Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τεχνολογία Πληροφορική & Υπολογιστών

Ελάχιστοι και Μέγιστοι Περιορισμοί στο Πρόβλημα του Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΑΜΕΡΑΝΗΣ

Επιβλέπων : Δημήτριος Φωτάκης
Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάιος 2018

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τεχνολογία Πληροφορική & Υπολογιστών

Ελάχιστοι και Μέγιστοι Περιορισμοί στο Πρόβλημα του Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΑΜΕΡΑΝΗΣ

Επιβλέπων : Δημήτριος Φωτάκης
Επ. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 25η Μαΐου 2018.

Δημήτριος Φωτάκης Επ. Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Αν. Καθηγητής Ε.Μ.Π.	Αριστείδης Παγουρτζής Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάιος 2018

Κωνσταντίνος Αμεράνης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ένα δύσκολο πρόβλημα για κάθε τουρίστα είναι αφού φτάσει στον προορισμό του να αποφασίσει ποια από τα πολλά αξιοθέατα της πόλης αξίζει να επισκεφτεί κατά την διάρκεια της σύντομης διαμονής του. Στην βιβλιογραφία αυτό αναφέρεται ως το πρόβλημα του προσανατολισμού. Αλλιώς έχει αναφερθεί και ως το πρόβλημα του περιπλανώμενου πωλητή με περιορισμούς, καθώς ο τουρίστας δεν προλαβαίνει να πάει σε όλα τα αξιοθέατα, αλλά μόνο σε μερικά. Τα τελευταία χρόνια έχουν δημοσιοποιηθεί πολλές προσεγγίσσεις σε αυτό το πρόβλημα που παίρνουν υπόψιν τους περισσότερα στοιχεία από την πραγματική ζωή. Για παράδειγμα, πόσο χρόνο θα χρειαστεί κανείς στις μετακινήσεις ή για να επισκεφτεί ένα αξιοθέατο, ή πόσο πολύ θα ευχαριστηθεί ένα συγκεκριμένο αξιοθέατο. Μια άλλη προσέγγιση περιέχει περισότερους από έναν περιορισμούς είναι για παράδειγμα να πάρεις κανείς υπόψιν του όχι μόνο τον χρόνο που θα χρειάστεί αλλά και πόσο κοστίζει η επίσκεψη σε κάθε αξιοθέατο.

Στην διπλωματική αυτή θα εισάγουμε άλλον έναν περιορισμό. Ο περιορισμός αυτός είναι ότι για κάθε είδος αξιοθεάτου πρέπει να έχουμε ένα συγκεκρίμμενο εύρος στο πλήθος τους. Για παράδειγμα, κάθε μέρα θέλουμε να επισκεφοτύμε ένα εστιατόριο. Αν δεν επισκεφτούμε κανένα, τότε θα πεινάσουμε. Από την άλλη, η επίσκεψη σε ένα δεύτερο εστιατόριο δεν θα μας προσφέρει κάτι.

Αρχικά θα παρουσιάσουμε την ιστορία του προβλήματος, τις διαφορετικές βαριάντες που έχουν εμφανιστεί στην βιβλιογραφία, καθώς και τις προτεινόμενες λύσεις. Στην συνέχεια θα ξεφύγουμε από την αόριστη γλώσσα του κειμένου και θα ορίσουμε το πρόβλημα αυστηρά με την βοήθεια των μαθηματικών. Θα παρουσιάσουμε το πρόβλημα με την βοήθεια του ακέραιου προγραμματισμού και θα εξηγήσουμε από που προέρχεται κάθε περιορισμός στην διαμόρφωση του προβλήματος.

Στη συνέχεια, θα αναλύσουμε περαιτέρω τον αλγόριθμο πάνω στον οποίο έχουμε βασιστεί, περιγράφωντας το πως δουλεύει και τις προκλήσεις που συνάντησαν οι συγγραφείς. Στο ίδιο κεφάλαιο θα περιγράψουμε και τα datasets με τα οποία θα δουλέψουμε και πάνω στα οποία θα μετρήσουμε την επιτυχία μας. Έχοντας περιγράψει τον προηγούμενο αλγόριθμο, θα περιγράψουμε τις δικές μας τροποποιήσεις και πως αυτές επηρεάζουν την λύση. Αφού αναλύσουμε την επίδραση κάθε μεταβλητής με εκτενή πειραματισμό, θα παρουσιάσουμε τα αποτελέσματα που αποκτήσαμε χρησιμοποιώντας αυτόν τον αλγόριθμο πάνω στα γνωστά datasets. Θα συγκρίνουμε τα αποτελέσματα αυτά με προηγούμενες προσπάθειες πάνω στο πρόβλημα χωρίς περιορισμούς και τελικά θα δούμε πως επηρεάζεται το αποτέλεσμα όταν θέτουμε περισσότερους περιορισμούς.

Λέξεις κλειδιά

περιήγηση, περιορισμοί, χρονικά παράθυρα, πρόβλημα σχεδιασμού τουριστικού ταξιδιού, ILS, μηχανική μάθηση, 2-OPT, ευρετικές συναρτήσεις, τυχαιοποίηση, εισαγωγή

Abstract

A difficult problem for every tourist is after reaching his destination to decide which of the many sights of the city are worth visiting during her short visit to the city. In the academic bibliography this is referred to as the orientation problem. Alternatively, it has been referred to as the constrained travelling salesman problem, the main difference from the regural TSP being that the salesman (or rather the tourist in this case) cannot visit all the sights, but only go to a few. In recent years, many approaches to this problem have been published that take into account most of the facts from real life. For example, how long you will need to travel or to visit an attraction, or how much a particular sight will please the visitor. Another approach contains more than one constraints. For example, except for the time one needs to also take into account how much it costs to visit each attraction.

In this diploma thesis, we will introduce another restriction. This constraint is that for each kind of attraction we must visit a certain amount of them. For example, each day we want to visit a restaurant. If we do not visit any, then we will be hungry. On the other hand, visiting a second restaurant will not offer us anything.

Initially we will present the history of the problem, the different variants that have appeared in the bibliography, as well as the proposed solutions. Then we will escape from the vague language of text and we will define the problem strictly with the help of mathematics. We will present the problem with the help of integer programming and will explain where each limitation of the problem is derived from.

Then, we will further analyze the algorithm on which we have based our own, describing how it works and the challenges faced by the writers. In the same chapter we will also describe the datasets with which we will work and on which we will measure our success. Having described the previous algorithm, we will describe our modifications and how they affect the solution. Analyzing the effect of each variable with extensive experimentation, we present the results we obtained using this algorithm on the aforementioned datasets. We will compare these results with previous attempts on the problem without limitations and finally we will see how the final result is affected when we set more constraints.

Key words

orienteering, constraints, time windows, tourist trip design problem, ILS, machine learning, 2-OPT, heuristic, randomization, insert, shake

Ευχαριστίες

Θα ήθελα να αφιερώσω αυτή την διπλωματική σε όλους τους ανθρώπους που βοήθησαν στην ολοκλήρωσή της, είτε άμεσα συνεισφέροντας σε αυτήν είτε έμμεσα, βοηθώντας με σε όλη την διάρκεια των φοιτητικών μου χρόνων. Πάνω από όλους θα ήθελα να ευχαριστήσω τον επιβλέποντα μου Δημήτρη Φωτάκη για την συνεχή υποστήριξή του, τις ιδέες τις οποίες προσέφερε κάθε φορά που δεν ήξερα πως να συνεχίσω και την καθοδήγησή του, όχι μόνο στα πλαίσια της διπλωματικής, αλλά ολόκληρης της ακαδημαϊκής μου ζωής. Μου δείξατε όχι μόνο αλγορίθμους αλλά και πως συμπεριφέρεται ένας καλός παιδαγωγός. Ο άλλος μεγάλος παράγοντας στην διαμόρφωση αυτής της διπλωματικής είναι ο Νίκος Βάθης, αλλιώς γνωστός ως Soft Silverwind, ο οποίος παρόλο που ταυτόχρονα δούλευε και ασχολιόταν με το δικό του διδακτορικό, πάντα είχε χρόνο για να συζητήσουμε ιδέες, κώδικα ή την μορφή του κειμένου. Χωρίς εσένα αυτή η διπλωματική θα χρειαζόταν άλλον έναν χρόνο αν όχι περισσότερο.

Θα ήθελα να ευχαριστήσω τον Νικόλαο Παπασπύρου, τον πρώτο μου καθηγητή στον προγραμματισμό, ο οποίος μαζί με τους Νίκο Βάθη, Παναγιώτη Αρώνη και Νικόλα Κορασίδη μου έμαθαν να προγραμματίζω και πως να φτιάχνω διασκεδαστικά πράγματα με υπολογιστές. Κατά την διάρκεια των χρόνων που πέρασα στο ΕΜΠ είχα την τύχη να γνωρίσω πολλούς ανθρώπους που με βοήθησαν ο καθένας με τον δικό του τρόπο. Θέλω να ευχαριστήσω το Κουαρτέτο Άντα, Ελπίδα, Δήμητρα, Ελένη για όλες τις όμορφες στιγμές που περάσαμε μαζί, τον Γιάννη, τον Γιώργο και τον Μηνά για όλες τις συζητήσεις, την Νικολέττα, την Μαριλένα, τον Παναγιώτη και τον Μάριο που πάντα μου κρατούσαν παρέα και τους Χρήστο, Γιώργο, Νίκο, Αργύρη, Βασίλη και Γιάννη για τον καιρό που περάσαμε μαζί. Κάθε ένας από αυτούς με έχει επηρεάσει με τον δικό του τρόπο και με έχει βοηθήσει να αναπτυχθώ.

Φυσικά από αυτή την λίστα δεν θα μπορούσε να λείπει η οικογένεια μου, η οποία με στήριξε σε κάθε βήμα της ζωής μου, σε κάθε εμπόδιο, σε κάθε ευκαιρία, σε κάθε δυσκολία και σε κάθε ευκαιρία. Στον αδερφό μου Βασίλη που πάντα ήταν και είναι ένα πρότυπο για μένα, τον αδερφό μου Ιάσονα με τον οποίο μοιραζόμαστε τόσο πολλά, τον πατέρα μου Χρήστο που πάντα με πίεζε για να είμαι ο καλύτερος εαυτός μου και την μητέρα μου Έφη που ήταν πάντα εκεί για μένα σε κάθε λύπη και κάθε χαρά. Τέλος, θα ήθελα να ευχαριστήσω την Νιόβη η οποία τα τελευταία χρόνια είναι το στήριγμά μου και το άτομο με το οποίο μπορώ να μοιραστώ κάθε στιγμή.

Κωνσταντίνος Αμεράνης,

Αθήνα, 25η Μαΐου 2018

Contents

List of Tables

List of Figures

Κεφάλαιο 1

Κείμενο στα Ελληνικά

1.1 Εισαγωγή

1.1.1 Το πρόβλημα του προσανατολισμού στην ακαδημαϊκή βιβλιογραφία

Στην επιστήμη των υπολογιστών το πρόβλημα του προσανατολισμού (Orienteering Problem) ορίζει N σημεία που το καθένα προσφέρει ένα διαφορετικό κέρδος και ένα κόστος για να πας από κάθε σημείο σε κάθε άλλο. Στόχος μας είναι να να βρούμε ένα μονοπάτι από ένα αρχικό σημείο σε ένα τελικό σημείο του οποίου το κόστος δεν ξεπερνάει μια συγκεκριμένη τιμή και μας δίνει το μέγιστο δυνατό κέρδος. Σκεφτείτε έναν φίλο που επισκέπτεται την πόλη σας και σας ζητάει να του προτείνετε που να πάει. Αποφασίζει να βγαίνει κάθε μέρα στην 9:00πμ και να γυρνάει στο ξενοδοχείο του στις 8:00μμ. Στην πόλη σας υπάρχουν πολλά ενδιαφέρονται μέρη, αλλά δεν έχει τον χρόνο να τα επισκεφτεί όλα. Ήδη έχετε ξεκινήσει μια λίστα με τα μέρη που απλά πρέπει να επισκεφτεί και μαζί με αυτά σκέφτεστε μέρη που είναι κοντά σε αυτά και πιστεύετε ότι θα του άρεσαν. Στην συνέχεια προσπαθείτε να τα βάλετε σε μια σειρά, βάζοντας τα σε μέρες και με τι σειρά θα ήταν καλό να τα επισκεφτεί. Ήδη λύνετε μια μορφή του προβλήματου του προσανατολισμού! Αν και για τον υποθετικό σας φίλο έχετε σκεφτεί θεματικές μέρες και ίσως ένα μονοήμερο σε κάποιον κοντινό προορισμό, το αρχικό πρόβλημα ήταν διαμορφωμένο αρκετά πιο απλά.

Το ΠτΠ εμφανίστηκε πρώτη φορά σαν το πρόβλημα του περιπλανώμενου πωλητή με περιορισμούς (Constrained Travelling Salesman Problem). Στο απλό πρόβλημα του περιπλανώμενου πωλητή πρέπει να επισκεφτούμε όλα τα μέρη στον ελάχιστο δυνατό χρόνο. Από την άλλη, στο ΠτΠ υπάρχει ένα όριο στον χρόνο που έχουμε για επισκέψεις. Στην επιστήμη των υπολογιστών το ΠτΠ ορίζεται ως ένα πρόβλημα βελτιστοποίησεις το οποίο επικεντρώνεται στο ποια μέρη πρέπει να επισκεφτεί κανείς και με ποια σειρά. Πέρα από την προφανή εφαρμογή του προβλήματος στο ομώνυμο άθλημα, υπάρχουν πολλά πρακτικά προβλήματα που μπορούν να αναχθούν στο πρόβλημα του προσανατολισμού.

Στα τελευταία τριάντα χρόνια το πρόβλημα του προσανατολισμού έχει εξελιχθεί από ένα σχετικά άσημο πρόβλημα σε ένα κεντρικό ερευνητικό ενδιαφέρον στο πεδία της διακριτής βελτιστοποίησης. Ενώ ξεκίνησε ως ένα αρκετά θεωρητικό πρόβλημα στην διασταύρωση της θεωρίας γράφων και της βελτιστοποίησης, τα τελευταία χρόνια πολύ πιο πρακτικές παραλλαγές έχουν αναλυθεί τα τελευταία χρόνια. Καθώς το πρόβλημα του προσανατολισμού οριμάζει, οι λύσεις στις διάφορες παραλλαγές του γίνονται όλο και πιο κατάλληλες για εφαρμογές στην πραγματική ζωή αφού σταδιακά όλο και περισσότεροι περιορισμοί από την πραγματική ζωή έχουν εισαχθεί στον ορισμό του. Ένα παράδειγμα μιας τέτοιας ιδιοτροπίας αποτελεί η χρονική φύση του υποκείμενου γράφου, είτε στις επιλογές που μπορούμε να κάνουμε κάθε στιγμή, είτε στον χρόνο που απαιτείται για την μετάβαση από το ένα μέρος στο επόμενο, όπως συμβαίνει και σε ένα φυσικό δίκτυο μεταφορών.

Λύσεις στο πρόβλημα του προσανατολισμού συμπεριλαμβάνουν πολλά προβλήματα της επιχειρησιακής έρευνας, όπως λογιστικά προβλήματα και προβλήματα δρομολόγησης οχημάτων. Άλλες εφαρμογές αποτελούν το πρόβλημα σχεδίασης ενός τουριστικού ταξιδιού (το οποίο αναφέραμε πριν από μερικές παραγράφους), το να αποφασίσει κανείς πως θα περάσει τον χρόνο του σε ένα θεματικό πάρκο, η δρομολόγηση drones που μαζεύουν πληροφορίες και το πρόβλημα του λη-

στή τραπεζών, όπου πρέπει να αποφασίσεις ποιες τράπεζες θα ληστέψεις πριν τελειώσει η βενζίνη στο αμάξι σου (δεν παίρνουμε υπόψιν την αστυνομία).

Σε αυτή την Διπλωματική παρουσιάζουμε μια νέα παραλλαγή αυτού του προβλήματος, το πρόβλημα του προσανατολισμο με χρονικά παράθυρα, ελάχιστους και μέγιστους περιορισμούς. Αυτή η νέα παραλλαγή προέρχεται από το πραγματικό παραδείγμα του προβλήματος σχεδίασης τουριστικού ταξιδιού, όπου κάθε μέρα ο τουρίστας θέλει να περάσει όσον το δυνατόν καλύτερα και να κερδίσει όσον το δυνατόν περισσότερα από την σύντομη επισκεψή του, ενώ ταυτόχρονα ικανοποιεί κάποιους περιορισμούς. Ένας τέτοιος περιορισμός είναι να επισκεφτεί ένα εστιατόριο. Αν δεν επισκεφτεί κανένα εστιατόριο, θα πεινάσει. Από την άλλη, μια δεύτερη επίσκεψη σε εστιατόριο δεν θα του προσφέρει επιπλέον ευχαρίστηση.

Ένας τρόπος για να λυθεί αυτή η νέα παραλλαγή είναι μέσω μιας Επαναληπτικής Τοπικής Έρευνας (ΕΤΕ). Θα κάνουμε μια εκτενή πειραματική έρευνα για να αποφασίσουμε τις τιμές για τις διάφορες παραμέτρους. Τα αποτελέσματα αυτής της υλοποίσης θα συγκριθούν με αλγορίθμους τελευταίας τεχνολογίας στις ίδιες τοπολογίες. Τελικώς, θα παρουσιαστεί μια υπόθεση για άλλους τρόπους επίλυσης και πιθανοί μελλοντικοί αλγόριθμοι.

1.1.2 Κίνητρο

Όπως έχουμε ήδη δείξει, πολλά προβλήματα μπορούν να αναχθεί στο ΠτΠ. Έχουμε ήδη αναφέρει το πρόβλημα σχεδίασης τουριστικού ταξιδιού. Άλλες εφαρμογές συμπεριλαμβάνουν η καθοδήγηση τεχνικών, όπου μια εταιρεία πρέπει να αποφασίσει που θα στείλει κάθε τεχνικό μέσα στην μέρα. Άλλα παραδείγματα είναι η επιλογή παιχνιδιών σε ένα λούνα παρκ και η καθοδήγηση drones πάνω από πιθανούς στόχους. Για τον λόγο αυτό, λύσεις στο ΠτΠ αποτελούν το αντικείμενο εξαιρετικού ενδιαφέροντος τόσο στην επιστημονική όσο και στην επαγγελματική κοινότητα. Μέχρι τώρα, θεωρούνταν ότι όλα τα σημεία είναι της ίδιας κατηγορίας και συνεπώς μπορούσαμε να ανταλλάξουμε οποιοδήποτε για κάποιο άλλο. Στην πραγματική ζωή όμως, μπορεί να θέλουμε να συμπεριλάβουμε ένα σημείο μιας συγκεκριμένης κατηγορίας. Το παράδειγμα που δώσαμε παραπάνω με το ένα εστιατόριο δεν είναι μοναδικό, καθώς θα μπορούσε να είναι οτιδήποτε, όπως ένα μουσείο ή μια συναυλία.

Ενώ οι μέγιστοι περιορισμοί έχουν μελετηθεί εις βάθος σε πολλά papers και σε πολλές παραλλαγές, οι ελάχιστοι περιορισμοί έχουν λίγο στο Sylejmani et al. (2012), δεν υπάρχει άλλη προηγούμενη δουλειά που να μελετάει αυτό το πρόβλημα.

1.1.3 Συνεισφορά διπλωματικής διατριβής

Οι κύριες συνεισφορές αυτής της εργασίας είναι οι επόμενες:

1. Σχεδιασμός και εφαρμογή ενός αλγορίθμου ILS για τη λύση του MMCTOPTW.
2. Σύγκριση με υπάρχοντες αλγόριθμους στη περίπτωση χωρίς περιορισμούς.
3. Σύγκριση της περίπτωσης με περιορισμούς με την απλή.

1.1.4 Περίγραμμα Κεφαλαίων

Στο κεφάλαιο 1.2, παρουσιάζουμε την ιστορία, τη βιβλιογραφία και το απαραίτητο τεχνικό υπόβαθρο που απαιτείται για την κατανόηση του ΠτΠ. Παρουσιάζονται μερικές από τις πιο γνωστές παραλλαγές μαζί με μια επισκόπηση των προτεινόμενων λύσεων. Στο τέλος παρουσιάζουμε την μαθηματική διατύπωση για το Πρόβλημα του Προσανατολισμού και της παραλλαγής που θα αποτελέσει το επίκεντρο για το υπόλοιπο της διατριβής, το πρόβλημα των ελαχίστων και μεγίστων περιορισμών στο πρόβλημα του ομαδικού προσανατολισμού με χρονικά παράθυρα.

Στο κεφάλαιο 1.3, εξετάζουμε προσεκτικά την προσέγγιση του Vansteenwegen et al. (2009b), στο έργο του οποίου αυτή η διατριβή βασίζεται σε μεγάλο βαθμό. Πρώτον, θα εξετάσουμε προσεκτικά τη μεθοδολογία που ακολουθήθηκε στο προαναφερθέν paper και πώς αυτή η διατριβή

επεκτείνη αυτή την προσέγγιση. Στη συνέχεια παρουσιάζουμε τον προτεινόμενο τροποποιημένο αλγόριθμο.

Στο κεφάλαιο 1.4 μέσω πειραματισμών σε διαθέσιμα σύνολα δεδομένων, προσδιορίζουμε τις βέλτιστες παραμέτρους που εμφανίζονται στον αλγόριθμό μας, προκειμένου να επιτευχθεί το καλύτερο δυνατό αποτέλεσμα. Οι ερωτήσεις που προκύπτουν αφορούν το στοχαστικό έναντια στο ντετερμινιστικό, το άπληστο κριτήριο που πρέπει να χρησιμοποιηθεί, αν η χρήση κινήσεων 2-OPT θα μας δώσει καλύτερο αποτέλεσμα και τέλος την επιλογή του βάρους που πρέπει να δώσουμε στο νέο κομμάτι του άπληστου κριτηρίου. Πρώτον, δείχνουμε ότι για ένα δεδομένο βάρος οι κατηγορίες κάθε ελάχιστου ακολουθούν κανονική κατανομή. Στην συνέχεια δείχνουμε ότι μια στατική τιμή αυτού του βάρους δεν είναι βέλτιστη και τελικά διερευνούμε μια μορφή Stochastic Gradient Descent (SGD) για να φτάσουμε σε μια βέλτιστη τιμή, ταυτόχρονα με την επίλυση του προβλήματος.

Στο κεφάλαιο 1.5, θα συγκρίνουμε πρώτα τον αλγόριθμό μας στο σενάριο χωρίς περιορισμούς έναντι της λύσης ILS στην οποία βασίζεται η εργασία μας. Αργότερα στο κεφάλαιο θα δειξουκμε την ομαλή μείωση του επιτευχθέντος αποτελέσματος καθώς οι περιορισμοί γίνονται αυστηρότεροι.

Στο κεφάλαιο 1.6, αναφέρουμε τις μελλοντικές εργασίες που θα μπορούσαν να γίνουν πάνω στο πρόβλημα και τους νέους τρόπους προσέγγισης. Επιπλέον, προτείνουμε επαυξήσεις στην τρέχουσα προσέγγιση και άλλες παραλλαγές παρόμοιες με την συγκεκριμένη.

