecture Notes in Computer Science*, page 79–89, 2010.

[61] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[62] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM*, 50(6):795–824, 2003.

[63] Kamal Jain and Vijay Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of The ACM - JACM*, 48, 03 2001.

[64] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906.

[65] Raja Jothi and Balaji Raghavachari. Improved approximation algorithms for the single-sink buy-at-bulk network design problems. *Journal of Discrete Algorithms*, 7, 05 2004.

[66] G. Karakostas and S.g. Kolliopoulos. Edge pricing of multicommodity networks for heterogeneous selfish users. *45th Annual IEEE Symposium on Foundations of Computer Science*, 2004.

[67] George Karakostas and Stavros G. Kolliopoulos. Stackelberg strategies for selfish routing in general multicommodity networks. *Algorithmica*, 53(1):132–153, 2009.

[68] David Karger and Michael Minkoff. Building steiner trees with incomplete global knowledge. pages 613–623, 02 2000.

[69] Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning and shortest path trees. *CoRR*, cs.DS/0205045, 2002.

[70] Philip N. Klein and R. Ravi. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *IPCO*, 1993.

[71] Y.a. Korilis, A.a. Lazar, and A. Orda. Achieving network optima using stackelberg routing strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997.

[72] Guy Kortsarz and David Peleg. Approximating the weight of shallow steiner trees. *Discrete Applied Mathematics*, 93(2):265–285, 1999.

[73] Madhukar R. Korupolu, C.Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.

[74] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. pages 404– 413, 1999.

[75] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[76] Henry Lin, Tim Roughgarden, Éva Tardos, and Asher Walkover. Stronger bounds on braess's paradox and the maximum latency of selfish routing. *SIAM J. Discrete Math.*, 25:1667–1686, 01 2011.

[77] Mohammad Mahdian and Martin Pal. Universal facility location. *Springer, Berlin, Heidelberg*, 2003.

[78] Madhav V Marathe, R Ravi, Ravi Sundaram, S.S Ravi, Daniel J Rosenkrantz, and Harry B Hunt. Bicriteria network design problems. *Journal of Algorithms*, 28(1):142–171, 1998.

[79] Kamesh Munagala, Serge Plotkin, and Adam Meyerson. Cost-distance: Two metric network design. *Annual Symposium on Foundations of Computer Science - Proceedings*, 38, 01 2001.

[80] Vinayaka Pandit. *Local Search Heuristics For Facility Location Problems*. PhD thesis, Department of Computer Science and Engineering Indian Institute of Technology Delhi, 2004.

[81] Ravi and Williamson. Erratum: An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 34(1):98–107, 2002.

[82] Robert W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, pages 65–67, 1973.

[83] Tim Roughgarden. Designing networks for selfish users is hard. pages 472–481, 11 2001.

[84] Tim Roughgarden. Stackelberg scheduling strategies. *Proceedings of the thirty-third annual ACM symposium on Theory of computing - STOC 01*, 2001.

[85] Tim Roughgarden. The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences*, 67(2):341–364, 2003. Special Issue on STOC 2002.

[86] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.

[87] L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, 2016.

[88] D. B. Shmoys, E. Tardos, and K.I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[89] Alexander Skopalik and Berthold Vöcking. Inapproximability of pure nash equilibria. STOC '08, page 355–364, New York, NY, USA, 2008. Association for Computing Machinery.

[90] Chaitanya Swamy and Amit Kumar. Primal–dual algorithms for connected facility location problems. *Algorithmica*, 40:245–269, 01 2004.

[91] Kunal Talwar. The single-sink buy-at-bulk lp has constant integrality gap. volume 2337, pages 475–486, 05 2002.

[92] Jens Vygen. From stars to comets: Improved local search for universal facility location. *Oper. Res. Lett.*, 35:427–433, 07 2007.

[93] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.