1.2 Ιστορία του προβλήματος και βασικές πληροφορίες

1.2.1 Ιστορία

Το Πρόβλημα του Προσανατολισμού (OP) περιγράφηκε για πρώτη φορά στο Golden et al. (1981), αλλά ο όρος OP εισήχθη για πρώτη φορά στο Tsiligirides (1984). Στη συνέχεια, στο Golden et al. (1987) αποδείχθηκε ότι το OP είναι NP-hard. Το OP έχει προκύψει από το TSP και πολλά έγγραφα αναφέρονται σε αυτό ως Constrained TSP (Laporte and Martello (1990), Gendreau et al. (1998a), Thomadsen and Stidsen (2003)). Μια πολύ μικρή παραλλαγή του OP είναι το Team OP (TOP). Σε αυτήν την παραλλαγή, αντί να αναζητήσουμε μόνο μία, ζητούνται M διαδρομές, κάθε μία από τις οποίες μπορεί να ξεκινά και να τελειώνει στις ίδιες κορυφές, αλλά να δεν μπορεί να μοιράζεται άλλους κόμβους. Ορισμένες από τις πρακτικές εφαρμογές του OP περιλαμβάνουν τον περιπλανώμενο πωλητή με ανεπαρκή χρόνο για να επισκεφτεί όλες τις πόλεις της πόλης και το πρόβλημα της παράδοσης καυσίμων στο οποίο ένας στόλος φορτηγών πρέπει να παραδώσει καύσιμα σε έναν μεγάλο αριθμό πελατών σε καθημερινή βάση και η στάθμη του αποθέματος καυσίμων κάθε πελάτη πρέπει να διατηρείται σε κατάλληλο επίπεδο ανά πάσα στιγμή. Ως εκ τούτου, κάθε πελάτης απονέμεται βαθμολογία ανάλογα με το τρέχον ή το προβλεπόμενο επίπεδο αποθέματος καυσίμων. Ο προσδιορισμός του υποσυνόλου των πελατών για την εξυπηρέτηση σε κάθε μέρα και η διαδρομή των φορτηγών είναι ουσιαστικά το OP. Άλλες εφαρμογές περιλαμβάνουν τον Οδηγό Σχεδιασμού Τουριστών Ταξιδιών (Thomadsen and Stidsen (2003); Vansteenwegen and Van Oudheusden (2007)) Για το αρχικό OP, υπάρχουν αρκετά δεδομένα συγκριτικής αξιολόγησης, τα οποία μπορούν να βρεθούν στα Tsiligirides (1984), Chao et al. (1993), Chao et al. (1996b) και Fischetti et al. (1998). Για το TOP υπάρχουν πολλά ακόμα τα οποία μπορούν να βρεθούν στο Chao et al. (1996a) και Dang et al. (2013b). Συνολικά, υπάρχουν διαθέσιμα 385 παραδείγματα με 21 έως 500 κορυφές για το OP και 472 παραδείγματα με 21 έως 399 κορυφές. Περισσότερες πληροφορίες μπορούν να βρεθούν στο A.1

Για την αντιμετώπιση του OP έχουν προταθεί αρκετοί ακριβείς αλλά και ευριστικοί αλγόριθμοι. Οι ακριβείς λύσεις μπορούν να βρεθούν στον πίνακα A και τις ευριστικές λύσεις στον πίνακα A. Οι ακριβείς λύσεις περιλαμβάνουν αλγορίθμους Branch and Bound (Laporte and Martello (1990), Ramesh et al. (1992)), Branch and Cut (Fischetti et al. (1998), Gendreau et al. (1998a), Feillet et al. (2005), Dang et al. (2013a)), Branch and Price (Boussier et al. (2007)) και Cutting Plane (Leifer and Rosenwein (1994)). Κάθε ένα από αυτά τα papers πειραματίζεται στην εξεύρεση των βέλτιστων λύσεων καθώς ξεδιπλώνουμε τον χώρο αναζήτησης.

Από την άλλη πλευρά οι ευριστικές μέθοδοι κυριαρχούν το πεδίο τα τελευταία χρόνια. Ενώ πριν από το 2000 παρατηρούμε ευριστικές μεθόδους κατασκευασμένες συγκεκριμένα για αυτό το πρόβλημα όπως το Centre of Gravity (Golden et al. (1987)), την ευριστική τεσσάρων φάσεων (Ramesh and Brown (1991)) ή την ευριστική πέντε βημάτων (Chao et al. (1996b), Chao et al. (1996a)), τα τελευταία χρόνια παρατηρείται η εφαρμογή γνωστών ευριστικών παραλλαγμένων για να ταιριάξουν στην φύση του προβλήματος. Τα πιο γνωστά papers περιλαμβάνουν Greedy Randomized Adaptive Search Procedure, Tabu Search, πολλές διαφορετικές παραλλαγές του Variable Neigborhood Search, Iterated Local Search και φυσικά συνδιασμούς αυτών, όπως είναι ο αλγόριθμος Simulated Annealing with Iterated Local Search. Φυσικά η λίστα δεν θα μπορούσε να είναι πλήρης και με μερικούς αλγορίθμους με νοημοσύνη σμήνους όπως είναι το Ant Colony System και Particle Swarm Optimization. Μια πλήρη λίστα των αλγορίθμων και των δημοσιεύσεων όπου παρουσιάστηκαν φαίνεται στον πίνακα A.

1.2.2 Παραλλαγές

Με τον καιρό έχουν εμφανιστεί πολλές παραλλαγές, κάποιες εμπνευσμένες από την πραγματική ζωή και άλλες αμφισβητώντας προηγούμενες υποθέσεις. Οι πιο γνωστές μεταξύ αυτών είναι οι ακόλουθες:

- (Ομαδικός) Προσανατολισμός με Χρονικά Παράθυρα - (Team) Orienteering Problem with Time Windows [(T)OPTW]
- Χρονικά Εξαρτημένος Προσανατολισμός - Time Dependent Orienteering Problem [TDOP]
- Τοξωτός Προσανατολισμός - Arc Orienteering Problem [AOP]
- Στοχαστικός Προσανατολισμός - Stochastic Orienteering Problem [SOP]
- Γενικός Προσανατολισμός - General Orienteering Problem [GOP]

1.2.2.1 (Ομαδικός) Προσανατολισμός με Χρονικά Παράθυρα

Στο πρόβλημα αυτό κάθε ανατίθεται σε κάθε κόμβο ένα χρονικό παράθυρο $[O_i, C_i]$ και μια επίσκεψη σε ένα κόμβο μπορεί να ξεκινήσει μόνο μέσα σε αυτό το χρονικό παράθυρο. Από γραφοθεωρητική σκοπιά αντικαθιστούμε τον στατικό γράφο με έναν χρονικό (temporal) γραφό. Αυτή η παραλλαγή προέρχεται κυρίως από την πραγματική ζωή όπου πολλά αξιοθέατα έχουν ώρες κοινού μέσα στις οποίες μπορεί κανείς να τα επισκεφτεί. Αυτή η μικρή διαφορά έχει μεγάλη επίδραση στις μεθόδους επίλυσης. Ενώ στο παραδοσιακό (T)OP οι περισσότερες λύσεις χρησιμοποιούν κινήσεις 2-OPT, στο (T)OPTW αυτή η μέθοδος δεν μπορείς να εφαρμοστεί αποτελεσματικά. Η ανταλλαγή δύο επισκέψεων μπορεί να οδηγήσει μία ή και τις δύο να προγραματισθούν εκτός του παραθύρου τους. Επιπλέον, ενώ στο (T)OP χρειαζόταν να κρατήσουμε υπόψιν μόνο μία μεταβλητή για τον πόσο χρόνο έχουμε ακόμα διαθέσιμο, στο (T)OPTW το πόσο μπορούμε να σπρώξουμε κάθε επίσκεψη διαφέρει.

1.2.2.2 Χρονικά Εξαρτημένος Προσανατολισμός

Όπως και στην προηγούμενη παραλλαγή, έτσι και εδώ, μιλάμε για έναν χρονικό γράφο. Το μέγεθος που εξαρτάται από τον χρόνο αυτή την φορά είναι το βάρος των ακμών. Το πόσο μακριά είναι δύο κόμβοι δεν εξαρτάται πλέον μόνο από τους κόμβους, αλλά και από τον χρόνο. Αυτή η παραλλαγή επίσης προκύπτει από την πραγματική ζωή όπου ανάλογα με την κίνηση και τα δρομολόγια των συγκοινωνιών ο χρόνος που απαιτείται για την μετάβαση από το ένα σημείο στο άλλο διαφέρει μέσα στην μέρα. Ένα άλλο παράδειγμα είναι ο χρόνος που απαιτείται να περιμένει κανείς στην ουρά σε ένα λούνα παρκ, ένα μέγεθος το οποίο αλλάζει μέσα στην μέρα.

1.2.2.3 Τοξωτός Προσανατολισμός

Σε αυτή την παραλλαγή το κέρδος βρίσκεται στις άκμες και όχι στις κορυφές του γράφου. Παραδείγματα για αυτό το πρόβλημα προέρχονται από τα πεδία των τηλεπικοινωνίων και των μεταφορών. Έχει αποδειχθεί ότι μια περίπτωση του AOP μπορεί να αναχθεί σε μια περίπτωση του OP. Οι προτεινόμενες λύσεις περιλαμβάνουν ILS, διάφορους ευριστικούς αλγορίθμους από το Gavalas et al. (2015), Branch-and-Cut και υβριδική Tabu Search με φάση διαφοροποίησης από την ακριβή λύση του ILP.

1.2.2.4 Στοχαστικός Προσανατολισμός

Αν και μέχρι τώρα, όλα τα μεγέθη στο πρόβλημα ήταν ντετερμινιστικά, υπάρχουν μερικές παραλλαγές του OP όπου ένα ή περισσότερα μεγέθη σχετίζονται με μια πιθανότητα ή μια κατανομή. Στο πρόβλημα Orienteering με Στοχαστικά Κέρδη (OPSP), το κέρδος που αποκομίζεται από μια κορυφή είναι μια τυχαία μεταβλητή που ακολουθεί μια κανονική κατανομή. Από την άλλη πλευρά, στο πρόβλημα του προσανατολισμού με στοχαστικούς χρόνους ταξιδιού και χρόνο εξυπηρέτησης (OPSTS) ο χρόνος που απαιτείται για να ολοκληρωθεί μια επίσκεψη σε μια κορυφή ή να διασχιθεί μια άκμη λαμβάνεται από μια γνωστή κατανομή. Ένα πρακτικό παράδειγμα του OPSP στο TTDP είναι να προγραμματίστει το ταξίδι μόνο βάσει της βαθμολογίας για κάθε υποψήφιο μέρος χωρίς

να ξέρει από πριν πόσο θα απολαύσει πραγματικά την επίσκεψη. Από την άλλη πλευρά, το OPSTS είναι σαν να πηγαίνει κανείς στη στάση του λεωφορείου χωρίς να έχει δει το χρονοδιάγραμμα. Το πότε θα έρθει το λεωφορείο είναι μια τυχαία μεταβλητή. Μια άλλη προτεινόμενη εφαρμογή είναι ένα θεματικό πάρκο όπου το μήκος της ουράς σε κάθε παιχνίδι προέρχεται από μια διανομή. Συνήθως, ο στόχος των λύσεων σε τέτοια προβλήματα είναι να εξασφαλιστεί ένα ορισμένο όριο με καλή πιθανότητα. Για παράδειγμα ότι θα φτάσετε στο ξενοδοχείο πριν από τις 20:00 με πιθανότητα μεγαλύτερη από 90% ή ότι θα συγκεντρώσετε παραπάνω από ένα ορισμένο κέρδος.

1.2.2.5 Γενικός Προσανατολισμός

Η διαφορά αυτής της παραλλαγής με τις προηγούμενες βρίσκεται στον τρόπο που υπολογίζουμε το συνολικό κέρδος μίας λύσης. Μέχρι στιγμής το συνολικό κέρδος ήταν το άθροισμα όλων των επιμέρους κερδών των κορυφών που επισκεφτήκαμε. Σε αυτή την παραλλαγή το η συνάρτηση κέρδους μπορεί να είναι υπέρ- ή ύπο-γραμμική. Αυτό το πρόβλημα προσπαθεί να περιλάβει την σχέση μεταξύ διαφορετικών κόμβων. Οι σχέσεις αυτές μπορεί να είναι συνεργατικές, όπως το να επισκεφτεί κανείς την ακρόπολη και το μουσείο της ακρόπολης ή ανταγωνιστικές, όπως η επίσκεψη σε δύο διαφορετικά εστιατόρια σε μικρό χρονικό διάστημα.

1.2.3 Ορισμός και Μαθηματικός Φορμαλισμός

Σε αυτή την ενότητα, θα δώσουμε έναν επίσημο ορισμό του ΟΠ και του MMCTOPTW. Έστω ένα πλήρες μη κατευθυνόμενο γράφημα με βάρη και N κορυφές, η κάθε μία με κέρδος P_i και χρόνο επισκέψεως V_i. Έστω ότι το βάρος της ακμής e_{ij} που δηλώνεται t_{ij} αντιπροσωπεύει το χρόνο που απαιτείται για την μετακίνηση από την κορυφή i στην κορυφή j. Οι τιμές t_{ij} ικανοποιούν την τριγωνική ανισότητα, δηλαδή $\forall i, j, k : t_{ij} \leq t_{ik} + t_{kj}$. Έστω μια αρχική κορυφή S και μια τελική κορυφή T. Ο στόχος του OP είναι να καθορίσει μια απλή διαδρομή από την κορυφή S στην κορυφή T διάρκειας μικρότερης ή ίσης με κάποια τιμή $T_{\max}$ έτσι ώστε να μεγιστοποιεί το $\sum P_i$ των κορυφών στο μονοπάτι.

Μαθηματικά, μπορεί να διατυπωθεί ως πρόβλημα ακέραιου προγραμματισμού (IP). Η μαθηματική διατύπωση του προβλήματος IP μπορεί επίσης να βρεθεί στο Vansteenwegen et al. (2011). Έστω $x_{ij} = 1$ όταν μια επίσκεψη στην κορυφή i ακολουθείται από μια επίσκεψη στην κορυφή j - 0 διαφορετικά. Έστω u_i είναι η θέση της κορυφής i στο απλό μονοπάτι.

$$Max \sum_{i=1}^{N-1} \sum_{j=2}^{N} P_i x_{ij} \tag{1.1}$$

$$\sum_{j \neq S} x_{Sj} = \sum_{i \neq T} x_{iT} = 1 \tag{1.2}$$

$$\sum_{i=1}^{N} x_{ik} = \sum_{j=1}^{N} x_{kj} \leq 1; \quad \forall k \neq S, T \tag{1.3}$$

$$\frac{V_S + V_T}{2} + \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij}(t_{ij} + \frac{V_i + V_j}{2}) \leq T_{\max} \tag{1.4}$$

$$1 = u_S < u_i < u_T = \max(u_i); \quad \forall i \neq S, T \tag{1.5}$$

$$u_i - u_j + 1 \leq (u_T - 1)(1 - x_{ij}) \quad \forall i, j \tag{1.6}$$

$$x_{ij} \in \{0, 1\} \tag{1.7}$$

Η αντικειμενική συνάρτηση 1.1 μεγιστοποιεί τη συνολική βαθμολογία που έχει συγκεντρωθεί. Οι περιορισμοί 1.2 βεβαιώνουν ότι η διαδρομή ξεκινά από την κορυφή S και τελειώνει στην κορυφή T. Οι περιορισμοί 1.3 εξασφαλίζουν τη συνέχεια της διαδρομής και εγγυώνται ότι κάθε κορυφή επισκέπτεται το πολύ μια φορά. Οι περιορισμοί 1.4 επιβάλλουν τον περιορισμό του χρόνου στον προϋπολογισμό. Οι περιορισμοί 1.5 και 1.6 είναι απαραίτητες για την αποφυγή των επιμέρους διαδρομών. Λόγω των προηγούμενων περιορισμών, κάθε κορυφή έχει μέγιστο εισερχόμενο και εξερχόμενο βαθμό 1. Από αυτό μπορούμε να συμπεράνουμε ότι εκτός από τις απομονωμένες κορυφές, το γράφημα θα περιέχει απλά μονοπάτια και κύκλους. Οι επιμέρους διαδρομές είναι οι κύκλοι σε αυτό το γράφημα. Ψάχνουμε για μια ενιαία πορεία και συνεπώς συμπεριλαμβάνουμε αυτούς τους περιορισμούς για την εξάλειψη των κύκλων. Αυτοί οι περιορισμοί συμπεριλαμβάνονται σύμφωνα με τη διατύπωση Miller-Tucket-Zemlin (MTZ) του TSP (Miller et al. (1960)).

Μια επέκταση στο αρχικό OP, που θα παρουσιάσει ενδιαφέρον για αυτή τη διατριβή, είναι το OP με Windows Time (OPTW). Σε αυτήν την επέκταση, η πείσκεψη σε κάθε κορυφή μπορεί να πραγματοποιηθεί μόνο κατά τη διάρκεια του χρονικού πλαισίου $[O_i, C_i]$. Μια άλλη επέκταση του OP είναι το Team OP (TOP), στο οποίο απαιτούνται D μονοπάτιο αντί για μόνο ένα, και ενώ όλα ξεκινούν και τελειώνουν στις ίδιες θέσεις, δεν μοιράζονται άλλους κόμβους. Επομένως, κάθε δραστηριότητα μπορεί να συμπεριληφθεί μόνο μία φορά σε όλα τα μονοπάτια. Τέλος, η επέκταση που παρουσιάζεται από αυτή τη διατριβή είναι το Minimum-Maximum Constraint TOPTW (MMCTOPTW), όπου εκτός από τα προηγούμενα χαρακτηριστικά, κάθε κορυφή έχει ένα διάνυσμα κατηγοριών $\overline{K_i}$. Επιπλέον, τα δύο διανύσματα m και M υποδηλώνουν τους ελάχιστους και μέγιστους περιορισμούς του προβλήματος αντίστοιχα. Ως αποτέλεσμα όλων των παραπάνω, η μαθηματική διατύπωση του OP πρέπει να επανεξεταστεί. Στη νέα διατύπωση, θα προστεθεί ένας ακόμη δείκτης στις περισσότερες από τις μεταβλητές μας για να υποδείξουμε την ημέρα που αναφέρουμε. Επιπλέον, θα εισαχθούν πρόσθετοι περιορισμοί για να εκφραστούν οι περιορισμοί χρόνου των χρονικών παραθύρων και οι ελάχιστοι και μέγιστοι περιορισμοί. Για να εκφράσουμε τους περιορισμούς των χρονικών παραθήρων, η μεταβλητή s_{id} δηλώνει την έναρξη της επίσκεψης στη δραστηριότητα i στο μονοπάτι d.

$$Max \sum_{d=1}^{D} \sum_{i=1}^{N-1} \sum_{j=2}^{N} P_i x_{ijd} \tag{1.8}$$

$$\sum_{d=1}^{D} \sum_{j \neq S} x_{Sjd} = \sum_{d=1}^{D} \sum_{i \neq T} x_{iTd} = D \tag{1.9}$$

$$\sum_{i=1}^{N} x_{ikd} = \sum_{j=1}^{N} x_{kjd} \leq 1; \quad \forall k \neq S, T; \forall d \tag{1.10}$$

$$\frac{V_S + V_T}{2} + \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ijd}(t_{ij} + \frac{V_i + V_j}{2}) \leq T_{\max}; \quad \forall d \tag{1.11}$$

$$s_{id} + T_i + t_{ij} - s_{jd} \leq M(1 - x_{ijd}); \quad \forall i, j, d \tag{1.12}$$

$$1 = u_{Sd} < u_{id} < u_{Td} = \max(u_i); \quad \forall i \neq S, T \tag{1.13}$$

$$u_{id} - u_{jd} + 1 \leq (u_{Td} - 1)(1 - x_{ijd}); \quad \forall i, j \tag{1.14}$$

$$O_i \leq s_{id} \leq C_i; \quad \forall i, d \tag{1.15}$$

$$|K_i|_1 = 1; \quad \forall i \tag{1.16}$$

$$\overline{m} \leq \sum_{d=1}^{D} \sum_{i=1}^{N-1} \sum_{j=2}^{N} \overline{K_i} x_{ijd} \leq \overline{M} \tag{1.17}$$

$$x_{ijd} \in \{0, 1\} \tag{1.18}$$

Η αντικειμενική συνάρτηση 1.8 τώρα είναι ένα άθροισμα σε όλες τις διαδρομές. Ο περιορισμός 1.9 εξασφαλίζει ότι κάθε μέρα ξεκινάει από *S* και τελειώνει στο *T*. Οι περιορισμοί 1.10 εξασφαλίζουν τη συνέχεια της πορείας σε κάθε ημέρα και διασφαλίζουν ότι κάθε κορυφή επισκέπτεται το πολύ μία φορά κάθε μέρα, όπως παραπάνω. Οι περιορισμοί 1.11 βεβαιώνουν ότι δεν υπάρχει υπέρβαση του προϋπολογισμού χρόνου κάθε διαδρομής. Οι περιορισμοί 1.12, 1.13 και 1.14 είναι απαραίτητες για την αποφυγή των επιμέρους διαδρομών σύμφωνα με την τυποποίηση MTZ του TSP. Οι περιορισμοί 1.15 διασφαλίζουν ότι οι χρόνοι έναρξης είναι εντός του επιτρεπόμενου χρονικού πλαισίου. Τέλος, οι περιορισμοί 1.16 διασφαλίζουν ότι κάθε κορυφή ανήκει ακριβώς σε μία κατηγορία. Οι περιορισμοί 1.17 εκφράζουν τις ελάχιστες και μέγιστες επιτρεπόμενες κορυφές κάθε κατηγορίας.

1.3 Σχεδιασμός και υλοποίηση

1.3.1 Σχεδιασμός και Υλοποίηση

Σε αυτό το κεφάλαιο θα δώσουμε πρώτα μια περιγραφή υψηλού επιπέδου του αλγορίθμου στο Vansteenwegen et al. (2009b), στη συνέχεια θα εμβαθύνουμε στις τεχνικές λεπτομέρειες και τέλος θα παρουσιάσουμε τον δικό μας αλγόριθμο.

Πιστεύουμε ότι ο όρος "POI" που χρησιμοποιείται στην αναφερόμενη δημοσίευση δεν περιγράφει ολόκληρη την έννοια αυτού που θέλουμε να εκφράσουμε με μια επίσκεψη, καθώς εκφράζει κυρίως μια χωρική οντότητα ενώ θέλουμε να δώσουμε έμφαση και στον χρονικό της χαρακτήρα. Για αυτόν τον λόγο, από εδώ και στο εξής, θα τις αναφερθούμε ως Activities, οι οποίες λαμβάνουν χώρα σε μια τοποθεσία, αλλά έχουν επίσης χρονικό παράθυρο. Αυτή η ιδέα περιγράφει καλύτερα γεγονότα όπως οι συναυλίες, δίνοντας έμφαση στο γεγονός ότι ενώ πολλές δραστηριότητες μπορούν να πραγματοποιηθούν στην ίδια θέση, η κάθε μία είναι διαφορετική.

Η προτεινόμενη λύση είναι μια επαναλαμβανόμενη τοπική αναζήτηση (ILS). Ο αρχικός αλγόριθμος περιγράφεται παρακάτω. Σε κάθε επανάληψη, οι Δραστηριότητες εισάγονται άπληστα στην τρέχουσα λύση χρησιμοποιώντας μία ευριστική συνάρτηση για να αξιολογήθουν οι πιθανοί υποψήφιοι, έως ότου δεν μπορούν να εισαχθούν περισσότεροι. Αφού εισαχθούν όλες οι δυνατές Δραστηριότητες, αξιολογείται η νέα λύση. Αν είναι καλύτερη από την καλύτερη λύση που έχουμε βρει μέχρι στιγμής, κάνουμε αυτήν την καλύτερη λύση. Στο τέλος του βρόγχου, αφαιρούνται τμήματα της τρέχουσας λύσης για να μπορέσουμε να δημιουργήσουμε μία νέα λύση. Επαναλαμβάνουμε αυτή τη διαδικασία έως ότου η λύση δεν βελτιωθεί για 150 επαναλήψεις. Καθώς οι Δραστηριότητες εισάγονται και στη συνέχεια αφαιρούνται, οι παραγόμενες λύσεις είναι ουσιαστικά δείγματα από μια κατανομή από την οποία διατηρούμε το ένα με το υψηλότερο κέρδος.

Ουσιαστικά, μπορούμε να διαιρέσουμε τον αλγόριθμο σε τέσσερα ξεχωριστά βήματα. Αρχικοποίηση, εισαγωγή, αξιολόγηση και αφαίρεση. Αρχικοποιούμε τα μονοπάτια, στη συνέχεια εισάγουμε όσο το δυνατόν περισσότερα μέσα σε έναν βρόγχο, αξιολογούμε τη νέα λύση, αφαιρούμε τμήματα του, για να αποκτήσουμε ένα νέο στο επόμενο βήμα.

Ενώ κατά την πρώτη ανάγνωση αυτό μπορεί να φαίνεται σαν ένας εύκολος αλγόριθμος, υπάρχουν κάποιες παγίδες στην εφαρμογή που πρέπει να ληφθούν υπόψη. Αφού βρούμε την επόμενη Δραστηριότητα για να εισάγουμε και πού να την εισάγουμε, η διατήρηση των δομών δεδομένων μας ώστε να είναι συνεπείς για την επόμενη επανάληψη δεν είναι ασήμαντη. Λόγω του χρόνικου παραθύρου, όπως αναφέρθηκε προηγουμένως, κάθε επόμενος χρόνος εκκίνησης πρέπει να υπολογιστεί εκ νέου. Το άλλο ερώτημα που γίνεται πιο δύσκολο είναι εάν μια συγκεκριμένη δραστηριότητα μπορεί να εισαχθεί στη λύση. Στο TOP, διατηρώντας μια μεταβλητή ανά μονοπάτι για το πόσο περισσότερο χρόνο μπορούμε να χρησιμοποιήσουμε για νέα POI είναι αρκετό, αλλά στο TOPTW απαιτείται μια μεταβλητή για κάθε Δραστηριότητα και πρέπει να ενημερώνεται μετά από κάθε εισαγωγή. Αυτή η μεταβλητή ονομάζεται μέγιστη μετατόπιση. Εκπροσωπεί πόσο αργότερα μπορεί να ξεκινήσει αυτή η Δραστηριότητα.

Μια σημαντική βελτίωση όσον αφορά το χρόνο εκτέλεσης είναι η επιλεκτική εξέτασει των Δραστηριοτήτων σε κάθε επανάληψη. Αν δεν μπορούσαμε να φτάσουμε σε μια Δραστηριότητα στην προηγούμενη επανάληψη, τότε μετά την προσθήκη μιας ακόμα Δραστηριότητας στη λύση μας, δεν θα είμαστε επίσης σε θέση να την φτάσουμε. Λόγω των χρονικών παραθύρων, θα μπορούσαμε να φτάσουμε για μια δραστηριότητα νωρίτερα από την ώρα έναρξης. Σε αυτή την περίπτωση, θα πρέπει να περιμένουμε. Αυτή είναι μια άλλη μεταβλητή που πρέπει να ενημερώνεται σε κάθε τροποποίηση της λύσης.

Όταν εισάγετε μια νέα δραστηριότητα το Vansteenwegen et al. (2009b) λαμβάνει υπόψη δύο πράγματα: Χρόνος και Κέρδος. Ενώ το κέρδος είναι απλό στη ρύθμιση των προσθέτων, ο χρόνος πρέπει να λαμβάνει υπόψη τόσο την επίσκεψη όσο και τον χρόνο ταξιδιού. Για το σκοπό αυτό, οι συγγραφείς έχουν ορίσει την μεταβλητή $Shift$, η οποία είναι πόσο αργότερα θα φτάσουμε στην επόμενη Δραστηριότητα εάν επισκεπτόμασταν τη νέα Δραστηριότητα. Συνδύασαν αυτά τα δύο με μια πολύ κομψή μέτρηση $Ratio = \frac{()^2}{Shift}$. Για κάθε υποψήφια Δραστηριότητα χρησιμοποιείται

το ελάχιστο $Shift$. Η εύρεση της ελάχιστης μετατόπισης απαιτεί την προσπάθεια εισαγωγής της Δραστηριότητας μεταξύ κάθε δύο διαδοχικών Δραστηριοτήτων που υπάρχουν στη λύση. Για μεγάλες περιπτώσεις, η πολυπλοκότητα της εύρεσης αυτής της ελάχιστης μετατόπισης δεν είναι αμελητέα.

Στη συνέχεια, θα επικεντρωθούμε στο βήμα της αφαίρεσης. Αφού αξιολογηθεί κάθε λύση, πρέπει να αφαιρέσουμε ένα μέρος από αυτήν, έτσι ώστε να δημιουργηθεί μια νέα, διαφορετική λύση. Στον αλγόριθμό τους, προσπαθούν να κουνήσουν λιγότερο από το 1/3 της τρέχουσας λύσης. Η εκ νέου εκκίνηση δεν προσφέρει τίποτα, καθώς η λύση είναι καθοριστική, οπότε θα καταλήξουμε να δημιουργήσουμε λύσεις που έχουμε ήδη δει. Για να επιτευχθεί αυτό, μετά από κάθε κούνημα, αυξάνουν τον αριθμό των Δραστηριοτήτων που θα αφαιρεθούν κατά 1. Επίσης, το σημείο έναρξης για της αφαίρεσης αυξάνεται με τον αριθμό των Δραστηριοτήτων που έχουν αφαιρεθεί.

Οι τύποι για την ενημέρωση των διαφόρων ποσοτήτων συνοψίζονται στο 3.1. Ο αρχικός ψευδοκώδικας παρατίθεται παρακάτω στο 3.1, 3.2 και 3.3.

1.3.1.1 Αλγόριθμος

Ο αρχικός αλγόριθμος πρέπει να επανεξεταστεί για να ληφθούν υπόψη οι ελάχιστοι και μέγιστοι περιορισμοί. Οι μέγιστοι πριορισμοί είναι πολύ εύκολο να τηρηθούν. Η μόνη αναγκαία τροποποίηση είναι στο βήμα εισαγωγής, όπου οι Δραστηριότητες που εξετάζονται, δεν θα τους παραβίαζαν εάν είχαν εισαχθεί. Από την άλλη πλευρά, η τήρηση των ελάχιστων απαιτήσεων δεν είναι τόσο απλή. Προφανώς, μικρά ελάχιστα σε συχνές κατηγορίες σε μεγάλα μονοπάτια θα ικανοποιούνται πάντα, ακόμη και με τον αρχικό αλγόριθμο. Ωστόσο, όταν οι περιορισμοί είναι πιο απαιτητικοί, οι λύσεις που παράγονται από τον αρχικό αλγόριθμο πιθανότατα δεν θα είναι έγκυρες. Επομένως, η ευρηστική συνάρτηση πρέπει να βελτιωθεί για να δώσει πιο εφικτές λύσεις πιο συχνά.

$$Ratio_i = \frac{(Profit_i)^2}{Shift_i}(1 + W\frac{Demand[Category_i]}{Supply[Category_i]}) \tag{1.19}$$

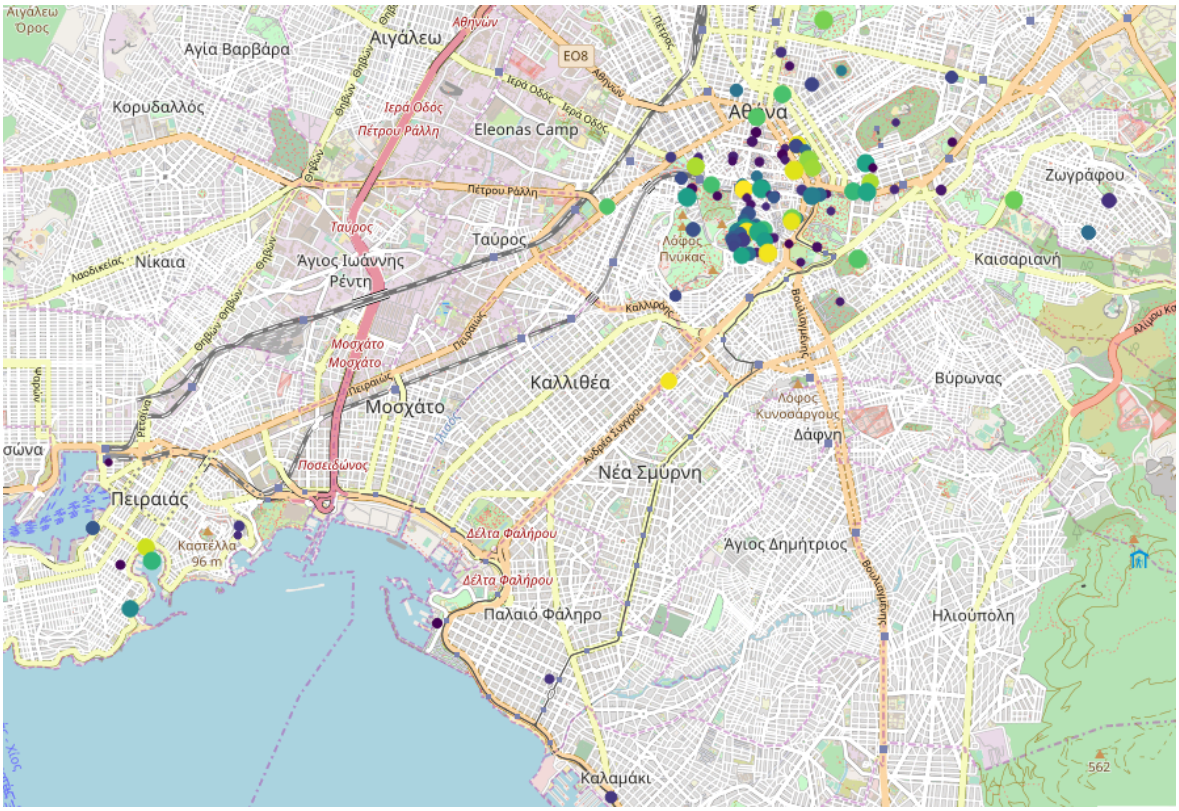
Αυτός ο κανόνας θα αναφέρεται από εδώ και στο εξής ως "τετραγωνικό κέρδος, πολλαπλασιαστική ζήτηση προς προσφορά". Παρόλο που εξακολουθούν να είναι δυνατές λύσεις που δεν είναι εφικτές, ο τρόπος με τον οποίο παράγονται οι λύσεις, δίνει προτεραιότητα (σε κάποιο βαθμό) σε Δραστηριότητες που ανήκουν σε κατηγορίες που έχουν ζητηθεί. Όπως θα φανεί, αυτή η απλή τροποποίηση είναι αρκετή για την επίτευξη λύσεων υψηλής ποιότητας για το MMCTOPTW. Μια άλλη τροποποίηση που βοήθησε στην παραγωγή καλύτερων λύσεων ποιότητας είναι η εισαγωγή μιας μικρής τυχαιότητας. Αυτό επιτυγχάνεται πολλαπλασιάζοντας το $Ratio_i$ με έναν τυχαίο αριθμό που προέρχεται από μια ομοιόμορφη κατανομή. Φυσικά, αυτές οι τροποποιήσεις δημιουργούν το ερώτημα των τιμών που πρέπει να έχουν αυτές οι παράμετροι. Παρόλο που οι αρχικοί συγγραφείς δεν χρησιμοποίησαν κινήσεις 2-OPT, θα προσπαθήσουμε μερικές παραλλαγές που θα μπορούσαν να μας βοηθήσουν στην επίτευξη καλύτερων ποιοτικών αποτελεσμάτων. Ο ψευδοκώδικας για το 2-OPT μπορεί να βρεθεί στο 3.4.

1.3.1.2 Περιγραφή των δεδομένων

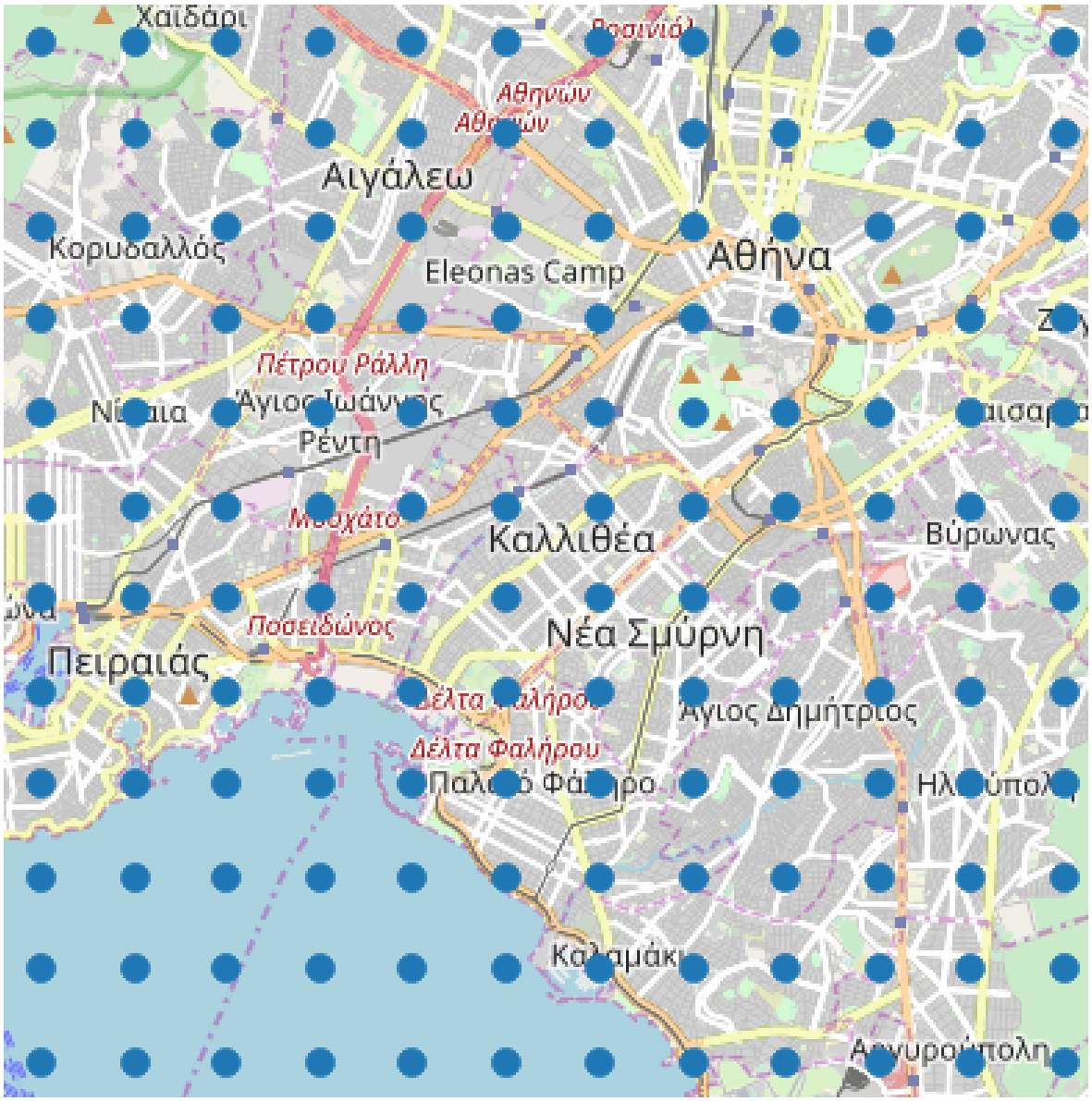
Για όλους τους υπολογισμούς, το σύνολο δεδομένων που περιλαμβάνει 113 τοποθεσίες (μουσεία και γκαλερί τέχνης, αρχαιολογικούς χώρους, μνημεία και ορόσημα, δρόμους και πλατείες, γειτονιές, εκκλησίες και θρησκευτική κληρονομιά, πάρκα) οι οποίες βρίσκονται στο κέντρο της Αθήνας και του Πειραιά. Από τις παραπάνω περιγραφόμενες τοποθεσίες δημιουργήθηκαν 20 διαφορετικές "τοπολογίες", διατηρώντας τις πραγματικές συντεταγμένες και επομένως τους χρόνους μετακίνησης, ωστόσο τυχαιοποιήθηκαν τα αντίστοιχα κέρδη, οι χρόνοι επίσκεψης και οι ώρες ανοίγματος και κλεισίματος. Επίσης, έχει προστεθεί τυχαία μια κατηγορία σε κάθε δραστηριότητα, όπως φαίνεται στην 3.2.

Το σύνολο δεδομένων ολοκληρώνεται από 144 ισοαπέχουσες θέσεις, οι οποίες μπορούν να χρησιμοποιηθούν ως σημεία εκκίνησης.

Παρακάτω, στο σχήμα 1.1, φαίνονται όλες οι δραστηριότητες και οι περιοχές στον χάρτη της Αττικής. Το μέγεθος του δείκτη είναι ανάλογο με το κέρδος της σχετικής Δραστηριότητας. Το χρώμα του δείκτη είναι πιο μπλε για δραστηριότητες με μικρούς χρόνους επίσκεψης και πιο έντονες για δραστηριότητες με μεγάλες χρόνους επίσκεψης. Ο χάρτης των περιοχών ακολουθείται στην εικόνα 1.2.



Σχήμα 1.1: Χάρτης των Δραστηριοτήτων



Σχήμα 1.2: Χάρτης των περιοχών

Προφανώς, μόνο οι έγκυρες περιοχές θα χρησιμοποιηθούν ως σημεία εκκίνησης και τερματισμού και όχι εκείνες που δεν είναι προσπελάσιμες (π.χ. υποβρύχιες).

1.3.2 Κώδικας

1.3.2.1 Μέθοδοι και Δομές Δεδομένων

Η τελική λύση αποτελείται από μονοπάτια. Κάθε μονοπάτι αποτελείται από μία σειριακή λίστα κόμβων. Τα μέλη του κόμβου παρατίθενται στο 3.3. Οι ιδιότητες του κόμβου παρατίθενται στην καρτέλα 3.4.

Πίνακας 1.1: Μέλη του Κόμβου

Όνομα	Τύπος	Περιγραφή
activity	Activity	Δείκτης στην αντίστοιχη Δραστηριότητα
arrive	int	Χρόνος που φτάνουμε στην Δραστηριότητα
start	int	Χρόνος που ξεκινάμε την Δραστηριότητα
max_shift	int	Χρόνος που αυτή η Δραστηριότητα μπορεί να μετατεθεί αργότερα

Πίνακας 1.2: Ιδιότητες του Κόμβου

Όνομα	Τύπος	Περιγραφή	Εξίσωση
wait	int	Χρόνος μεταξύ άφιξης και εκκίνησης	start-arrive
max_start	int	Το αργότερο που μπορεί να ξεκινήσει η Δραστηριότητα	arrive+max_shift
leave	int	Χρόνος αναχώρησης από την Δραστηριότητα	start+visit

Το σύστημα επίλυσης έχει πρωτοτυποποιηθεί σε Python 2.7 και έχει βελτιστοποιηθεί ώστε να λειτουργεί σε Rust. Ο επιλυτής χρειάζεται στην αρχικοποίησή του τα εξής:
Ο αριθμός των διαδρομών
Οι χρόνοι έναρξης και λήξης για κάθε ημέρα
Οι διαθέσιμες Δραστηριότητες
Η περιοχή έναρξης και λήξης
Τα ελάχιστα απαιτούμενα από κάθε κατηγορία
Τα μέγιστα που επιτρέπονται από κάθε κατηγορία

1.4 Εύρεση βέλτιστων τιμών παραμέτρων

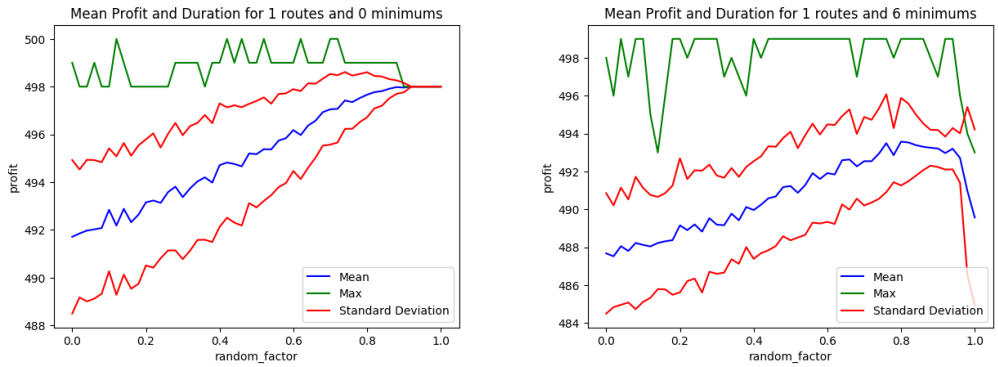
Ένας αριθμός από παραμέτρους έχουν εισαχθεί και πρέπει να ληφθεί υπόψη η επιλογή των τιμών για κάθε μία από αυτές. Ενώ οι περισσότερες από τις παραμέτρους έχουν μία τιμή που είναι προτιμότερη από όλες τις άλλες και επομένως μπορούν να χρησιμοποιηθούν με ασφάλεια σε όλες τις εφαρμογές του αλγορίθμου, η βέλτιστη τιμή για μερικές από αυτές αλλάζει ανάλογα με τη περίπτωση. Σε αυτό το τμήμα, θα αναλύσουμε τις διάφορες εναλλακτικές που εξετάσαμε για κάθε μία και θα παρουσιάσουμε πειραματικά αποτελέσματα για τις επιλογές μας.

1.4.1 Συνάρτηση $Ratio$

Η πρώτη επιλογή που πρέπει να γίνει είναι αυτή της συνάρτησης $Ratio$. Η βασική απόφαση είναι να χρησιμοποιήσουμε ως αριθμητή το $(Profit)$ (το οποίο είναι η τυπική ευριστική για το αντίστοιχο πρόβλημα του knapsack) ή $(Profit)^2$ (που προτείνεται από το Vansteenwegen et al. (2009b)). Για να απαντήσουμε σε αυτήν την ερώτηση, εκτελέσαμε την περίπτωση χωρίς περιορισμούς για κάθε τοπολογία για 100 φορές και συγκρίναμε τα αποτελέσματα σε σχέση με το μέσο κέρδος και τον μέσο χρόνο ολοκλήρωσης. Τα αποτελέσματα φαίνονται στα σχήματα B.1-B.6.
Στο πρώτο σύνολο δεδομένων είναι σαφές ότι το $profit^2$ παίρνει πολύ λιγότερο χρόνο, αλλά δίνει υποδυέστερα αποτελέσματα από την άποψη του κέρδους για τις περισσότερες τοπολογίες. Ο λόγος για τους μειωμένους χρόνους είναι ότι αυτό το κλάσμα ευνοεί τις δραστηριότητες με μεγαλύτερα κέρδη και, συνεπώς, μεγαλύτερους χρόνους επίσκεψης. Ως αποτέλεσμα, λιγότερες δραστηριότητες συμμετέχουν στην τελική λύση και ως εκ τούτου, κάθε νέα λύση υπολογίζεται με λιγότερες εισαγωγές, οι οποίες με τη σειρά τους χρειάζονται λιγότερο χρόνο. Δώστε προσοχή στο $t11$ για 1 διαδρομή, όπου οι δραστηριότητες είναι, κατά μέσο όρο, μόνο 60% εκείνες των λύσεων με $profit$ στον αριθμητή της συνάρτησης $Ratio$.
Για το σύνολο δεδομένων Montemanni, τα αποτελέσματα είναι αρκετά όμοια από την άποψη τόσο του χρόνου όσο και του κέρδους. Μπορούμε να δούμε ότι υπάρχουν μερικές αιχμές στο χρόνο, αλλά μπορούν να αποδοθούν σε στατιστικό σφάλμα παρά σε πραγματικές διαφορές. Στο τελευταίο σύνολο δεδομένων, μπορούν να γίνουν οι ίδιες παρατηρήσεις όπως στο προηγούμενο. Καθώς αυτή η εργασία επικεντρώνεται κυρίως στην παραγωγή ποιοτικών λύσεων με πολλούς ελάχιστους περιορισμούς, απαιτείται να συμπεριληφθούν περισσότερες δραστηριότητες στην τελική λύση. Ως εκ τούτου, το υπόλοιπο της εργασίας θα επικεντρωθεί στα αποτελέσματα χρησιμοποιώντας τον αριθμητή $profit$.

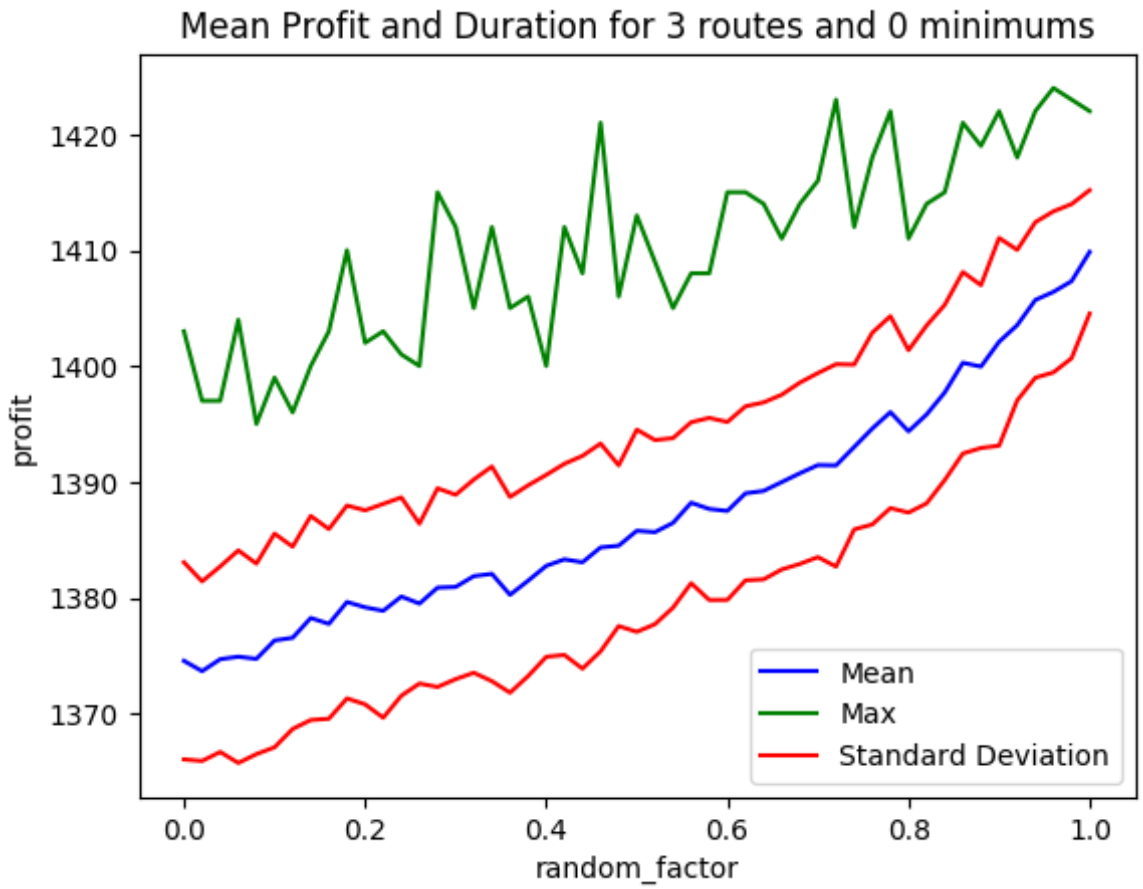
1.4.2 Τυχαιότητα

Η δεύτερη παράμετρος που χρήζει μιας τιμής είναι η τυχαιότητα. Συγκεκριμένα, το εύρος τιμών που θα έχουν οι τυχαίες τιμές. Όπως αναφέρθηκε παραπάνω, αφού υπολογίσουμε το $Ratio$ για κάθε Δραστηριότητα, θα το πολλαπλασιάσουμε με μια τυχαία τιμή που προέρχεται από μια ομοιόμορφη κατανομή. Έστω ότι το εύρος ίναι $[low, high], low, high \in R^+$. Παρατηρήστε ότι επειδή θα χρησιμοποιήσουμε αυτήν την τιμή για να πολλαπλασιάσουμε την αναλογία, αυτό είναι ισοδύναμο με οποιοδήποτε άλλο εύρος $[a * low, a * high], a \in R^+$. Επομένως, χωρίς απώλεια της γενικότητας, μπορούμε να περιορίσουμε το εύρος μας σε $[low, 1], low \in [0, 1]$. Ως αποτέλεσμα, χρειαζόμαστε μόνο μία τιμή, low αντί για δύο. Όταν $low = 1$, ο τυχαίος αλγόριθμος εκφυλίζεται στον αντίστοιχο ντετερμινιστικό. Ελλείψη μιας επίσημης ανάλυσης, θα καταφύγουμε σε εκτεταμένες δοκιμές σε τρέχουσες τοπολογίες για να εξάγουμε αποτελέσματα. Για το σκοπό αυτό, η πρώτη τοπολογία του Γαβαλά έτρεξε για 1 έως 3 δρομολόγια που ξεκινούν από το κέντρο της Αθήνας (όπου οι Δραστηριότητες είναι πιο πυκνές) και με ελάχιστα [0, 2, 4, 6] της κατηγορίας 1. Η ώρα έναρξης ήταν 600 και η ώρα λήξης ήταν 1020 για όλες τις διαδρομές. Παρακάτω, οι γραφικές εμφανίζουν μέση, μέγιστη και τυπική απόκλιση του κέρδους, καθώς και το μέσο όρο του χρόνου που χρειάστηκε για τον υπολογισμό της λύσης. Κάθε περίπτωση εκτελέστηκε 100 φορές για να εξαλειφθεί η στοχα-



Σχήμα 1.3: Κέρδος έναντι τυχαιότητας για 1 μονοπάτι

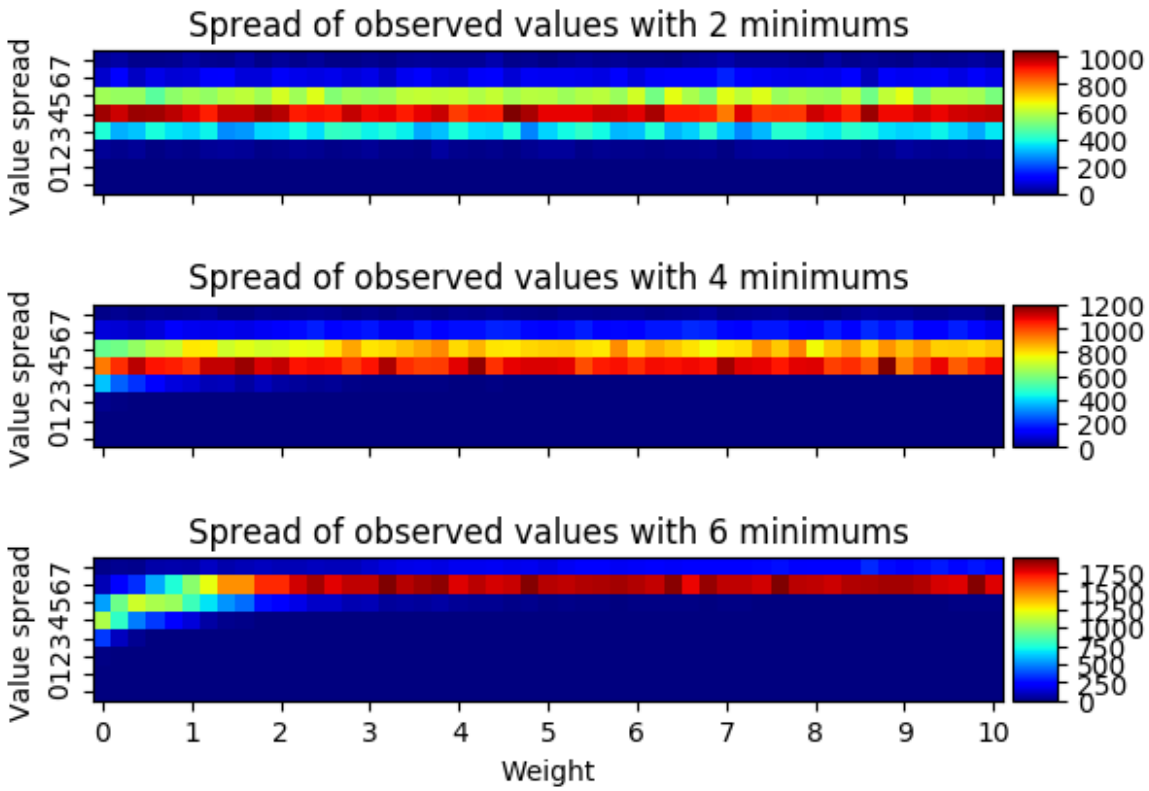
στική φύση των παραγόμενων λύσεων. Περιλαμβάνουμε μόνο τα πιο ενδιαφέροντα στοιχεία εδώ στο 1.3 και 1.4. Τα πλήρη αποτελέσματα μπορούν να βρεθούν στα σχήματα B.7-B.13.
Ο λόγος για τον οποίο αποφασίσαμε να συμπεριλάβουμε αυτές τις τρεις γραφικές είναι ότι εμφανίζουν τρεις διαφορετικές συμπεριφορές. Στο πρώτο σχήμα, βλέπουμε το μέσο κέρδος να αυξάνεται σταθερά μέχρι το $low = 0.9$ από το οποίο και μετά παραμένει σταθερό. Παράλληλα με την αύξηση αυτή, παρατηρούμε μια σταθερή μείωση της τυπικής απόκλισης, μέχρις ότου όλες οι λύσεις είναι ίδιες (std = 0). Αυτή η συμπεριφορά υποστηρίζει μια μεγαλύτερη τιμή low, ή ακόμα και έναν ντετερμινιστικό αλγόριθμο. Η δεύτερη γραφική παρουσιάζει μια πολύ απότομη πτώση του μέσου κέρδους μετά το $low = 0.95$. Σε αυτή την περίπτωση μπορούμε να δούμε τα πλεονεκτήματα της εισαγωγής μιας μικρής τυχαιότητας. Είναι πολύ πιο δύσκολο να κολλήσει σε ένα τοπικό βέλτιστο όταν υπάρχει περισσότερη τυχαιότητα. Αυτή η εικόνα απεικονίζει απόλυτα γιατί διαφοροποιήσαμε την λύση μας από τη βάση της δουλειάς μας και δεν επιλέγουμε πάντα τη Δραστηριότητα με το μέγιστο $Ratio$. Αυτός ο μηχανισμός μας επιτρέπει να διερευνήσουμε πιο ποικίλες λύσεις και να μην έχουμε κολλήσει στην επαναδημιουργία της ίδιας λύσης ξανά και ξανά. Τέλος, η τελευταία γραφική μας δείχνει τι συμβαίνει όταν ο χώρος αναζήτησης δεν αξιοποιείται αρκετά. Σε αυτή την περίπτωση, η τυχαιότητα δεν προσφέρει καλύτερες λύσεις, αλλά μας αναγκάζει να εξερευνήσουμε σε λιγότερο ελπιδοφόρες περιοχές, θέτοντας σε κίνδυνο την ικανότητά μας να επιτύχουμε τη βέλτιστη λύση.
Τα συμπεράσματα από τις τρεις γραφικές είναι τα παρακάτω. Όσο μεγαλύτερος είναι ο τυχαίος παράγοντας, τόσο λιγότερες διαφορές υπάρχουν ανάμεσα στις διαφορετικές λύσεις. Το ακραίο για αυτό το συμπέρασμα μπορεί να φανεί στο πρώτο σχήμα, όπου για τυχαίο παράγοντα ίσο με 1, όλες οι παραγόμενες λύσει είναι οι ίδιες. Όσο μεγαλύτερος είναι ο χώρος αναζήτησης, τόσο μεγαλύτερη είναι η διακύμανση στις λύσεις. Επίσης, για μεγάλους χώρους αναζήτησης, το μέγιστο ακολουθεί το μέσο όρο. Η τυχαιότητα προέρχεται από τον τυχαίο παράγοντα και την τυχαία αφαίρεση κόμβων. Όσο μεγαλύτερος είναι ο τυχαίος παράγοντας, τόσο πιο πιθανό είναι να παραχθεί μια λύση υψηλής ποιότητας, αλλά είναι μερικές φορές λιγότερο πιθανό να παράχθει η καλύτερη δυνατή λύση.
Από αυτά τα στοιχεία, καθώς και από τα υπόλοιπα στο Παράρτημα Β, είναι ξεκάθαρο ότι για μικρές περιπτώσεις, θα πρέπει να εισαχθεί περισσότερη τυχαιότητα μετά από λίγες επαναλήψεις, ενώ για μεγαλύτερους χώρους αναζήτησης, αν δεν υπάρχει αρκετός χρόνος, θα πρέπει να εξερευνηθούν οι ντετερμινιστικές λύσεις αντι αυτού. Σε μελλοντικές εργασίες θα θέλαμε να δούμε την τυχαιότητα να μην παραμένει ίδια κατά τη διάρκεια ολόκληρης της αναζήτησης, αλλά να ξεκινάει από την παραγωγή ντετερμινιστικών λύσεων και καθώς επιστρέφουμε στις ίδιες λύσεις που δείχνουν ότι η φάση της εκμετάλλευσης ολοκληρώνεται ως επί το πλείστον, να ξεκινάμε να εισάγουμε περισσότερη τυχαιότητα ώστε να εξερευνούμε μεγαλύτερες περιοχές του χώρου αναζήτησης.



Σχήμα 1.4: Κέρδος έναντι τυχαιότηττας για 3 μονοπάτια

1.4.3 Χρησιμοποιώντας κινήσει 2-OPT

Στην δημοσίευσή τους Vansteenwegen et al. (2009b), οι συγγραφείς επέλεξαν να μην εκτελούν κινήσεις 2-OPT, λόγω της μικρής αλληλεπικάλυψης των Χρονικών Παραθύρων. Ωστόσο, στο πιο ρεαλιστικό σετ δεδομένων Gavalas et al. (2015) υπάρχει σημαντική αλληλεπικάλυψη μεταξύ των χρονικών παραθύρων των διάφορων Δραστηριοτήτων και ως εκ τούτου θεωρήθηκε ευεργετικό να συμπεριληφθούν και οι δύο επιλογές στο σύστημα μας. Επειδή μετά από κάθε επιτυχημένη κίνηση 2-OPT, πρέπει να γίνουν περισσότερες εισαγωγές, κάθε επανάληψη διαρκεί περισσότερο. Από την άλλη πλευρά, κατά την εκτέλεση του 2-OPT, ο αλγόριθμος να παράγει καλύτερα αποτελέσματα. Αρχικά, δοκιμάσαμε να πραγματοποιούμε 2-OPT, αφού είχαμε εισαγάγει όλες τις Δραστηριότητες που μπορούσαμε και στη συνέχεια προσπαθήσαμε να εισάγουμε ξανά όσο το δυνατόν περισσότερες Δραστηριότητες. Τα αποτελέσματα αυτής της προσέγγισης παρουσιάζονται στις γραφικες παραστάσεις B.14-B.19 ως κλάσμα του αντίστοιχου μέσου κέρδους που επιτυγχάνεται χωρίς κινήσεις 2-OPT.
Αν και η χρήση του 2-opt είναι συστηματικά καλύτερη, δεν βελτιώνει σημαντικά την απόδοση. Το σύνολο δεδομένων Montemanni ωφελείται περισσότερο από αυτή την τροποποίηση, αλλά παρόλα αυτά, η μέση βελτίωση είναι κάτω από 1%. Η εκτέλεση του βήματος εισαγωγής μετά από 2-opt, παίρνει πολύ χρόνο, καθώς όλες οι Δραστηριότητες πρέπει να επανεξεταστούν. Αυτό απαιτεί πολύ χρόνο και ως εκ τούτου η προσέγγιση αυτή είναι πολύ πιο αργή.
Πριν όμως απορρίψουμε αυτή την τεχνική, προσπαθήσαμε επίσης να αντικαταστήσουμε Δραστηριότητες αντί να εισάγουμε νέες. Τα σχετικά αποτελέσματα μπορούν να φανούν στις γραφικές B.20-B.25 Κατά μέσο όρο, η χρήση 2-OPT παράγει ελαφρώς καλύτερες λύσεις. Ωστόσο, αυτή η τεχνική χρησιμοποιείται καλύτερα σε ορισμένα σύνολα δεδομένων και λιγότερο σε άλλες. Στο

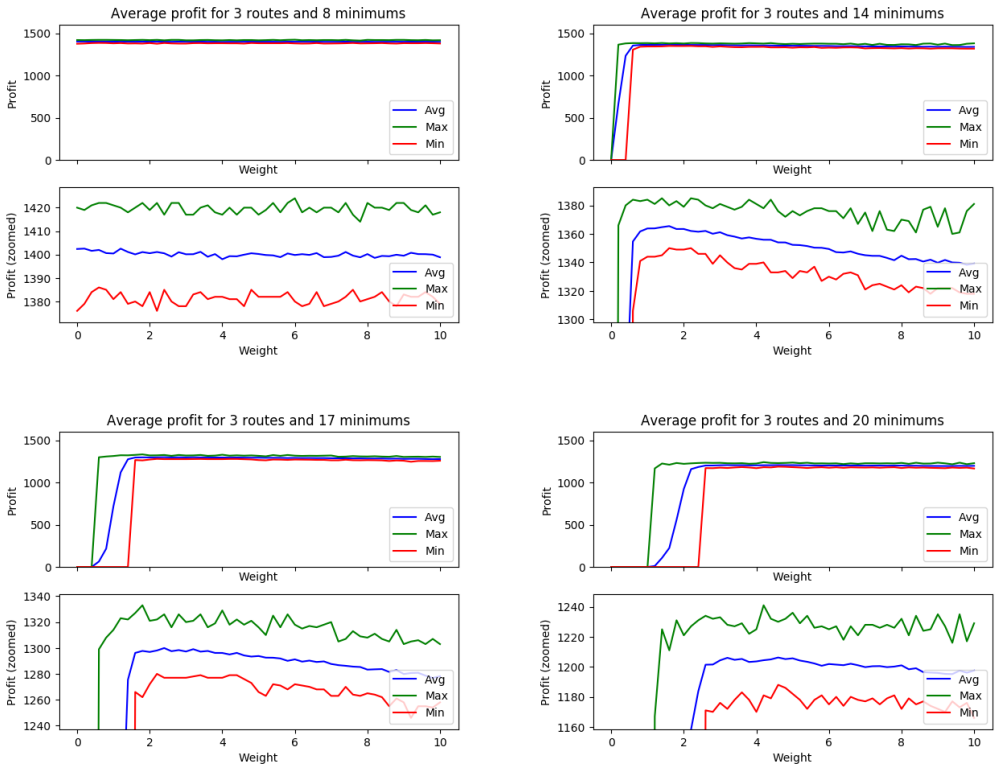


Σχήμα 1.5: Πλήθος Δραστηριοτήτων σε κάθε λύση με κατηγορία '1' σε σχέση με το βάρος και τα ελάχιστα για ένα μονοπάτι

πλαίσιο του ILS, η εκτέλεση 2-OPT είναι αρκετά δαπανηρή όσον αφορά τον χρόνο υπολογισμού και κατά συνέπεια δεν εφαρμόζεται εύκολα και αποτελεσματικά. Στο πλαίσιο της μελλοντικής εργασίας, θα μπορούσε να χρησιμοποιηθεί μια πιο εποικοδομητική και καθοδηγούμενη χρήση 2-OPT, η οποία θα αποφέρει συγκρίσιμα ή και καλύτερα αποτελέσματα, χωρίς να απαιτείται τόσο μεγάλη υπολογιστική ισχύς.

1.4.4 Η επιλογή του w

Τέλος, υπάρχει το πρόβλημα της τιμής του w στη συνάρτηση $Ratio$. Έχοντας μια πολύ χαμηλή τιμή θα οδηγήσει σε λίγες εφικτές λύσεις και ως εκ τούτου, τις περισσότερες φορές θα δαπανηθούν υπολογίζωντας λύσεις που δεν μπορούν να χρησιμοποιηθούν. Από την άλλη πλευρά, μια πολύ μεγάλη αξία θα έχει αρνητικές επιπτώσεις στον παράγοντα κέρδους και συνεπώς θα δημιουργεί λύσεις χαμηλότερης ποιότητας. Στην πραγματικότητα, ανάλογα με το χρόνο που υπάρχει, τη θέση της τρέχουσας διαδρομής, τα ελάχιστα που έχουμε και την τοπολογία, υπάρχει μια διαφορετική βέλτιστη τιμή w. Ωστόσο, δεδομένου ότι η τιμή αυτή είναι εξαιρετικά ασταθής, υπάρχει μικρή ελπίδα να καθοριστούν αυτές οι βέλτιστες τιμές για όλες τις περιπτώσεις *a priori*. Σε αυτή την ενότητα θα δείξουμε πρώτα ότι το πλήθος των Δραστηριοτήτων κάθε κατηγορίας σε μια λύση είναι δείγματα από μια διακριτή κανονική κατανομή. Τότε θα δείξουμε πώς αυτή η κατανομή διαφοροποιείται για διαφορετικά βάρη. Τέλος, θα παρουσιάσουμε μερικούς λαγορίθμους μάθησης που βασίζονται στην Στοχαστική Κατάβασης Κλίσης (SGD).
Ξεκινώντας, χρησιμοποιήσαμε μια ενιαία τιμή για όλες τις τοπολογίες και τις κατηγορίες. Στο σχήμα 1.5 έχουμε καταγράψει την καταμέτρηση των δραστηριοτήτων που ανήκουν στην κατηγορία '1' για διαφορετικά ελάχιστα και διαφορετικά σταθερά βάρη. Για κάθε βάρος παράγουμε περίπου 2000 διαφορετικές λύσεις.



Σχήμα 1.6: Κέρδος σε σχέση με ένα βάρος για 8,14,17 και 20 ελάχιστα της κατηγορίας '1' για 3 μονοπάτια

Εξετάζοντας την γραφική, είναι σαφές ότι όταν εφαρμόζονται λίγοι ή καθόλου περιορισμοί, ο αριθμός των Δραστηριοτήτων που υπάρχουν στην λύση με την κατηγορία '1' ακολουθεί κανονική κατανομή. Το ίδιο ισχύει για χαμηλές τιμές βάρους για πιο περιορισμένες ρυθμίσεις. Σε πιο περιορισμένες ρυθμίσεις, η κανονική κατανομή εξακολουθεί να υπάρχει κάτω από το καθορισμένο όριο, αλλά συγκεντρώνεται περισσότερο στο κατώτατο όριο καθώς το βάρος αυξάνεται. Αυτό μπορεί να φανεί στην εξασθενημένη μπλε γραμμή όταν απαιτούμε τουλάχιστον έξι Δραστηριότητες της κατηγορίας '1' και το βάρος είναι μεταξύ 2 και 4. Αυτή η υπόθεση ενισχύεται περαιτέρω όταν εξετάζουμε μεγαλύτερες περιπτώσεις με πολλούς περιορισμούς.
Καθώς αλλάζουν τα ελάχιστα, ο αριθμός και οι διάρκειες των μονοπατιών, το ίδιο συμβαίνει και με τη βέλτιστη τιμή για το σταθερό βάρος. Για να δούμε πώς αλλάζει αυτό, τρέξαμε τον αλγόριθμο για διαφορετικά βάρη και διαφορετικά ελάχιστα για να δούμε πώς επηρεάζεται το κέρδος. Προκειμένου να εξουδετερωθούν τα αποτελέσματα της τυχαιότητας, εκτελέσαμε κάθε πείραμα 100 φορές. Από όλα αυτά τα αποτελέσματα, επιλέξαμε να συμπεριλάβουμε εδώ τα πιο εντυπωσιακά από αυτά που θα μας βοηθήσουν να πάρουμε μια διαίσθηση καθώς θα διατυπώσουμε το επόμενο βήμα μας.
Όπως μπορούμε να δούμε από την πρώτη εικόνα, όταν ο ελάχιστος αριθμός Δραστηριοτήτων της κατηγορίας '1' είναι μικρότερος από ό,τι κανονικά θα υπήρχε σε μια τυχαία λύση, το βάρος δεν έχει καμιά απολύτως επίδραση. Από την άλλη πλευρά, στην περίπτωση που θα υπήρχαν συνήθως λιγότερες Δραστηριότητες με αυτήν την κατηγορία (σε αυτή τη ρύθμιση, είναι περίπου 12), απαιτείται μη μηδενική τιμή για την επίτευξη καλύτερων αποτελεσμάτων. Το σχήμα της κατανομής των κερδών που παράγονται σε σχέση με το βάρος μπορεί να φανεί από $minimums = 14$ και αργότερα. Αρχικά, υπάρχει ένα εύρος όπου ο αλγόριθμος αποτυγχάνει να βρει οποιεσδήποτε εφικτές λύσεις και επομένως το μέγιστο κέρδος είναι μηδέν. Στη συνέχεια, καθώς αυξάνεται το βάρος, το ίδιο συμβαίνει και με την πιθανότητα εύρεσης μιας εφικτής λύσης. Αυτό μπορεί να φανεί στο άλμα γύρω από το βάρος $w = 0.5$ για $minimums = 14$. Η πιθανότητα να βρεθεί μια έγκυρη λύση αυξάνεται μέχρις ότου όλες οι διαδρομές παράγουν μια εφικτή λύση, οπότε το ελάχιστο κέρδος

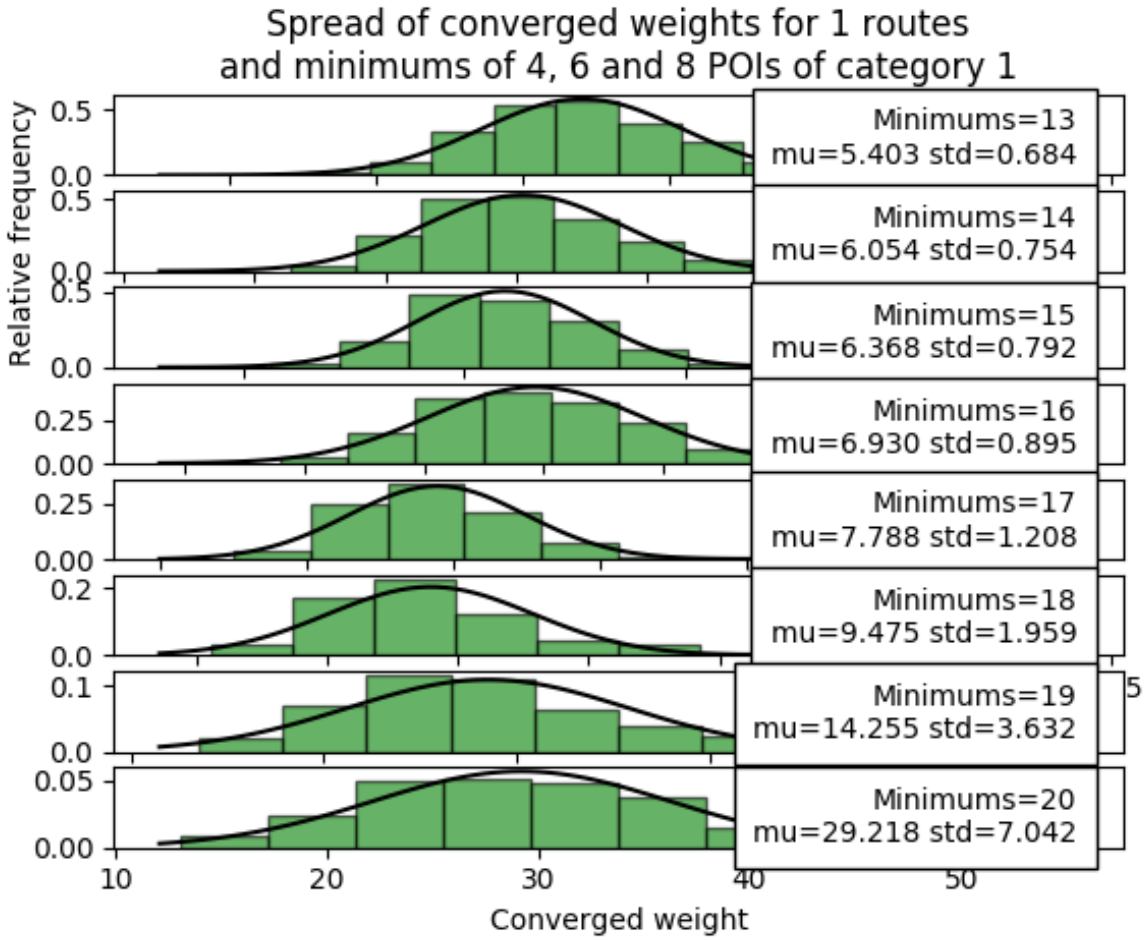
πηδά περίπου στην ίδια τιμή με το μέγιστο. Κοντά σε αυτό, παρατηρείται επίσης το μέγιστο μέσο κέρδος. Μετά από αυτό, το μέσο κέρδος μειώνεται σιγά-σιγά, καθώς η ευριστική είναι τώρα περισσότερο επικεντρωμένη στην εισαγωγή των Δραστηριοτήτων αυτής της κατηγορίας, παρά για την επίτευξη υψηλού κέρδους. Στην ουσία, ο υπολογιζόμενος λόγος παραμορφώνεται από την επιλογή του καλύτερου κέρδους και περισσότερο από την επιλογή Δραστηριοτήτων με τις απαιτούμενες κατηγορίες.
Η επιλογή ενός στατικού βάρους λειτουργεί για ορισμένες τοπολογίες, αλλά κατά πάσα πιθανότητα κάθε επιλογή θα απέχει πολύ από τη βέλτιστη τιμή. Επομένως, στην επόμενη ενότητα, θα προσπαθήσουμε να βρούμε αυτή τη βέλτιστη αξία ταυτόχρονα με την παραγωγή νέων υποψηφίων λύσεων. Για να το επιτύχουμε αυτό, στρεφόμαστς προς τη μηχανική μάθηση.
Προκειμένου να χρησιμοποιηθεί μηχανική μάθηση, το πρώτο πράγμα που πρέπει να γίνει είναι να επινοήθει μια αντικειμενική λειτουργία για να βελτιστοποιηθεί. Η τιμή τους βάρους που ψάχνουμε είναι ασφαλώς πάνω από το όριο όπου δεν υπάρχουν έγκυρες λύσεις. Στη συνέχεια, η τιμή που ψάχνουμε πρέπει να είναι πάνω από το όριο όπου η εκτέλεση ενδέχεται να μην επιστρέψει μια λύση. Είναι βέβαιο ότι μια κακή λύση είναι καλύτερη από καμία λύση. Τέλος, θα θέλαμε να καταλήξουμε σε μια τιμή που δεν υπερβαίνει κατά πολύ το προηγούμενο όριο, καθώς οι μεγάλες αξίες θα οδηγήσουν σε χαμηλότερα κέρδη. Από την παραπάνω περιγραφή βλέπουμε ότι στην πραγματικότητα ψάχνουμε για $\min weight : P(|weight) = 1$. Στην αρχή αυτής της υποενότητας, δείξαμε ότι για ένα δεδομένο βάρος, η πιθανότητα πόσων δραστηριοτήτων θα περιέχονται σε μια τυχαία λύση, ακολουθεί μια κανονική κατανομή. Ως αποτέλεσμα, αυτό που πραγματικά αναζητούμε είναι το μικρότερο βάρος που θα εξασφαλίσει ότι θα υπάρχουν πολλές έγκυρες λύσεις για να διαλέξουμε. Μια αρκετά καλή προσέγγιση είναι η χρήση μιας γραμμικής συνάρτησης. Συνεπώς, η λειτουργία που χρησιμοποιούμε είναι

$$P(w) = \begin{cases} 0 & w < \frac{b}{a} \\ aw - b & \frac{b}{a} \leq w \leq \frac{b+1}{a} \\ 1 & \frac{b+1}{a} < w \end{cases}$$

Έχοντας καταλήξει σε μια συνάρτηση βαθμολόγησης, τώρα πρέπει να επιλέξουμε έναν αλγόριθμο μάθησης. Δεδομένου ότι έχουμε έναν τρόπο να βρούμε γρήγορα τυχαίες λύσεις με δεδομένο βάρος, χρησιμοποιώντας το ILS, η αξιοποίηση της γνωστής τεχνικής Stochastic Gradient Descent είναι μια καλή επιλογή.
Αρχικά, δοκιμάζουμε έναν απλό αλγόριθμο SGD, ο οποίος έδωσε μια νέα τυχαία λύση με απαιτήσεις D που παράχθηκαν με βάρος W ενημερώνεται ως εξής:

$$W_{new} = W + D * step \tag{1.20}$$

Η μεταβλητή $step$ είναι της μορφής $step = \exp(-\frac{\ln(0.05) * iteration}{estimated\ total\ iterations})$. Με αυτή τη στρατηγική ενημέρωσης, είναι σαφές ότι, κατ' αρχήν, το W θα πρέπει να συγκλίνει σε μια τιμή όπου όλες οι λύσεις είναι έγκυρες. Δυστυχώς, επειδή το W λαμβάνεται υπόψη μόνο όταν δεν έχει τηρηθεί το ελάχιστο για αυτή την κατηγορία, ο αριθμός των Δραστηριοτήτων με αυτήν την κατηγορία σπάνια θα είναι πάνω από το καθορισμένο όριο. Ως αποτέλεσμα, το W συγκλίνει σε πολύ υψηλότερη τιμή από την επιθυμητή. Τα ιστογράμματα του τελικού W για τις περιπτώσεις με 3 μονοπάτια με απαίτηση 13-20 ελάχιστα εμφανίζονται παρακάτω για να παρέχουν μια εύκολη παρατήρηση για το πόσο αυτή η λειτουργία υπερβαίνει το επιθυμητό.
Μπορούμε να δούμε ότι η συγκλίνουσα τιμή είναι τόσο μεγάλη που στο τέλος κυριαρχεί πλήρως σε οποιαδήποτε άλλη συνιστώσα της συνάρτησης $Ratio$. Ως αποτέλεσμα, η επίτευξη ενός καλού κέρδους δεν είναι πλέον η προτεραιότητα αυτής της συνάρτησης. Αντίθετα, εστιάζει απλώς στη συμπερίληψη απαιτούμενων δραστηριοτήτων χωρίς να λαμβάνει υπόψη το κέρδος. Αντ' αυτού μπορούσαμε να εισάγουμε Δραστηριότητες μόνο από τις απαιτούμενες κατηγορίες. Η προηγούμενη λειτουργία τείνει να συγκλίνει σε μεγάλες τιμές, πράγμα που είναι επιζήμιο για το τελικό

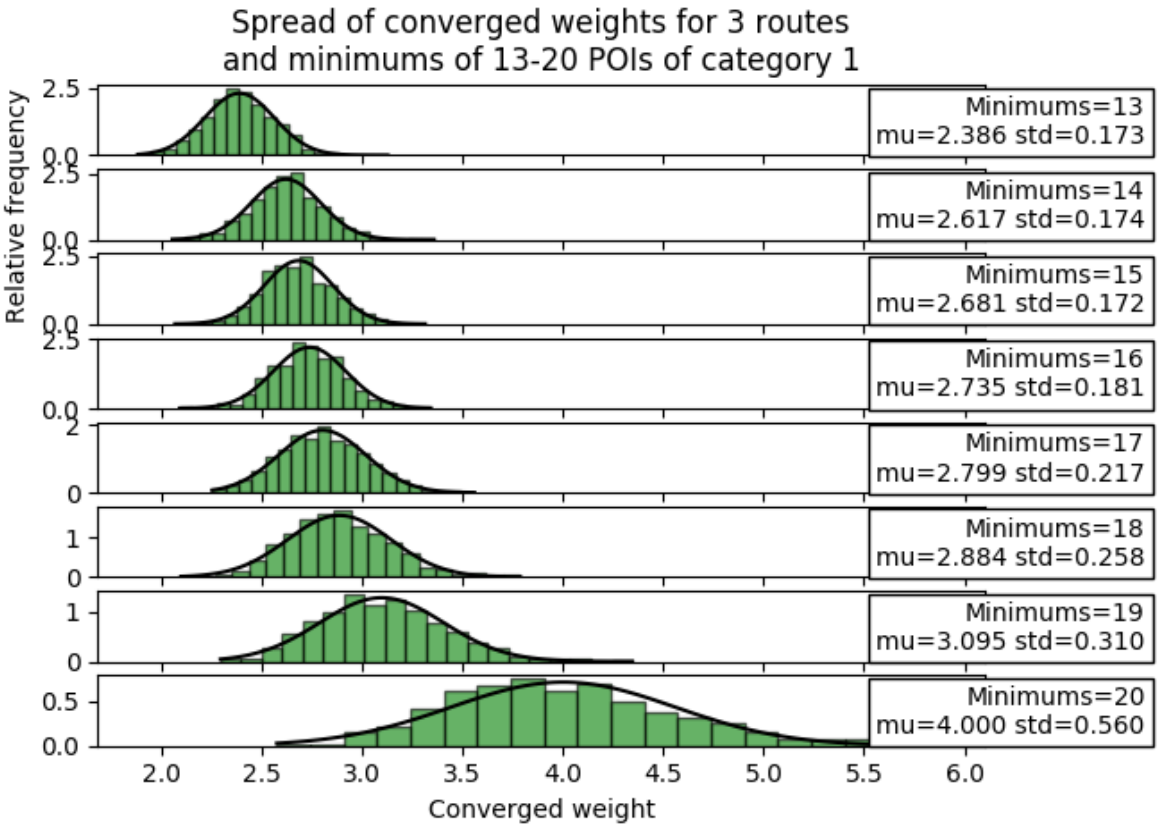


Σχήμα 1.7: Ιστόγραμμα των τελικών βαρών για διαφορετικά ελάχιστα

αποτέλεσμα. Για να αποφευχθεί αυτό, θα χρησιμοποιήσουμε κανονικοποίηση τροποποιώντας την προηγούμενη συνάρτηση ως εξής.

$$W_{new} = W + (D - \lambda W) * step \tag{1.21}$$

Αυτή η τροποποίηση περιορίζει την τιμή του W και βοηθάει στο να πλησιάσει την επιθυμητή τιμή. Για τις 3 διαδρομές και τη ζήτηση της κατηγορίας 1 Δραστηριότητες στην περιοχή 13-20 έχουμε τις ακόλουθες κατανομές.
Ένα εύκολο συμπέρασμα από την γραφική 1.8 είναι ότι η προσθήκη κανονικοποίησης συμβάλλει σημαντικά στον περιορισμό των ακραίων τιμών του W, ειδικά σε απαιτητικές περιπτώσεις, όπου είναι αδύνατη η εξασφάλιση ότι όλες οι λύσεις που παράγονται είναι έγυρες είναι αδύνατο. Στην ουσία, η κανονικοποίηση είναι ένας τρόπος για να μειώσουμε την αποδεκτή πιθανότητα έγκυρων λύσεων. Καθώς το W αυξάνεται, το ίδιο συμβαίνει και με την έκπτωση που οδηγεί σε χαμηλότερο σταθερό σημείο. Ένα άλλο σημείο που πρέπει να σημειωθεί είναι πώς η κανονικοποίηση επηρεάζει την απόκλιση των τελικών βαρών. Όπως μπορεί εύκολα να φανεί, χωρίς κανονικοποίηση υπάρχει μεγάλη απόκλιση μεταξύ των παραγόμενων τιμών, που κυμαίνονται έντονα γύρω από τον μέσο όρο. Από την άλλη πλευρά, χρησιμοποιώντας κανονικοποίηση, η τυπική απόκλιση είναι σημαντικά χαμηλότερη και, όπως αναμένεται, είναι μεγαλύτερη για πιο περιοριστικές περιπτώσεις.
Τέλος, ας σχεδιάσουμε τα συμπεράσματα από το σχήμα 1.6 και ας τα συνδυάσουμε με την παραπάνω ανάλυση για να δούμε πόσο καλή είναι η προσέγγισή μας.
Από τα παραπάνω στοιχεία μπορούμε να διαπιστώσουμε ότι με μεγάλη εμπιστοσύνη ο αλγόριθμος θα καταλήξει σε μια πολύ καλή τιμή για το W και ως εκ τούτου υπάρχει μεγάλη πιθανότητα να πα-



Σχήμα 1.8: Ιστόγραμμα των τελικών βαρών για διαφορετικά ελάχιστα με κανονικοποίηση

ράγονται πολλές έγκυρες και άκρως ανταγωνιστικές λύσεις βελτιώνοντας τη συνολική ποιότητα της τελικής επιστρεφόμενης λύσης.
Τέλος, προσπαθήσαμε να προσθέσουμε ορμή στη συνάρτηση ενημέρωσης. Ο τύπος που χρησιμοποιήθηκε είναι:

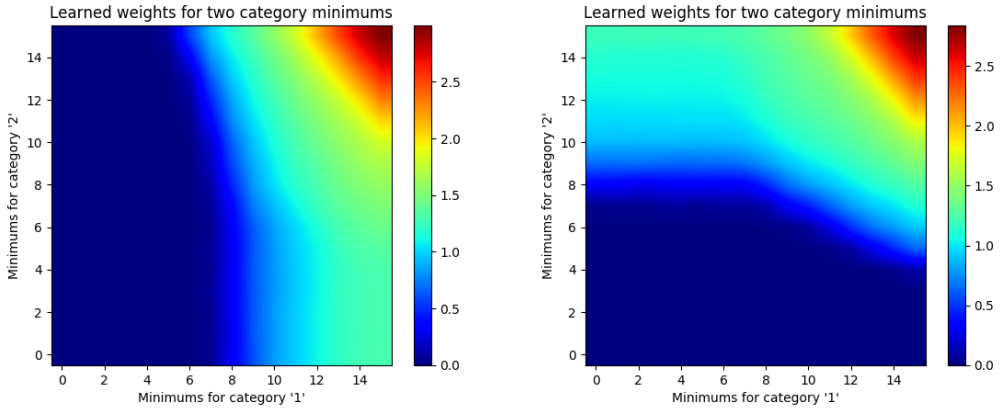
$$V_i = V_{i-1} * 0.7 + (D - 0.15 * W) \tag{1.22}$$

$$W_i = W_{i-1} + step_i * V_i \tag{1.23}$$

Δυστυχώς, το αποτέλεσμα αυτής της τροποποίησης ήταν η ίδια κανονική κατανομή, αλλά με μεγαλύτερη διακύμανση. Τα αποτελέσματα και των 3 λειτουργιών ενημέρωσης εμφανίζονται στο 1.3. Η πρώτη συνάρτηση χωρίς κανονικοποίηση υπερβαίνει κατά πολύ την αναζητούμενη αξία. Με την προσθήκη της κανονικοποίησης, πλησιάζουμε πολύ την αξία που επιθυμούμε, διασφαλίζοντας ότι θα βρεθεί μια καλή λύση. Τέλος, προσθέτωντας ορμή, καταλήγουμε σε καλές λύσεις, αλλά οι λύσεις είναι πιο απλωμένες. Για το λόγο αυτό, επιλέγουμε τη δεύτερη λειτουργία ενημέρωσης.
Τέλος, θα δείξουμε ότι τα τελικά βάρη εξαρτώνται όχι μόνο από τα ελάχιστα για αυτήν την κατηγορία, αλλά και από τα ελάχιστα για άλλες κατηγορίες. Αυτή η εξάρτηση μεταξύ κατηγοριών μπορεί να παρατηρηθεί σε έναν 2D χάρτη στο 1.9
Η λογαριθμική συνάρτηση έχει εφαρμοστεί στα βάρη, ώστε να δούμε μια ομαλότερη μετάβαση και όχι μόνο ένα τμήμα που είναι κόκκινο και το υπόλοιπο μπλε. Μπορούμε να δούμε ότι το βάρος που αποκτήσαμε για κάθε κατηγορία είναι συνάρτηση των ελάχιστων για όλες τις κατηγορίες και όχι μόνο για εκείνες που αντιπροσωπεύουν. Ως αποτέλεσμα, βλέπουμε ότι η βέλτιστη λύση εξαρτάται όχι μόνο από τα ελάχιστα της κάθε μεμονωμένης κατηγορίας, αλλά ως συνάρτηση του συνόλου του διανύσματος m. Ευτυχώς, η προσέγγισή μας είναι αρκετά ισχυρή ώστε να μπορέσει να αντιμετωπίσει αυτήν την πολυπλοκότητα.

Ελάχιστα	Χωρίς κανονικοποίηση		Κανονικοποίηση		Με ορμή	
	avg	std	avg	std	avg	std
13	5.403	0.684	2.386	0.173	2.408	0.264
14	6.054	0.754	2.617	0.174	2.627	0.271
15	6.368	0.792	2.681	0.172	2.700	0.245
16	6.930	0.895	2.735	0.181	2.732	0.283
17	7.788	1.208	2.799	0.217	2.808	0.292
18	9.475	1.959	2.884	0.258	2.924	0.363
19	14.255	3.632	3.095	0.310	3.122	0.453
20	29.218	7.042	4.000	0.560	4.040	0.748

Πίνακας 1.3: Σύγκριση συναρτήσεων ενημέρωσης για διαφορετικά ελάχιστα



Σχήμα 1.9: Αλληλεξάρτηση τελικών βαρών σε σχέση με ελάχιστα για δύο κατηγορίες

1.5 Συγκρίσεις

1.5.1 Σύγκριση με προηγούμενα καλύτερα

Σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε τις παραμέτρους που έχουν επιλεχθεί στο προηγούμενο κεφάλαιο για την παραγωγή αποτελεσμάτων προκειμένου να συγκριθούν με παρόμοιες λύσεις. Τα αποτελέσματά μας εκτελέστηκαν σε επεξεργαστή Intel® Xeon® E5 - 2630 v3 @ 2.40GHz με χωρητικότητα 62 GB μνήμης RAM, ένα μεγάλο βήμα από το υλικό που χρησιμοποιήθηκε για τα αποτελέσματα των προηγούμενων εργασιών. Για το λόγο αυτό δεν θα συγκρίνουμε τους χρόνους λύσης στην ανάλυση μας.
Αρχικά, θα συγκρίνουμε τον αλγόριθμό μας με τον αλγόριθμο ILS από το Vansteenwegen et al. (2009b). Ο αλγόριθμος μας βρήκε συντριπτικά καλύτερα αποτελέσματα, ξεπερνώντας τον προκάτοχό του σε 217 από τις 304 περιπτώσεις, παράγοντας το ίδιο αποτέλεσμα σε 61 και δίνοντας χειρότερα απότελέσματα σε λιγότερες από 26 περιπτώσεις. Συγκεκριμένα, στο σύνολο δεδομένων Solomon (c*_100, r*_100, rc*_100), μόνο 1-2 περιπτώσεις ήταν χειρότερες, ενώ στις εκτεταμένες περιπτώσεις από Montemanni και Gambardella (c*_200, r*_200, rc*_200) που βασίζονται στον Solomon, παρατηρήσαμε την πλειονότητα των χειρότερων περιπτώσεων με 7 δείγματα που δεν επιδεικνύουν βελτίωση για 1 διαδρομή. Το πιο σημαντικό, για 3 μονοπάτια υπήρχαν 14 περιπτώσεις που παρήγαγαν τα ίδια αποτελέσματα και για 4 μονοπάτια 27 περιπτώσεις παρήγαγαν τα ίδια, βέλτιστα αποτελέσματα. Στο σύνολο δεδομένων Cordeau (pr1–10) υπάρχουν κατά μέσο όρο 7.5 περιπτώσεις που είναι καλύτερα με τις υπόλοιπες να είναι χειρότερες. Ομοίως, για το εκτεταμένο σύνολο δεδομένων Cordeau (pr11–20) υπάρχουν κατά μέσο όρο 9.25/10 καλύτερες περιπτώσεις. Κατά μέσο όρο, οι λύσεις μας είναι καλύτερες κατά 1.43%. Το ελάχιστο χάσμα που παρατηρήθηκε ήταν πάνω από 10% σε μία από τις περιπτώσεις Solomon. Αντίθετα, το μέγιστο χάσμα ήταν λίγο πάνω από 4% σε μία από τις εκτεταμένες περιπτώσεις Cordeau. Το μεγαλύτερο μέσο χάσμα

υπέρ του αλγορίθμου μας παρατηρήθηκε επίσης στο εκτεταμένο σύνολο δεδομένων Cordeau που ήταν 2.55%. Το μέγιστο μέσο όριο ήταν στο σύνολο δεδομένων Montemanni και Gambardella που βασίστηκε στο σύνολο δεδομένων Solomon. Ωστόσο, ο μέσος όρος ήταν ακόμα αρνητικός, καταλήγοντας στο συμπέρασμα ότι ο αλγόριθμός μας ήταν κατά μέσο όρο 0.67% καλύτερα. Πρέπει να σημειωθεί ότι αυτό είναι και το σύνολο δεδομένων όπου όλες οι περιπτώσεις ήταν οι ίδιες για 4 διαδρομές, μειώνοντας τον μέσο όρο, καθώς όλοι οι κόμβοι συμπεριλήφθηκαν και στις δύο λύσεις. Είναι φυσικό ότι για λιγότερες διαδρομές οι τιμές των διακενών είναι μεγαλύτερες, καθώς το απόλυτο κέρδος είναι μικρότερο και ως εκ τούτου μια ήπια αλλαγή, ας πούμε 10 πόντοι μπορεί να σημαίνει μια μεγάλη διαφορά πάνω από 5%. Για περισσότερες διαδρομές όπου το απόλυτο κέρδος είναι περίπου 600, μια λύση που έχει κέρδος 610 είναι μόλις 1.64% καλύτερη. Μπορούμε να δούμε αυτή την τάση μέσα από τους μέσους όρους για κάθε αριθμό διαδρομών. Όπως διαπιστώσαμε, από τη μια πλευρά, μικρές περιπτώσεις όπως το αρχικό σύνολο δεδομένων Solomon με το μικρό χώρο αναζήτησης είναι ευκολότερο να βρεθούν καλύτερες λύσεις, από την άλλη πλευρά, μεγαλύτερα σύνολα δεδομένων όπως τα Montemanni έχουν μελετηθεί λιγότερο και επίσης να παρουσιάσουν καλύτερα αποτελέσματα.
Ένας άλλος λόγος που δεν βρίσκουμε καλά αποτελέσματα για περισσότερα μονοπάτια είναι το μέγεθος του χώρου αναζήτησης. Ενώ για περισσότερα μονοπάτια δαπανώνται απαραίτητα περισσότερη υπολογιστική ισχύ, το μέγεθος του χώρου αναζήτησης και οι πιθανές λύσεις εκρήγνυνται εκθετικά, πράγμα που σημαίνει ότι η ποιότητα των προβλημάτων μας συνεχίζει να επιδεινώνεται περαιτέρω για κάθε νέο μονοπάτι που πρέπει να εξετάσουμε. Στο μέλλον, αυτό θα μπορούσε να διορθωθεί αποφασίζοντας δυναμικά πότε θα σταματήσει η αναζήτηση για μια καλύτερη λύση.
Με λίγα λόγια, ο αλγόριθμός μας συγκρίνεται πολύ ευνοϊκά με προηγούμενες προσπάθειες επίλυσης αυτού του προβλήματος. Παρουσιάζουμε τώρα τα αποτελέσματα σε πίνακες A.14-A.17. Κάθε πίνακας περιέχει αποτελέσματα για διαδρομές 1-4. Η πρώτη στήλη περιέχει το όνομα της τοπολογίας. Τα επόμενα τρία απεικονίζουν το αποτέλεσμα της ILS, το αποτέλεσμα που παράξαμε και την ποσοστιαία διαφορά μεταξύ τους. Ένα αρνητικό ποσοστό σημαίνει ότι ο αλγόριθμός μας είναι καλύτεροw. Το ποσοστιαίο κενό 0.00 σημαίνει ότι οι δύο λύσεις είναι ίσες. Όπως μπορούμε να δούμε, ο αλγόριθμος μας ταιριάζει με τα προηγούμενα αποτελέσματα και συγκρίνεται ευνοϊκά με προηγούμενες λύσεις.
Για να μην βαρεθεί ο αναγνώστης με πολλά δεδομένα, οι πίνακες αυτοί έχουν μετακινηθεί στο appendix A και εδώ συμπεριλαμβάνουμε μόνο μια περίληψη για κάθε σύνολο δεδομένων που μετράει σε πόσες περιπτώσεις δημιουργήσαμε ένα καλύτερο, χειρότερο ή το ίδιο αποτέλεσμα και πόσο είναι το ελάχιστο, το μέγιστο και το μέσο χάσμα.

Πίνακας 1.4: Περιληπτικά αποτελέσματα για το την περίπτωση χωρίς περιορισμούς

Name	routes	Worse	Same	Better	Max	Min	Average
righiniTOPTW2	1	1	8	20	0.68	-10.41	-2.10
	2	1	8	20	0.95	-6.36	-1.52
	3	2	0	27	0.14	-5.05	-1.62
	4	0	1	28	0.00	-4.08	-1.83
MontemanniTOPTW1	1	7	0	20	3.65	-3.57	-1.03
	2	4	1	22	2.34	-3.57	-0.98
	3	0	14	13	0.00	-2.86	-0.68
	4	0	27	0	0.00	0.00	0.00
righiniTOPTW3	1	1	2	7	1.91	-4.77	-1.71
	2	2	0	8	2.71	-5.10	-1.52
	3	4	0	6	1.29	-3.58	-1.15
	4	1	0	9	0.56	-2.84	-1.09
MontemanniTOPTW2	1	0	0	10	-0.60	-9.75	-5.15
	2	2	0	8	4.19	-7.66	-1.65
	3	0	0	10	-0.25	-5.76	-2.32
	4	1	0	9	2.70	-3.99	-1.08
Γενικά	1	9	10	57	3.65	-10.41	-2.50
	2	9	9	58	4.19	-7.66	-1.42
	3	6	14	56	1.29	-5.76	-1.44
	4	2	28	46	2.70	-4.08	-1.00
Όλα τα δεδομένα	0	26	61	217	4.19	-10.41	-1.43

1.5.2 Δυσκολότερες τοπολογίες

Σε αυτή την ενότητα θα παρουσιάσουμε πώς συμπεριφέρεται ο αλγόριθμος μας καθώς ζητάμε όλο και περισσότερα ελάχιστα. Μπορούμε να δούμε μια σταθερή μείωση του κέρδους, καθώς μεταβαίνουμε σε πιο απαιτητικά σενάρια. Σε σενάρια όπου δεν έχουν επιστραφεί λύσεις, αντικαταστήσαμε με τιμές NaN. Στα υπόλοιπα κελιά, παρουσιάζουμε τον μέσο όρο των έγκυρων λύσεων που επιστράφησαν. Αυτό σημαίνει ότι ενώ σε μερικές από τις πιο απαιτητικές περιπτώσεις, μόνο μερικές επιλύσεις παρήγαγαν ένα αποτέλεσμα, για να μην αποκλίνουν αυτά τα αποτελέσματα προς πολύ μικρές τιμές, το αποτέλεσμα για κάθε ρύθμιση θα είναι ο μέσος όρος μόνο των μη μηδενικών λύσεων. Σε αυτήν την ενότητα συμπεριλήφθηκε επίσης το σύνολο δεδομένων που χρησιμοποιήθηκε στο Gavalas et al. (2015). Σε γενικές γραμμές, μπορούμε να δούμε ότι καθώς εισάγουμε περισσότερα ελάχιστα, το επιστρεφόμενο αποτέλεσμα παραμένει το ίδιο ή μειώνεται όπως αναμενόταν.
Ωστόσο, σε 34 από τις συνολικά 96 περιπτώσεις μία από τις περιορισμένες περιπτώσεις που παράγονται προκύπτει καλύτερη κατά τουλάχιστον ένα πόντο κατά μέσο όρο από το μη περιορισμένο. Αν και αυτό μπορεί να φαίνεται παράξενο, μπορούμε να δούμε ότι δεν είναι. Με την εισαγωγή ελαχίστων, αλλάζουμε τις προτεραιότητες του αλγορίθμου. Σε ορισμένες περιπτώσεις, αυτή η ιεράρχηση οδηγεί σε καλύτερα αποτελέσματα, ενώ σε άλλες επιδεινώνεται. Ωστόσο, μόνο σε 2 τοπολογίες μία από τις περιορισμένες περιπτώσεις είναι καλύτερη κατά 5 ολόκληρους πόντους κατά μέσο όρο.
Στις περισσότερες περιπτώσεις, βλέπουμε ότι στο απεριόριστο περιβάλλον και το ελαφρώς περιορισμένο όπου απαιτούνται μόνο δύο Δραστηριότητες της πρώτης κατηγορίας, το μέσο κέρδος παραμένει το ίδιο. Από την άλλη πλευρά, καθώς απαιτούμε όλο και περισσότερες Δραστηριότητες αυτής της κατηγορίας, μπορούμε να δούμε ξαφνικές πτώσεις στα επιτευχθέντα αποτελέσματα. Ορισμένες από αυτές τις πτώσεις είναι τόσο ορατές που δείχνουν σαφώς ότι η παραγόμενη λύση δεν θα ληφθεί υπόψη εάν δεν συμπεριλάβαμε τα ελάχιστα. Από αποτελέσματα όπως αυτά, είναι σαφές ότι ο αλγόριθμός μας λειτουργεί και παράγει τα αναμενόμενα αποτελέσματα.

Γενικά, τα πειραματικά αποτελέσματα που λήφθησαν ήταν πολύ κοντά σε αυτά που περιμέναμε από τις αντίστοιχες θεωρητικές προβλέψεις. Επιπλέον, ο αλγόριθμός μας έχει συγκριθεί ευνοϊκά με τους τελευταίους αλγορίθμους στα γνωστά σύνολα δεδομένων. Εν ολίγοις, ο αλγόριθμός μας αποτελεί πολύτιμη προσθήκη στη βιβλιογραφία και οι ιδέες και η ανάλυση που χρησιμοποιήθηκαν για τη διαμόρφωση του μπορούν να αποτελέσουν τη βάση για μελλοντική εργασία στον τομέα.
Τέλος, κάθε αλγόριθμος χρειάζεται ένα όνομα. Έχουμε επιλέξει τον όρο Learning Randomized Weighted Iterated Local Search ή LRWILS για συντομία. Το τυχαίο μέρος αναφέρεται στον τυχαίο παράγοντα που χρησιμοποιήσαμε για να ξεφύγουμε από τα τοπικά ελάχιστα στις μικρές περιπτώσεις και το σταθμισμένο μέρος αναφέρεται στον νέο όρο που προσθέσαμε στον αλγόριθμο ILS, προκειμένου να δώσουμε προτεραιότητα στις σωστές Δραστηριότητες. Τέλος, ενώ λύνεται το στιγμιότυπο, ο LRWILS βρίσκει τα καλύτερα δυνατά βάρη για την περίπτωση αυτή.

1.6 Συμπέρασμα και μελλοντική έργα

1.6.1 Περίληψη

Στην παρούσα διπλωματική εργασία παρουσιάσαμε μια νέα παραλλαγή του προβλήματος Orienteering εμπνευσμένο από το Πρόβλημα Σχεδιασμού του Τουριστικού Ταξιδιού. Το κίνητρό μας για τη διαμόρφωση αυτής της νέας παραλλαγής καθιστά τον ορισμό του TTDP πιο πραγματικό και πιο πρακτικό με βλέψεις προς την εφαρμογή στην πραγματική ζωή που μπορεί να βοηθήσει πραγματικούς τουρίστες να επιλέξουν πώς να δομήσουν τις επισκέψεις τους σε μια νέα πόλη την οποία θα επισκεφθούν για λίγες μέρες. Ξεκινώντας από τον αλγόριθμο στο Vansteenwegen et al. (2009b), το επεκτήναμε με ένα νέο κομμάτι για να ωθήσουμε τον αλγόριθμο ώστε να εισαγάγει περισσότερες Δραστηριότητες με τις απαραίτητες κατηγορίες. Στον ίδιο αυτό αλγόριθμο εξετάσαμε παραλαγές όπως την τυχαιότητα, τις κινήσεις 2-OPT και τα αποτελέσματα της επιλογής να χρησιμοποιήθει το κλάσμα $profit^2/shift$ και πώς αυτές οι αυξήσεις επηρεάζουν το τελικό αποτέλεσμα του αλγορίθμου. Καταλήξαμε στο συμπέρασμα ότι η προσθήκη τυχαότητας βοηθά στην εξερεύνηση του χώρου αναζήτησης, ενώ ο ντετερμινιστικός αλγόριθμος βοηθά στην αξιοποίηση καλών λύσεων. Για μικρές περιπτώσεις όπου ξοδεύουμε αρκετή υπολογιστικής δύναμη, η εξερεύνηση μας βοηθάει να ξεφύγουμε από τοπικά ελάχιστα. Από την άλλη πλευρά, σε μεγάλες περιπτώσεις όπου δεν υπάρχει αρκετός χρόνος για να διερευνηθεί σωστά ο χώρος αναζήτησης, η εστίαση στην εκμετάλλευση αποδίδει καλύτερα αποτελέσματα. Λόγω των χρονικών ορίων που απαιτούνται, οι κινήσεις 2-OPT είναι πολύ δύσκολο να χρησιμοποιηθούν σωστά και αν και παράγουν καλύτερα αποτελέσματα, χρειάζεται πολύς χρόνος για να εκτελεστεί σωστά και ως εκ τούτου δεν είναι οικονομικά αποδοτικό. Όσον αφορά τη συνάρτηση αναλογίας, το $profit^2$ παράγει συγκρίσιμα αποτελέσματα, αλλά συνήθως περιλαμβάνει λιγότερες Δραστηριότητες στην τελική λύση. Προκειμένου να επιτύχουμε την καλύτερη δυνατή λύση, έπρεπε να βρούμε το κατάλληλο βάρος. Χρησιμοποιήσαμε μηχανική μάθηση και σχεδιάσαμε έναν αλγόριθμο Stochastic Gradient Descent για να μας βοηθήσει να βρούμε αυτή τη βέλτιστη τιμή. Είδαμε πως η κανονικοποίηση βοήθησε να δεσμευθεί αυτή η τιμή πιο κοντά στην επιθυμητή περιοχή και παρουσίασε ανταγωνιστικά αποτελέσματα με άλλους αλγόριθμους στην απεριόριστη ρύθμιση και αποτέλεσματα στην αντίστοιχη περιορισμένη.

1.6.2 Μελλοντική δουλειά

Υπάρχουν πολλές οδοί για να εξερευνήθεί μελλοντική δουλειά. Σε άμεση σχέση με αυτό το πρόβλημα εξετάζεται η περίπτωση όπου κάθε Δραστηριότητα μπορεί να έχει περισσότερες από μία κατηγορίες. Σε αυτό το έργο κάθε δραστηριότητα είχε μόνο μία κατηγορία. Αλλά στην πραγματικότητα, μια Δραστηριότητα μπορεί να εμπίπτει σε πολλές κατηγορίες. Έχοντας πολλές κατηγορίες αλλάζει ουσιαστικά τον τρόπο επίλυσης αυτού του προβλήματος. Μια ερώτηση είναι πώς τα διαφορετικά βάρη πρέπει να συνδυαστούν με αυτές τις κατηγορίες.
Μια άλλη λεωφόρος που πρέπει να εξερευνήθεί είναι το πώς πρέπει να χρησιμοποιηθεί η τυχαιότητα. Ο αλγόριθμος θα μπορούσε να αρχίσει ντετερμινιστικά για υψηλή εκμετάλλευση και με την πάροδο του χρόνου να χαλαρώνει για να επιτύχει καλύτερη εξερεύνηση. Το χρονοδιάγραμμα αυτής της χαλάρωσης και το πώς σχετίζεται με το μέγεθος του χώρου αναζήτησης είναι μια άλλη ενδιαφέρουσα ερώτηση.
Σε αυτή τη διπλωματική διατριβή έχουμε ασχοληθεί με την εξαρχής επίλυσει μίας μόνο περίπτωσης. Ωστόσο, η επίλυση πολλαπλών περιπτώσεων στην ίδια τοπολογία θα μπορούσε να μας δώσει νέους τρόπους προσέγγισης του προβλήματος. Για παράδειγμα, αντί να χρησιμοποιήσουμε SGD θα μπορούσαμε να εκπαιδεύσουμε ένα μοντέλο για να μας παράσχει ένα a priori βάρος για να χρησιμοποιήσουμε λαμβάνοντας υπόψη όλες τις μεταβλητές εισόδου. Επιπλέον, η επίλυση πολλών περιπτώσεων παράλληλα θα μπορούσε να θέσει τις δικές του προκλήσεις. Πρέπει οι περιπτώσεις να είναι συσσωρευμένες ή απλά να έχουν πρόσβαση σε στατικές πληροφορίες και κάθε μία από αυτές να λυθεί σε διαφορετική CPU είναι η πιο αποτελεσματική προσέγγιση που μπορούμε να έχουμε;

Δεδομένου ότι ο νόμος του Moore έχει παραμείνει για 35 χρόνια από την σύλληψη του Προγράμματος Προσανατολισμού και καθώς έχουν σχεδιαστεί όλο και καλύτερες ευριστικές για την επίλυση αυτού του προβλήματος, έχουμε αποκτήσει την ικανότητα να επιλύουμε περιπτώσεις για όλο και μεγαλύτερες τοπολογίες. Ωστόσο, όλα τα σύγχρονα σύνολα δεδομένων εξακολουθούν να είναι γύρω στους 100 κόμβους, μακριά από τα πραγματικά σύνολα δεδομένων όπου υπάρχουν εκατό Δραστηριότητες ακόμη και σε μια μικρή πόλη και χιλιάδες στις μεγάλες μητροπόλεις σε όλο τον κόσμο. Το να είναι σε θέση να κλιμακώσει αυτές τις λύσεις στις χιλιάδες και τα εκατομμύρια είναι ζωτικής σημασίας για τη δυνατότητα μετασχηματισμού αυτού από μία θεωρητική δημοσίευση σε μια λύση που πραγματικά βοηθά τους ανθρώπους να λαμβάνουν πιο ενημερωμένες αποφάσεις.
Αναζητώντας τη βιβλιογραφία σε αυτό το πρόβλημα παρατηρούμε ότι όλοι οι αλγόριθμοι εκτελούν ένα καινούργιο ξεκίνημα. Αυτό σημαίνει ότι όταν αρχίζουν να επιλύουν μια νέα περίπτωση του προβλήματος, δεν χρησιμοποιούν καμία προηγούμενη γνώση και εμπειρία. Για τους περισσότερους κοινούς αλγόριθμους, οι διάφορες πιθανές είσοδοι που μπορούν να ληφθούν είναι τόσο πολλές που δεν είναι δυνατή η απομνημόνευση. Ωστόσο, χρησιμοποιώντας μηχανική μάθηση θα μπορούσε κανείς να υποθέσει ότι μετά την επίλυση δισεκατομμυρίων διαφορετικών περιπτώσεων για μια πόλη θα μπορούσε να δημιουργηθεί ένα μοντέλο που θα μπορούσε να λύσει αποτελεσματικά οποιαδήποτε από αυτές τις περιπτώσεις καλύτερα και ταχύτερα από οποιονδήποτε από τους κοινούς αλγόριθμους. Ακόμη και αν κάποιος δεν στραφεί στη Μηχανική Μάθηση, η παράλληλη επίλυση πολλών περιπτώσεων στην ίδια τοπολογία θα ήταν ένα ενδιαφέρον πρόβλημα τόσο από την άποψη της μηχανικής όσο και από την άποψη της πληροφορικής.

Chapter 1

Introduction

1.1 Orienteering as a sport

> Orienteering is a group of sports that requires navigational skills using a map and compass to navigate from point to point in diverse and usually unfamiliar terrain whilst moving at speed. Participants are given a topographical map, usually a specially prepared orienteering map, which they use to find control points. Originally a training exercise in land navigation for military officers, orienteering has developed many variations. Among these, the oldest and the most popular is foot orienteering.
>
> From Wikipedia, The free encyclopedia

Orienteering as a sport originates in late 18th century Sweden as military training. The actual term "orienteering" was first used in 1886 at the Swedish Military Academy Karlberg. The first competitive event was held for Swedish military officers on 28 May 1893 at the yearly games of the Stockholm garrison. The first civilian competition was held four and a half years later, in Norway on 31 October 1897 and was sponsored by the Tjalve Sports Club.
The sport gained popularity with the development of more reliable compasses in the 1930s. In 1932 the first international event between orienteers from Sweden and Norway was held outside Oslo, Norway. In the subsequent years, the sport was spread to Finland, Switzerland, the Soviet Union and Hungary. By the end of the 1970s, orienteering had spread all over the world, with many countries having established national federations and championships.
Orienteering events can be distinguished by

- Method of travel: Foot, Bike, Ski, Canoe, Car
- Length: Ultrasprint, Sprint, Middle, Long
- Time the competition is held: Day, Night
- Order of controls: cross-country (specific order), score (free to decide order)
- Number of competitors: Individual, Team, Relay

Ultrasprint events are held in specially constructed labyrinths and allows for simultaneous local competitions to be held concurrently in different geographical locations as parts of a larger tournament. Sprint events are usually 12–15 minutes long and are usually set in urban settings or parks. Middle races are about 30 minutes long, while the winner for long races takes about 75–90 minutes to complete the race.

In cross-country races, all contestants must follow the same route and placement is decided in order of completion. Usually, contestants are spaced out in a staggered start. On the other hand, in score races, there are usually mass starts with different points for each control point, depending on difficulty and one point per minute penalty. Of particular interest is the large-scale, endurance-style version of a Score-O known as rogaine, in which some events last (often) 24 hours. For these kinds of competition, which originated in Australia, very large areas are used.
In relay races, several athletes run different parts of the course and the total amount of the team's time is taken into consideration. To avoid having athletes group together, various techniques have been applied. Team orienteering usually follows the score format and the members of the team disperse to capture as many of the control points in as little time as possible.

1.2 Orienteering in Combinatorial Optimization

In Computer Science the Orienteering Problem (OP) is focused on the score variant of the sport. In terms of the field, given N points each giving a different score and a cost to go from each point to another, we need to devise a path from a starting to finishing point whose cost does not exceed a predetermined constant. Imagine a friend who visits your home city and asks you to give him a suggestion of where to go for the few days that he will be visiting. He decides to go out every day at 9:00 AM and be back at his hotel by 8:00 PM. In your home city there are many interesting places, but he doesn't have the time to visit and appreciate all of them. Already you are making a mental list of the best places to visit, along with what lies close to them and would be worthwhile. Then, you try to order the candidate visits in days and succession, based on what is close. You are already solving an instance of the OP! While you are also taking into account thematic days and maybe including a day trip somewhere close, the original problem formulation was a bit more mundane.
The OP was first introduced as the Constrained Traveling Salesman Problem. In the Traveling Salesman Problem we need to tour all nodes in the minimal time possible. On the other hand, in the OP there is a limit to the time we have to tour nodes. In CS, the OP is formulated as an optimization problem focusing on which control points should be visited and in what order. Beyond the obvious use case in solving question about the sport, many practical problems can be reduced to the OP.
In the last thirty years the Orienteering Problem (OP) has come from relative obscurity to a central place in the area of discrete optimization. Starting from a highly abstracted problem in the crossroads of graph theory and optimization, much more concrete variants and solutions have been introduced during the recent years. As the OP matures, solutions for its many variants become more and more suited for real life applications as gradually more real life constraints and peculiarities are inserted in the problem definition. An example of such a peculiarity is the temporal nature of the underlying graph, either in the possible choices at each step of the solution, or the time it takes to transition from one node to another, much as it happens in a physical transportation network.
Solutions to the OP include many operational research problems, such as logistics or routing vehicles for deliveries of goods or services. Other applications are the Tourist Trip Design Problem (which we mentioned a few paragraphs ago), deciding how to spend your time in a theme park, routing information collecting drones and the bank robber problem, where you have to choose which banks to steal from before your car gas tank runs out (police is not taken into consideration).

In this Diploma Thesis, we introduce a new variant of that problem, the Minimum-Maximum Constraints Orienteering Problem with Time Windows (MMCTOPTW). This new variant comes from the real life example of the Tourist Trip Design Problem (TTDP), where in each day the tourist wants to have the best time and get the most out of his short visit, whilst at the same time satisfying some constraints. One such constraint is to visit exactly one restaurant. If he visits no restaurants he will go hungry. On the other hand, visiting a second restaurant will give him no further utility.
As a way to solve this new variant, an ILS algorithm is considered and extensive experimental results are conducted to help evaluate parameter choices. The results of this implementation are then compared to state of the art algorithms in the same topologies. Finally, a speculation of other candidate solution algorithms is presented and possible future extensions.

1.3 Motivation

As we have highlighted, many problems can be reduced to an instance of the OP. We have already mentioned the TTDP. Other applications include the routing technicians, where a company has to decide where to send each technician in a single day. Other applications that can be reduced to the OP are choosing rides in an amusement park and routing drones for information collection above candidate targets. For that reason, solutions to the OP are of extreme interest to both the scientific and the industrial community. The application that will be the main focus of this thesis is the Tourist Trip Design Problem (TTDP) (Gavalas et al. (2014a), Souffriau et al. (2008), Wörndl (2016)). Until this point, all POIs were considered to be of the same category, meaning that they were interchangeable. But in real life, we might want to include a POI of a certain category. The example we gave above is that of a restaurant, but it could also be anything else, such as a museum or a concert.
Although minimum constraints have been briefly considered in Sylejmani et al. (2012), as of the writing of this thesis, to the best of the author's knowledge, there is no previous work dealing with minimum constraints.

1.4 Thesis contribution

The main contributions of this work are the following:

1. Design and implementation of an ILS algorithm for the solution of the MMCTOPTW.
2. Comparison with existing algorithms in the unconstrained setting.
3. Comparison of the constrained settings to the unconstrained.

1.5 Chapter outline

In Chapter 2, we present the history, bibliography and necessary technical background that is needed to understand the Orienteering Problem. A few of the most cited variants are introduced along with an overview of the proposed solutions. We finish by introducing the mathematical formulation for the Orienteering Problem, and the variant that will be the focus for the rest of the thesis, the Minimum-Maximum Constraint Team Orienteering Problem with Time Windows.

In Chapter 3, we take a further look into the approach of Vansteenwegen et al. (2009b), on whose work this thesis is largely based. First, we will take a closer look into the methodology followed in the aforementioned paper and how this work furthers that approach. Then we present the proposed modified algorithm.
In Chapter 4 through experimentation on available datasets, we determine the optimal parameters that appear in our algorithm, in order to achieve the best result. Questions that arise consider the stochastic versus deterministic, the greedy criterion that should be used, whether using 2-OPT moves will provide us with a better result and finally, the choice of the weight we need to give to our new part of the greedy criterion. Firstly, we show that for a given weight the categories of each minimum follow a normal distribution. Then we show that a static value of this weight is not optimal and finally we explore a form of Stochastic Gradient Descent (SGD) to arrive to an optimal value, at the same time as solving the problem.
In Chapter 5, we will first compare our algorithm in the unconstrained scenario against the ILS solution our work is based on. Later in the chapter we will establish the smooth decrement of the achieved result as the constraints become tighter
In Chapter 6, we list future work that could be undertaken in the problem and new venues to approach it. Furthermore, we propose augmentations to the current approach and additional variants similar to this.

Chapter 2

Background

2.1 History

The Orienteering Problem (OP) was first described in Golden et al. (1981), but the term OP was first introduced in Tsiligirides (1984). Subsequently, Golden et al. (1987) proved that the OP is NP-hard. OP has stemmed from the TSP and many papers have refer to it as constrained TSP (Laporte and Martello (1990); Gendreau et al. (1998a); Thomadsen and Stidsen (2003)). A very slight variation of the OP is the Team OP (TOP). In this variant, instead of seeking just one, M routes are requested, each of which may start and end at the same vertices, but share no other nodes. Some of the practical applications of the OP include the traveling salesman with insufficient time to visit all cities in Tsiligirides (1984) and the home fuel delivery problem in Golden et al. (1987), where a fleet of trucks has to deliver to a large number of customers on a daily basis and each customer's fuel inventory level must be maintained at an adequate level at all times. Hence, each customer is awarded an urgency score depending on his current or projected fuel inventory level. Determining the subset of customers to service each day and the path of the trucks is essentially the OP. Others include the Mobile Tourist Guide and Tourist Trip Design Problem (Thomadsen and Stidsen (2003); Vansteenwegen and Van Oudheusden (2007)). For the original OP, several benchmark instances exist. These can be found in Tsiligirides (1984), Chao et al. (1993), Chao et al. (1996b) and Fischetti et al. (1998). Additionally, several more exist for the TOP and can be found in Chao et al. (1996a) and Dang et al. (2013b). In total, 385 instances are available with 21 to 500 vertices for the OP and 472 instances with 21 to 399 vertices for the TOP. More information can be found in Vansteenwegen et al. (2011) and Gunawan et al. (2016). A list of the details for these datasets can be found at A.1

To tackle the OP several exact as well as heuristic algorithms have been proposed. The exact solutions can be found in table A and the heuristic solutions in table A. The exact solutions include Branch and Bound (Laporte and Martello (1990), Ramesh et al. (1992)), Branch and Cut (Fischetti et al. (1998), Gendreau et al. (1998a), Feillet et al. (2005), Dang et al. (2013a)), Branch and Price (Boussier et al. (2007)) and Cutting Plane (Leifer and Rosenwein (1994)) algorithms. Each one of these papers plays on finding the optimal solutions as we unfold the search space.

On the other hand the heuristic approaches follow the trend of the field. While before 2000 we see hand crafted algorithms like Centre of Gravity (Golden et al. (1987)) and the Four-phase heuristic (Ramesh and Brown (1991)) or the Five-step heuristic (Chao et al. (1996b), Chao et al. (1996a)), in recent years the trend is applying known heuristics to this problem. In the new millennium we see a clear trend towards heuristic approaches instead of exact algorithms. During the last eighteen years there have been more than 19 heuristic approaches compared to about 6 papers discussing exact

algorithms. A plethora of metaheuristic optimization algorithms have been applied to the OP and the TOP. Tabu search (Gendreau et al. (1998b), Archetti et al. (2007)), several variants of Variable Neighborhood Search (Archetti et al. (2007), Vansteenwegen et al. (2009c), Liang et al. (2013)), a technique to which researches return to through the years and use to solve almost all variants of the TOP. Among other metaheuristics used are Ant Colony Optimization (Ke et al. (2008), Gambardella et al. (2012)), Iterated Local Search (Vansteenwegen et al. (2009b), Gunawan et al. (2015a)), Greedy Randomized Adaptive Search Procedure also known as GRASP (Campos et al. (2014), Marinakis et al. (2015), Particle Swarm Optimization aka PSO (ŞEVKLİ and SEVİLGEN (2010), Sevkli and Sevilgen (2010), Muthuswamy and Lam (2011b), Dang et al. (2011), Dang et al. (2013b)), Simulated Annealing (Vincent and Lin (2014)) and of course the other oldest metaheuristic algorithm, Genetic Algorithms (Ferreira et al. (2014)).

These attempts have built on one another and have met various degrees of success. While most of them have failed in producing better than the already known solutions, they have helped us into gaining a deeper understanding of the underlying problem. Some papers have tried to win in time instead of quality. Exact algorithms like Branch-and-Cut and swarming algorithms like PSO and ACO require non-negligible time of computation to arrive to a good result. On the opposite side of these, algorithms like Iterated Local Search (ILS) produce results of comparable quality (but not quite the same) in a very small fraction of the time. While for offline application, we can invest more time to achieve a better solution, in an online context or a system shared by many users where each computing cycle is weighted against the additional quality it brings to the final answer, algorithms that produce a good-enough solution in very little time seem very appealing.

While in the early years after the formulation of the problem, mainly exact solutions were proposed, most contemporary papers attempt to solve this problem utilizing heuristics. This shift is logical if we bear in mind that the OP is NP-hard and the flourish of the heuristic optimization field in the early 2000s. While in the beginning researchers aimed to solve even small instances with only a few nodes, modern researchers aim solving ever larger datasets and ever faster solutions. As computers now are used interactively (in contrast with batch processing jobs in the 1990s) our perception of time has changed and if we want to transfer these solutions to end users, solving durations of more than a few seconds can seem like an eternity and highly discourage users to continue using such a solution.

2.2 Variations

Over time, a number of variants have emerged, others stemming by real life applications and others from revisiting previous assumptions. The most notable among them are following:

- (Team) Orienteering Problem with Time Windows [(T)OPTW]
- Time Dependent Orienteering Problem [TDOP]
- Arc Orienteering Problem [AOP]
- Stochastic Orienteering Problem [SOP]
- General Orienteering Problem [GOP]

2.2.1 (Team) Orienteering Problem with Time Windows [(T)OPTW]

In the OPTW, each vertex is assigned a time window $[O_i, C_i]$ and a visit to a vertex can only start during this time window. In terms of graph theory, we substitute the static graph with a temporal one. This variation stems mostly from real life problems, such as the TTDP, where most attractions are not always open and especially some events might have a very specific starting time. This slight variation has a profound impact on solution methods. While in traditional (T)OP most solutions use some form of 2-OPT moves, in (T)OPTW this approach cannot be applied efficiently. Exchanging the visits between two vertices can lead to one or both of the vertices to be scheduled outside of their respective Time Windows and as a result, make the route invalid. The other difference that must be taken into account is that in this variation we might have to wait at a vertex for its opening and therefore adding a visit halfway across the route will move every other visit after it by a possibly different amount. In the classic OP if the next vertex is moved by X amount, then all further vertices have to move also by that amount. This does not hold for the OPTW.
Kantor and Rosenwein (1992) were the first to solve the OPTW. They first describe an insertion heuristic of "score over insertion time". This is the same greedy criterion that is used in the continuous Knapsack Problem. It has proven a very good baseline and will form the basis for our own algorithm later. Secondly, a depth-first search algorithm is proposed that constructs partial paths using the insertion heuristic and beginning in the start vertex. Righini and Salani (2009) designed an bi-directional dynamic programming algorithm, which is exact. In Vansteenwegen et al. (2009b) an Iterated Local Search (ILS) is proposed, where for each *insertion step*, all vertices are considered in between two included vertices and for each vertex the best possible place for insertion is chosen and then the vertex with the best $\frac{score^2}{shift}$ is inserted in the solution. Our approach is largely based on this work. After no more insertions can be performed the solution is evaluated, in the *shake step* a contiguous part of the solution is removed and we start again. Their approach has the unique trait of finding a solution in a matter of seconds, as opposed to most approaches which need a few minutes of computation. The algorithm finishes when there has been no improvement for 150 iterations. In Gavalas et al. (2015) two clustered algorithms are used in an effort to reduce the time spent on traveling. Gunawan et al. (2015a) using ILS and Gunawan et al. (2015b) using Hybrid Simulated Annealing and ILS (SAILS) are currently state-of-the-art algorithms for the (T)OPTW. Labadie et al. (2011) proposes a hybridization of a Greedy Randomized Adaptive Search Procedure (GRASP) and Evolutionary Local Search (ELS) algorithm (GRASP-ELS), Cura (2014) uses Artificial Bee Colony (ABC), yet another metaheuristic. More details can be found on Gunawan et al. (2016). Gambardella et al. (2012) has used Enhanced Ant Colony System (EACS), Lin and Vincent (2012) uses fast and slow Simulated Annealing, Labadie et al. (2012) an LP-based Granular Variable Neighborhood Search and Duque et al. (2015) the Pulse Algorithm. A lot of work is being done on this problem and garners the most attention after the original (T)OP. The switch from exact solutions to heuristics is very striking in the previously presented papers, in favor of the ability to solve larger datasets in a reasonable time frame. A more complete list can be found on A.

2.2.2 Time Dependent Orienteering Problem [TTDP]

In the OPTW we abandoned the notion of a static graph in terms of vertices. We couldn't visit any vertex, any time. In this variation, we abandon instead the notion of a static graph in favor of a

temporal one in terms of edges. How far away two nodes are depends not only from the nodes, but is also a function of time. This variation also stems from real life problems. One only needs to think of traffic jams every morning and afternoon in his favorite city, where the nodes are stationary and the time to travel from one to another changes over time or of moving targets, whose relative distance is obviously a function of time. Another application arises from the schedules of public transportation where the time we have to wait for a train depends on the time we get to the station. Another interesting application is navigating in a theme park where going from one ride to the next depends mostly on the time you will have to wait in line waiting for your turn. That time is a function of time depending on the peak hours of the theme park.
This variation was first introduced in Fomin and Lingas (2002). The solutions that have been proposed include Dynamic Programming (Li (2012)) ILS (Gunawan et al. (2014), Garcia et al. (2010)) which extends the work from Vansteenwegen et al. (2009b), ACS (Verbeeck et al. (2014)) and Adapted Genetic Algorithm (Abbaspour and Samadzadegan (2011)). A contains more information on each of these solutions.

2.2.3 Arc Orienteering Problem [AOP]

This variation assigns profits to the edges instead of the vertices. Examples for this problem are taken from telecommunications and transportation. It has been shown that an AOP instance can be reduced to an OP instance. Proposed solutions include ILS, various approximation algorithms by Gavalas et al. (2015), Branch-and-Cut, and Hybrid Tabu Search and diversification phase with the exact solution of ILP models.

2.2.4 Stochastic Orienteering Problem [SOP]

While up until now, all sizes in the problem were deterministic, there are a few variations of the OP were one or more sizes are associated with a probability or a distribution. In Orienteering Problem with Stochastic Profits (OPSP) the profit accrued by a vertex is a normal distribution. On the other hand, in Orienteering Problem with Stochastic Travel and Service Times (OPSTS) the time taken to complete a visit to a vertex or traversing an edge is taken from a known distribution. A practical example of OPSP in the TTDP is having to plan the trip with only seeing the reviews for each candidate place and not knowing before hand how much you will enjoy actually going there. On the other hand the OPSTS is going to the bus stop without having seen the timetable. When the bus will pick you up is a stochastic amount. Another proposed application is going to a theme park where the length of the line at each attraction comes from a distribution. Usually, the goal of the solutions to such problems is to guarantee a certain threshold with good probability. That you will reach you hotel before 8:00 PM with a probability greater than 90% or that you will collect above of a certain amount of profit.
Proposed solutions include bi-objective Genetic Algorithm, Variable Neighborhood Search, Monte Carl sampling, Mixed Integer Linear Programming Approximation, Local Search and Adaptive Variable Neighborhood Search. More information about these solutions can be found on table A

2.2.5 General Orienteering Problem [GOP]

The difference in this variation lies in the objective function. In the previous variations, the objective function was implicitly defined as the sum of the profits of the visited nodes (or edges in the case of AOP). In this variation, the objective function can be super- or sub-modular. This problem tries to capture relationships between vertices that might benefit (or not) from the selection of another node. An example of a cooperative relationship could be visiting Acropolis first and then the adjacent museum of Acropolis. A competitive relationship could be visiting two restaurants back to back. In Geem et al. (2005) each node was associated with a vector of attributes $S(i) = (S_1(i), S_2(i), \ldots, S_g(i))$. Each attribute also has a weight W_j. The final objective function is listed in 2.2.5.

$$Maximize\overline{Z} = \sum_{j=1}^{g} W_j[\{\sum_{i \in P}[S_j(i)]^k\}^{1/k}] \tag{2.1}$$

Furthermore, $\sum_{j=1}^{g} W_j = 1$ and k in a non-negative exponent. Of course, by setting $g = 1$ and $k = 1$ we get the original OP. As k approaches infinity, $\overline{Z}$ approaches Z, where

$$MaximizeZ = \sum_{j=1}^{g} W_j\{\max_{i \in P}(S_j(i)\} \tag{2.2}$$

Papers on GOP are listed in table A.

2.3 Definition and Mathematical Formulation

In this section, we will give a formal definition of the OP and of the MMCTOPTW. Let there be a full weighted undirected graph with N vertices, each with a profit P_i and a visiting time V_i. Let the weight of the edge e_{ij} denoted t_{ij} represent the time needed to move from vertex i to vertex j. The values of t_{ij} adhere to the triangle inequality, meaning $\forall i, j, k : t_{ij} \leq t_{ik} + t_{kj}$. Let there be a starting vertex S and an ending vertex T. The goal of the OP is to determine a simple path from vertex S to vertex T of duration less than or equal to some value $T_{\max}$, such that it maximizes the $\sum P_i$ of the vertices in the path.

Mathematically, it can be formulated as an Integer Programming (IP) problem. The mathematical formulation of the IP problem can also be seen in Vansteenwegen et al. (2011). Let $x_{ij} = 1$ when a visit to vertex i is followed by a visit to vertex j - 0 otherwise. Let u_i be the position of vertex i in the path.

$$Max \sum_{i=1}^{N-1} \sum_{j=2}^{N} P_i x_{ij} \tag{2.3}$$

$$\sum_{j \neq S} x_{Sj} = \sum_{i \neq T} x_{iT} = 1 \tag{2.4}$$

$$\sum_{i=1}^{N} x_{ik} = \sum_{j=1}^{N} x_{kj} \leq 1; \quad \forall k \neq S, T \tag{2.5}$$

$$\frac{V_S + V_T}{2} + \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}(t_{ij} + \frac{V_i + V_j}{2}) \leq T_{\max} \tag{2.6}$$

$$1 = u_S < u_i < u_T = \max(u_i); \quad \forall i \neq S, T \tag{2.7}$$

$$u_i - u_j + 1 \leq (u_T - 1)(1 - x_{ij}) \quad \forall i, j \tag{2.8}$$

$$x_{ij} \in \{0, 1\} \tag{2.9}$$

The objective function 2.3 maximizes the total collected score. Constraints 2.4 ensure that the path starts at vertex S and ends at vertex T. Constraints 2.5 ensure the continuity of the path and guarantee that each vertex is visited at most once. Constraints 2.6 enforce the time budget limitation. Constraints 2.7 and 2.8 are necessary to prevent subtours. Because of the previous constraints each vertex has in and out degree of at most 1. From this, we can deduce that other than the isolated vertices, the graph will contain paths and cycles. Subtours are the cycles in this graph. We are looking for a single path and therefore we include these constraints to eliminate the cycles. These subtour elimination constraints are reiterated according to the Miller-Tucket-Zemlin (MTZ) formulation of the TSP (Miller et al. (1960)).

An extension to the original OP, that will be of interest in this thesis, is the OP with Time Windows (OPTW). In this extension, each vertex can be visited only during the time frame $[O_i, C_i]$. Another extension of the OP is the Team OP (TOP), in which D paths are requested instead of a single path, all starting and ending at the same positions, but sharing no other nodes. Therefore, each activity can be visited only once in all the routes. Finally, the extension presented by this thesis is the Minimum-Maximum Constraint TOPTW (MMCTOPTW), where in addition to the previous features, every vertex has a category vector $\overline{K_i}$. Additionally, the two vectors m and M denote the minimum and maximum constraints of the problem respectively. As a result of the all the above, the mathematical formulation of the IP needs to be revisited. In the new formulation, one more subscript will be added to most of our variables to indicate the day that we refer to. Moreover, additional constraints will be introduced to express the Time Windows constraints and the Minimum-Maximum constraints. In order to express the Time Windows constraints, the variable s_{id} states the start of service at activity i on route d.

$$Max \sum_{d=1}^{D}\sum_{i=1}^{N-1}\sum_{j=2}^{N} P_i x_{ijd} \tag{2.10}$$

$$\sum_{d=1}^{D}\sum_{j\neq S} x_{Sjd} = \sum_{d=1}^{D}\sum_{i\neq T} x_{iTd} = D \tag{2.11}$$

$$\sum_{i=1}^{N} x_{ikd} = \sum_{j=1}^{N} x_{kjd} \leq 1; \quad \forall k \neq S, T; \forall d \tag{2.12}$$

$$\frac{V_S + V_T}{2} + \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ijd}(t_{ij} + \frac{V_i + V_j}{2}) \leq T_{\max}; \quad \forall d \tag{2.13}$$

$$s_{id} + T_i + t_{ij} - s_{jd} \leq M(1 - x_{ijd}); \quad \forall i, j, d \tag{2.14}$$

$$1 = u_{Sd} < u_{id} < u_{Td} = \max(u_i); \quad \forall i \neq S, T \tag{2.15}$$

$$u_{id} - u_{jd} + 1 \leq (u_{Td} - 1)(1 - x_{ijd}); \quad \forall i, j \tag{2.16}$$

$$O_i \leq s_{id} \leq C_i; \quad \forall i, d \tag{2.17}$$

$$|K_i|_1 = 1; \quad \forall i \tag{2.18}$$

$$\overline{m} \leq \sum_{d=1}^{D}\sum_{i=1}^{N-1}\sum_{j=2}^{N} \overline{K_i} x_{ijd} \leq \overline{M} \tag{2.19}$$

$$x_{ijd} \in \{0, 1\} \tag{2.20}$$

The objective function 2.10 now sums over all routes. Constraint 2.11 ensures that each day starts from S and ends at T. Constraints 2.12 provide the continuity of the path in each day and ensures that each vertex is visited at most once in all days, same as above. Constraints 2.13 ensure that each route's time budget is not exceeded. Constraints 2.14, 2.15 and 2.16 are necessary to prevent subtours according to the MTZ formulation of the TSP. Constraints 2.17 ensure that start times are within the allowed time frame. Constraints 2.18 ensure that each vertex belongs to exactly one category. Constraints 2.19 express the minimum and maximum allowed vertices of each category.

Chapter 3

Design and Implementation

3.1 Introduction

In this chapter we will first give a high level description of the algorithm in Vansteenwegen et al. (2009b), then delve deeper into the technicalities and finally, present our own algorithm.
We believe that the term ”POI” used in the referenced paper does not capture the whole meaning of what we want to express with a visit, as it expresses mainly a spatial entity while we want to give emphasis to its temporal nature as well. For that reason, from now on, we will refer to them as Activities, which are taking place in a location, but also have a Time Window. This notion, better captures events such as concerts, giving an emphasis to the fact that while many Activities can take place in the same place, each one is different.
The proposed solution is an Iterated Local Search (ILS). The original algorithm is described below. In every iteration, Activities are inserted greedily into the current solution using a heuristic to rate possible candidates, until no more can be inserted. After inserting all possible Activities, the new solution is evaluated. If it is better than the best found solution, we make this the best solution. At the end of the loop, we shake parts of the current solution in order to be able to create a new one. We repeat this procedure until the solution has not been improved for 150 iterations. As Activities are inserted and then removed, the produced solutions are essentially samples from a distribution from which we keep the one with the highest profit.
Basically, we can divide the algorithm into four discrete steps. Initialization, insertion, evaluation and shaking. First we initialize the paths, then in a loop insert as long as possible, evaluate the new solution, shake parts of it, in order to acquire a new one in the next step.
While on first reading this might seem like an easy algorithm, there are quite a few implementation trappings that must be taken into consideration. After we have found the next Activity to insert and where to insert it, keeping our data structures consistent for the next iteration is not trivial. Due to the Time Windows, as we have mentioned previously, each subsequent starting time needs to be recalculated. The other question that becomes more difficult is whether a certain Activity can be inserted in the solution. In the TOP, keeping a path variable of how much more time we can use for new POIs is enough, but in the TOPTW a variable for each Activity is required and needs to be updated after every single insertion. This variable is called maximum shift for. It represents how much later this Activity can start.
A major improvement in terms of execution time is not considering all Activities in each iteration. If we were unable to reach an Activity in the previous iteration, then after adding one more Activity to our solution we will also be unable to reach it. Due to the time windows, we might arrive for an

Listing 3.1: Pseudo code for the insertion step.

```
For each non included Activity:
|   Determine the best possible insert position and Shift;
|   Calculate Ratio;
Insert Activity with highest ratio (j);
Node j: calculate Arrive, Start, Wait;
For each Node after j (until Shift == 0):
|   Update Arrive, Start, Wait, MaxShift, Shift;
Node j: update MaxShift;
For each Node before j:
|   Update MaxShift;
```

Activity earlier than its opening time. In that case, we will have to wait. This is another variable that needs to be updated in every modification of the solution.

When inserting a new Activity Vansteenwegen et al. (2009b) takes into account two things: Time and Profit. While profit is straightforward in the additive setting, time needs to take into account both visit and travel times. To that end, the authors have defined $Shift$, which is how much later we would arrive to the next Activity if we first went to the new Activity. They have combined these two in a very elegant metric $Ratio = \frac{(Profit)^2}{Shift}$. For each candidate Activity the minimum $Shift$ is used. Finding the minimum shift requires trying to insert the Activity between every two consecutive Activities present in the solution. For large instances, the complexity of finding this minimum shift is not negligible.

Next, we will focus on the shake step. After each solution has been evaluated, we need to shake a part of it, so that a new, different solution can be created. In their algorithm, they try to shake less than 1/3 of the current solution. Starting over doesn't offer anything as the solution is deterministic. To achieve that, after each shake, they increase the number of Activities to drop by 1. Also the starting place to perform the shake is increased by the number of Activities dropped.

The formulas for updating various quantities are summarized in 3.1. The original pseudo code is listed below in 3.1, 3.2 and 3.3.

Table 3.1: Update formulas.

Name	Formula	Name	Formula
$Wait_i$	$\max[0, open - arrive]$	$MaxShift_i$	$\min[Close_i - Start_i, Wait_{i+1} + MaxShift_{i+1}]$
$Shift_j$	$t_{ij} + Wait_j + Visit_j + t_{jk}$	$Ratio_i$	$(Profit_i)^2/Shift_i$

3.1.1 Algorithm

The original algorithm needs to be revisited to take into consideration the minimums and maximums. Maximums are very easy to honour. The only modification needed is in the insert step, where the Activities that are considered, would not violate them if they were inserted. On the other hand, implementing a solution to respect the minimums set is not as straightforward. Obviously, small minimums of frequent categories in large tours will always be satisfied, even with the original algorithm. However, when the constraints are more exacting, the solutions produced by the original

Listing 3.2: Pseudo code for the shake step.

```
For each tour:
|    Delete the set of visits (i => j);
|    Calculate Shift;
|    For each Node after j (until Shift == 0):
|    |    Shift Node towards the beginning of the tour;
|    |    Update Arrive, Start, Wait, MaxShift, Shift;
|    For each Node before i:
|    |    Update MaxShift;
```

Listing 3.3: Pseudo code for the ILS heuristic.

```
S ← 1;
P ← 1;
NumberOfTimesNoImprovement ← 0;
while NumberOfTimesNoImprovement < 150 do
|    while not local optimum do
|    |    Insert;
|    If Solution better than BestFound then
|    |    BestFound ← Solution;
|    |    R ← 1;
|    |    NumberOfTimesNoImprovement ← 0;
|    Else
|    |    NumberOfTimesNoImprovement ← NumberOfTimesNoImprovement+1;
|    Shake Solution (R, S)
|    S ← S + R;
|    R ← R+1;
|    If S>=Size of the smallest Tour then
|    |    S ← S−Size of smallest Tour;
|    If R==n/(3*m) then
|    |    R ← 1;
Return BestFound;
```

algorithm will most likely be invalid. Therefore, the heuristic must be improved upon to yield more feasible solutions more frequently.

$$Ratio_i = \frac{(Profit_i)^2}{Shift_i}(1 + W\frac{Demand[Category_i]}{Supply[Category_i]}) \tag{3.1}$$

This rule will be mentioned from now on as the "squared profit, multiplicative demand over supply". While obtaining infeasible solutions is still possible, the way that the solutions are generated, prioritize (to some degree) Activities which belong to categories that have been requested. As it will be shown, this simple modification is enough to obtain high quality solutions to the MMCTOPTW. Another modification that has helped produce better quality solutions is inserting a little randomness. This is achieved by multiplying $Ratio_i$ by a random number drawn from a uniform distribution. Naturally, these modifications give rise to the question of values that these parameters should have. Although the original authors did not utilize 2-OPT moves, we will try a few variations that might help us in obtaining better quality results. The pseudocode for the 2-OPT can be found in 3.4.

Listing 3.4: Pseudo code for the 2-OPT.

```
HasChanged ← False;
for every Route in Solution
|   for 0 < i < length(Route)
|   |   for i < j < length(Route)
|   |   |   TravelBefore ← TravelTime(Route[i−1], Route[i])
|   |   |                + TravelTime(Route[j], Route[j+1]);
|   |   |   TravelAfter  ← TravelTime(Route[i−1], Route[j])
|   |   |                + TravelTime(Route[i], Route[j+1]);
|   |   |   if TravelBefore > TravelAfter
|   |   |   |   Construct NewRoute;
|   |   |   |   if (NewRoute is feasible) and
|   |   |   |      (NewRoute has greater MaxShift)
|   |   |   |   |   replace Route with NewRoute;
|   |   |   |   |   HasChanged ← True;
return HasChanged;
```

3.1.2 Description of Dataset

For all computations, the dataset from Gavalas et al. (2015) with 113 sites (museums and art galleries, archaeological sites, monuments and landmarks, streets and squares, neighbourhoods, churches and religious heritage, parks), which are mostly situated around Athens down-town and Piraeus areas has been used. From the above described sites 20 different "topologies" have been generated, maintaining the real coordinates and hence the travelling times, however their respective profits, visiting hours, opening and closing hours have been randomized. To each of these topologies a category classification has been added. Also, a category has been added to each Activity at random as shown in 3.2.
Furthermore, the datasets from Vansteenwegen et al. (2009b) have been utilised to give a clearer image of how the various algorithms cope with different datasets. The same randomization has been used to generate categories for these topologies as well.

Table 3.2

Category	1	2	3	4	5	6
Probability	30%	20%	20%	10%	10%	10%

The dataset is completed by 144 equispaced locations, which can be used as starting positions.
Below, in figure 3.1, all activities and areas are depicted on the map of Attica. The size of the marker is proportionate to the profit of the associated Activity. The color of the marker is bluer for activities with small visiting times and redder for activities with large visiting times. The map of areas follows in 3.2

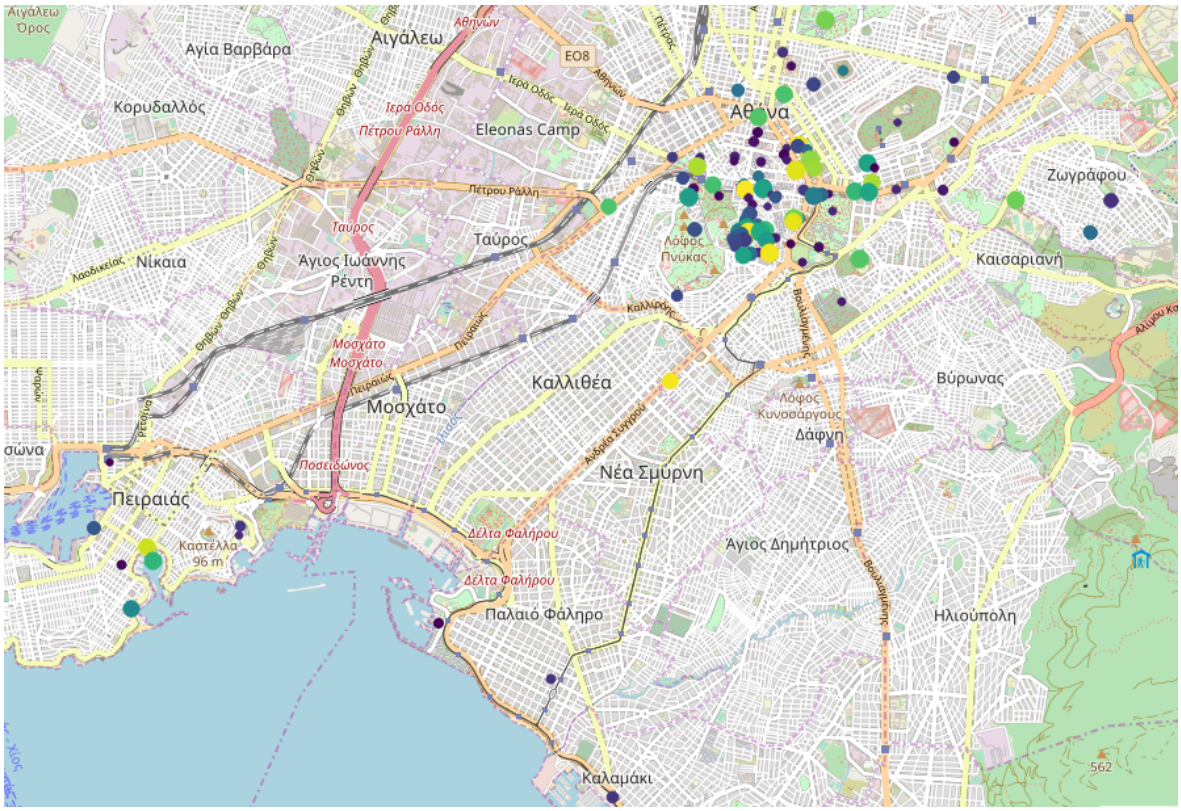


Figure 3.1: Map of Activities

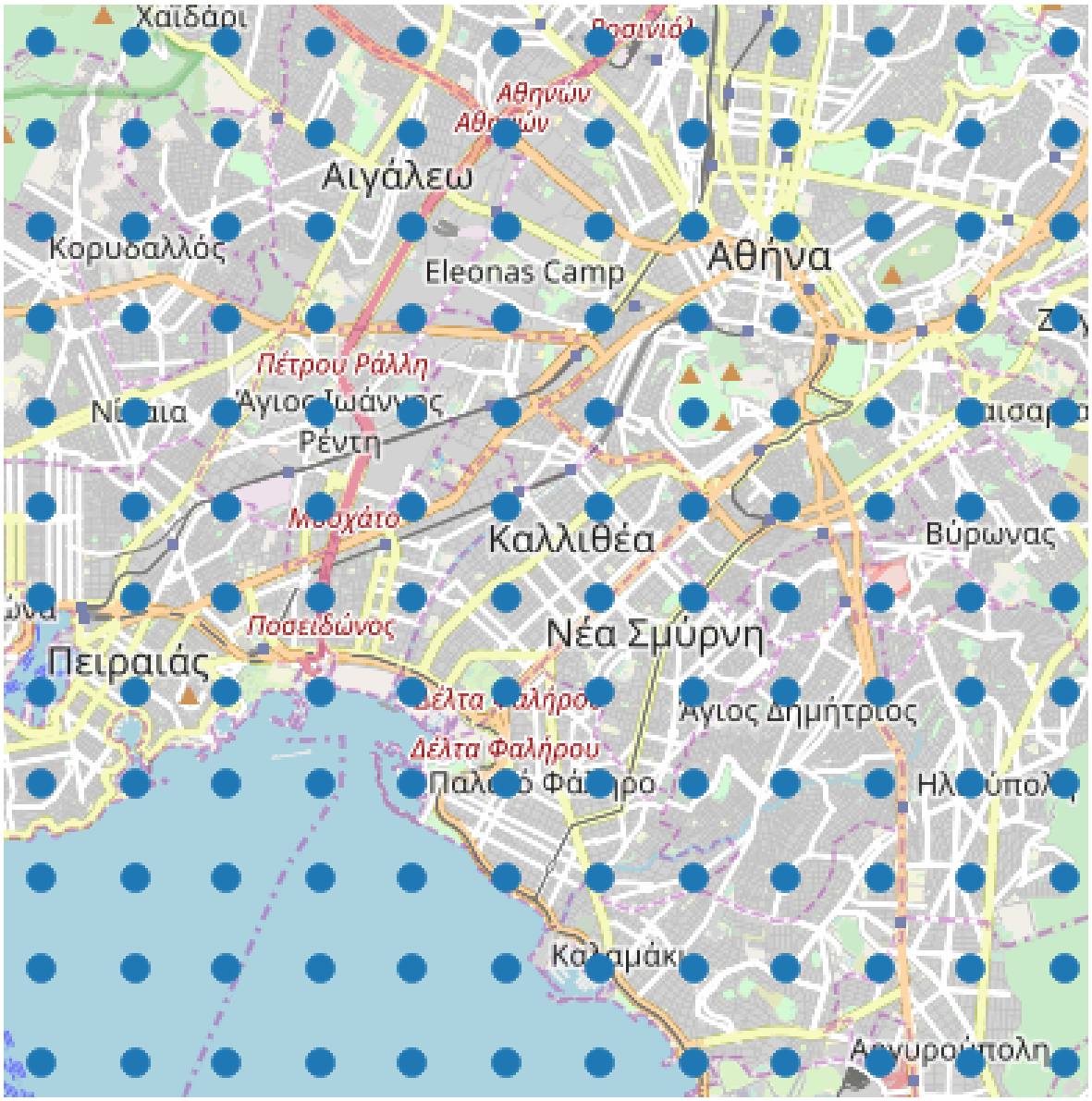


Figure 3.2: Map of Areas

Obviously only valid areas will be used as starting and finishing points and not those that are unreachable (e.g. underwater).

3.2 Code

3.2.1 Methods and Data Structures

The final solution consists of routes. Each route consists of an ordered list of Nodes. The members of the Node are listed in 3.3. The properties of Node are listed in 3.4.
The solving system has been prototyped in Python 2.7 and optimized to run in Rust. The solver needs in its initialization the following:

1. The number of routes
2. The start and end times for each day

Table 3.3: Members of Node

Name	Type	Description
activity	Activity	Pointer to the associated activity
arrive	int	Time that we arrive for this activity
start	int	Time that the activity is started
max_shift	int	Time that this activity can be pushed forward

Table 3.4: Properties of Node

Name	Type	Description	Formula
wait	int	Time between arrive and start	start-arrive
max_start	int	The latest time at which the activity can be started	arrive+max_shift
leave	int	Time of departure from the Activity	start+visit

Table 3.5: Methods offered by the solver

Name	Arguments	Description
solve		The main method of the solver. Returns a list of lists of Nodes.
insert	curr_solution	Inserts a single new Activity to the current solution. If no Activity can be added, then it returns False.
shake	curr_solution	Removes a number of Activities from each route.
2-OPT	curr_solution	Performs 2-OPT moves on the current solution, increasing the maximum allowed shift.

3. The available activities
4. The starting and ending area
5. The minimums required from each category
6. The maximums allowed from each category

The methods of the Solver are listed in 3.5.

Chapter 4

Hyperparameter optimization

A number of parameters have been introduced and the choice of values for each of these must be considered. While most of the parameters have one value that is preferable over all others and therefore can be safely used in all runs of the algorithm, the optimal value for some of them changes depending on the setting. In this section, we will analyse the different alternatives we considered for each one and present experimental results for our choices.

4.1 $Ratio$ function

The first choice that has to be made is that of the augmented ratio function. The main decision is whether to use as the nominator $(profit)$ (which is the standard heuristic for the corresponding knapsack problem) or $(profit)^2$ (which is proposed by Vansteenwegen et al. (2009b)). To answer that question, we ran the unconstrained setting for every topology for 100 times and we will compare the results in respect to average profit and average time to completion. The results can be seen in figures B.1-B.6.
In the first dataset, it is clear that $profit^2$ takes significantly less time, but has sub par results in terms of profit for most topologies. The reason for the reduced times is that this ratio function favors activities with larger profits, and hence, larger visiting times. As a result, fewer activities participate in the final solution and therefore, each new solution is computed using fewer insertions, which in turn take less time. Pay attention to $t11$ for 1 route, where the activities are, on average, only 60% that of the solutions with $profit$ in the nominator of the ratio function.
For the Montemanni dataset, results are quite similar in terms of both time and profit. We can see that there are a few spikes in time, but they can be attributed to statistical error rather than actual information. In the last dataset, the same observations as in the previous dataset can be made. As this work focuses mainly on producing quality solutions with a lot of minimums constraints, it is required to include more activities in the final solution. Therefore, the rest of this work will focus on results utilizing the $profit$ ratio.

4.2 Random factor

The second parameter in need of a value is that of randomness. Specifically, the range the random values will have. As we mentioned above, after calculating the ratio for each Activity, we will multiply it with a random value drawn from a uniform distribution. Let the range be $[low, high], low, high \in R^+$. Notice that because we will use this value to multiply the ratio, this

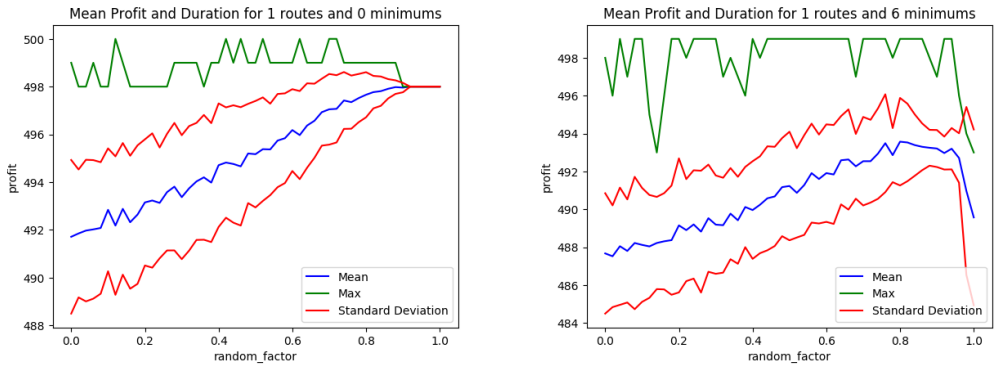


Figure 4.1: Profit vs Random factor for 1 route

is equivalent to any other range $[a * low, a * high], a \in R^+$. Therefore, without loss of generality, we can limit our range in $[low, 1], low \in [0, 1]$. As a result, we only need one value, low instead of two. When $low = 1$, the randomized algorithm degenerates to the corresponding deterministic one. Barring a formal analysis, we will resort to extensive testing on current topologies to extract results. To that purpose the first topology of Gavalas was run for 1 to 3 routes starting from the middle of Athens (where Activities are the most dense) and with minimums [0, 2, 4, 6] of category 1. Start time was 600 and the end time was 1020 for all routes. Below, the plots show average, max and standard deviation of the profit, as well as the mean of the time it took to on average to compute the solution. Each setting was run 100 times to eliminate the stochastic nature of the produced solutions. We include only the most interesting of the figures here in 4.1 and 4.2. The full results can be found in figures B.7–B.13.

The reason we decided to include these three figures is that they display three different behaviors. In the first figure, we see the mean profit to steadily rise until $low = 0.9$ from which point it remains steady. Alongside that increase, we see a steady decline in the standard deviation, until all solutions are the same (std = 0). This behaviour argues towards a larger low value, or even for a deterministic algorithm. The second figure shows a very steep decline in the mean profit after $low = 0.95$. In this setting we can see the advantages of introducing a little randomness. It is a lot more difficult to get stuck in a local optimum when there is more randomness. This figure illustrates perfectly why we differentiated ourselves from the basis of our work and do not always pick the Activity with the maximum $Ratio$. This mechanism allows us to explore more varied solutions and not being stuck in recreating the same solution again and again. Lastly, the final figure shows us what happens when the search space is not exploited enough. In that case, randomness doesn't offer better solutions, but instead makes us explore in less promising areas, jeopardizing our ability to obtain an optimal solution.

The takeaways from the three figures are the following.

1. The larger the random factor is, the less variance exists in the solutions. The extreme for this takeaway can be seen in the first figure, where for random factor equal to 1, all of the produced solutions are the same.

2. The larger the search space, the more variance there is in the solutions. Also, for large search spaces, the max follows the mean. Randomness stems from the random factor and the random shake.

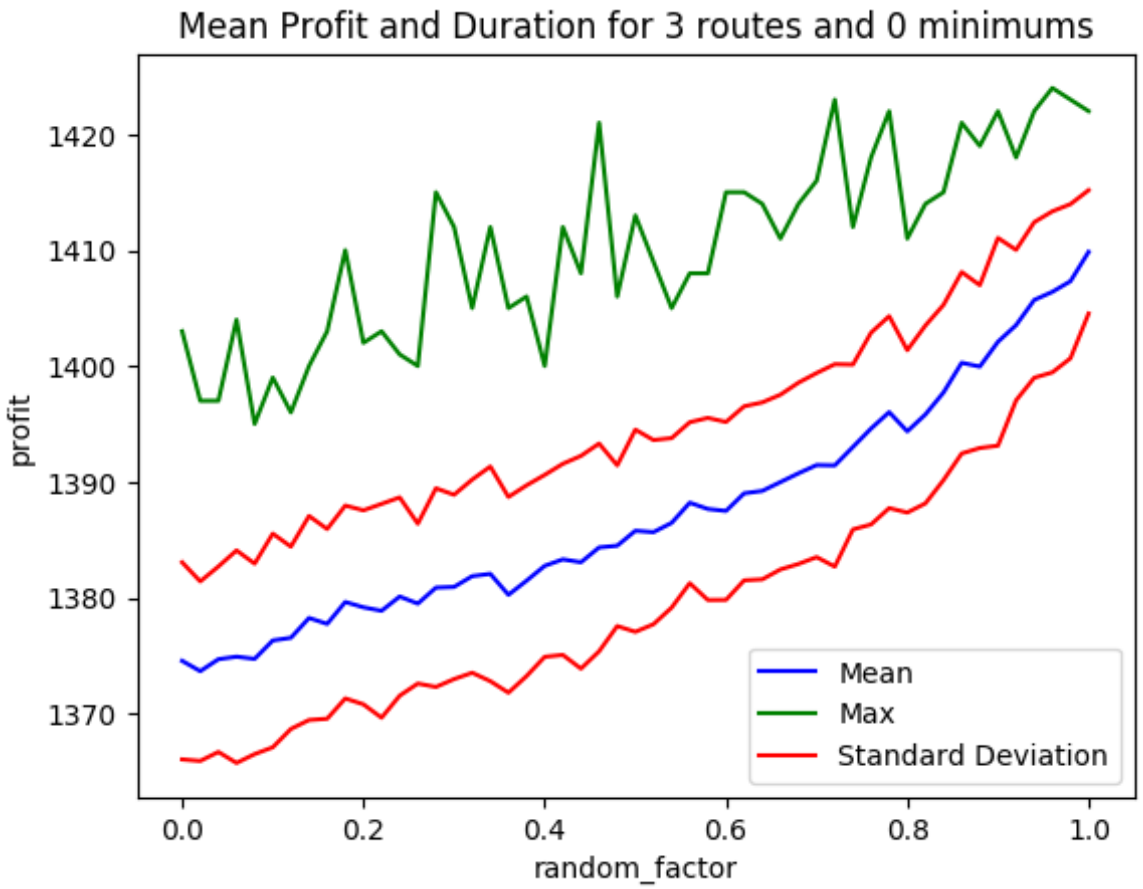


Figure 4.2: Profit vs Random factor for 3 routes

3. The larger the random factor, the more likely it is for a high quality solution to be produced, but it is some times less likely to produce the best possible solution.

From these figures, as well from the rest in Appendix B, it is clear that for small cases, more randomness should be introduced after few iterations, while for larger search spaces, if there is not sufficient time, the deterministic solutions should only be explored instead. In future work we would like to see the randomness not stay the same during the whole search, but instead start from producing deterministic solutions and as we cycle back to the same solutions indicating that the exploitation phase is mostly completed, start introducing more randomness to explore greater areas of the search space.

4.3 Using 2-OPT moves

In their seminal paper Vansteenwegen et al. (2009b), the authors chose not to perform 2-OPT moves, due to the small overlap of the Time Windows. However, in the more realistic dataset of Gavalas et al. (2015) there is considerable overlap between the Time Windows of the various activities and therefore it was deemed beneficial to include 2-opt moves in our system. Because after each successful 2-OPT move, more inserts need to be performed, each iteration takes longer. On the other hand, when performing 2-OPT, the algorithm should produce better results. Initially, we tried performing 2-opt moves after we had inserted all the Activities we could and then try inserting again as many

Activities as possible. The results of that approach are shown in figures B.14-B.19 as a fraction of the respective average profit achieved without 2-OPT moves.
Although using 2-OPT is systematically better, it doesn't improve the output greatly. The Montemanni dataset benefits most from that modification, but even so, the average improvement is under 1%. Performing the insert step after 2-OPT, takes a lot of time, as all Activities need to be revisited. That takes a lot of time and therefore this approach is a lot slower.
Before dismissing this technique though, we also tried replacing Activities instead of inserting new ones. The relative results can be seen in figures B.20-B.25 On average, using 2-opt produces slightly better solutions. However, this technique is better utilized in some datasets and less so in others. In the context of ILS, performing 2-OPT is quite expensive in terms of computation time and therefore not easily and efficiently applied. As part of future work, a more sporadic and guided use of 2-OPT could be employed, which will yield comparable or even better results, without demanding as much computational power.

4.4 The choice of w

Finally, there is the problem of the value of w in our $Ratio$ function. Having a very low value will lead to few feasible solutions and therefore, most of the time will be spent computing solutions which cannot be utilised. On the other hand, a very large value will have adversarial effects on the profit factor and hence generate solutions of lower quality. In truth, depending on the time there is, the location of the current route, the minimums we have, and the topology, there is a different optimal value of w. However, since that value is highly volatile, there is little hope of determining those optimal values for all instances *a priori*. In this section we will first demonstrate that the count of each category in a solution are samples from a discrete normal distribution. Then we will show how this distribution varies for different weights. Finally, we will present a couple learning algorithms based on Stochastic Gradient Descent (SGD).
Starting off, we used a single value for all topologies and categories. In figure 4.3 we have plotted the count of Activities belonging to category '1' for different minimums and different constant weights. For each weight we produce about 2000 different solutions.
Looking at the figure, it is clear that when little or no constraints apply, the number of Activities present in the solution with category '1' follows a normal distribution. The same holds true for low values of weight for more constrained settings. In more constrained settings, the normal distribution is still present below the appointed threshold, but gets more concentrated towards the threshold as the weight increases. This can be seen in the fading blue line when we require a minimum of six Activities of category '1' and the weight is between 2 and 4. This hypothesis is further reinforced when we look at larger instances with a lot of constraints.
As the minimums, number and durations of routes change, so does the optimal value for the constant weight. In order to see how this changes, we ran the algorithm for different weights and different minimums to see how the profit is affected. In order to derandomize results we ran each experiment 100 times. Of all those results, we have chosen to include here the most striking of those that will help us get an intuition as we formulate our next step.
As we can see from the first image, when the minimum number of Activities of category '1' is less than what would normally be present in a random solution, weight has no impact whatsoever. On the

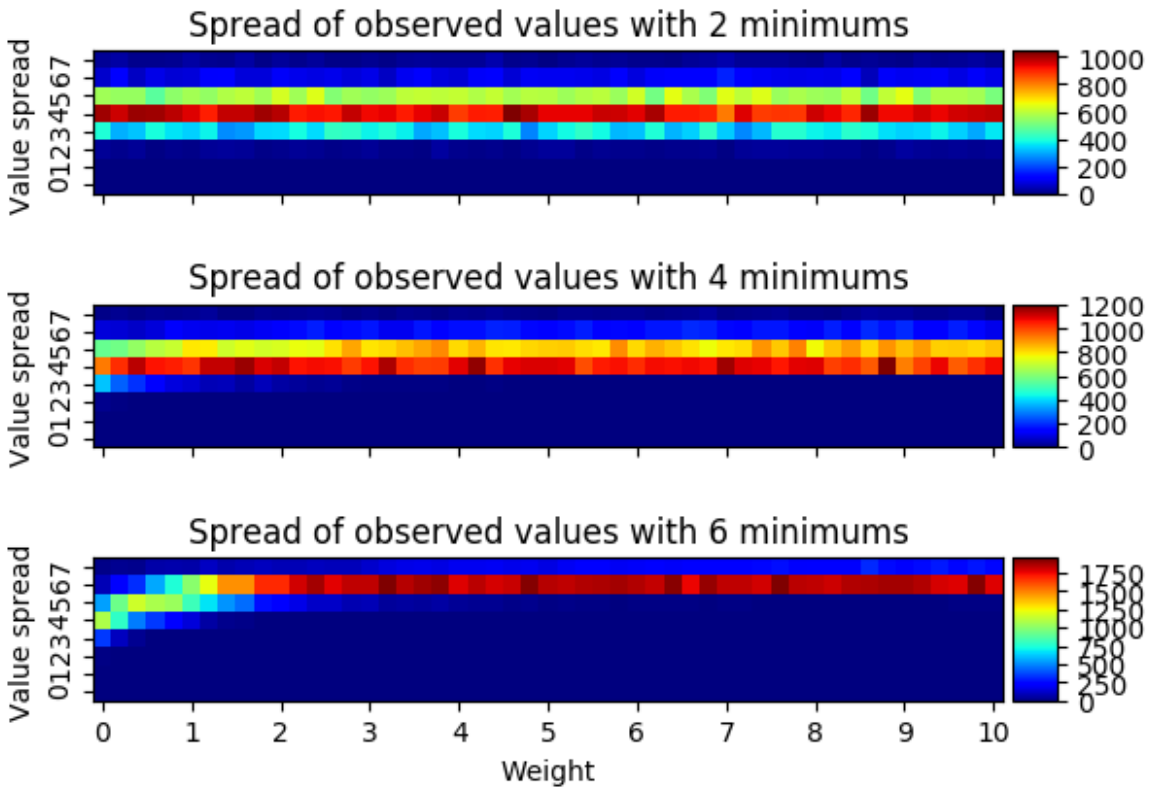


Figure 4.3: Count of Activities in each solution with category '1' respective to weight and minimums for one route

other hand, in the case there would normally be fewer Activities with that category, (in this setting, it is about 12), a non zero value is needed to achieve better results. The shape of the distribution of the profits produced relative to the weight can be seen from $minimums = 14$ and later. First, there is a range where the algorithm fails to find any feasible solutions and therefore the maximum profit is zero. Then, as the weight increases, so does the probability of finding a feasible solution. This can be seen in the jump around $weight = 0.5$ for $minimums = 14$. The probability of finding a valid solution increases until all runs produce a feasible solution, at which point the minimum profit jumps at about the same value as the maximum. Close to that, the maximum average profit is also observed. After that, the average profit slowly declines, as the heuristic is now more concerned about inserting Activities of that category, rather than achieving a high profit. In essence, the calculated ratio is skewed away from selecting the best profit over shift and more to selecting Activities with needed categories.

Selecting a static weight works for some topologies, but most likely every selection will be far from the optimal value. Therefore, in the next section, we will attempt to find that optimal value at the same time as producing new candidate solutions. To achieve this, we turn to learning.

In order to use learning, the first thing that needs to be done is devising an objective function to optimize. The weight value we are looking for is certainly above the threshold where no valid solutions are found. Next, the value that we are looking for needs to be above the threshold where a run might not return a solution. Admittedly, a bad solution is better than no solution at all. Lastly, we would like to arrive to a value not far above the previous threshold, as large values will result

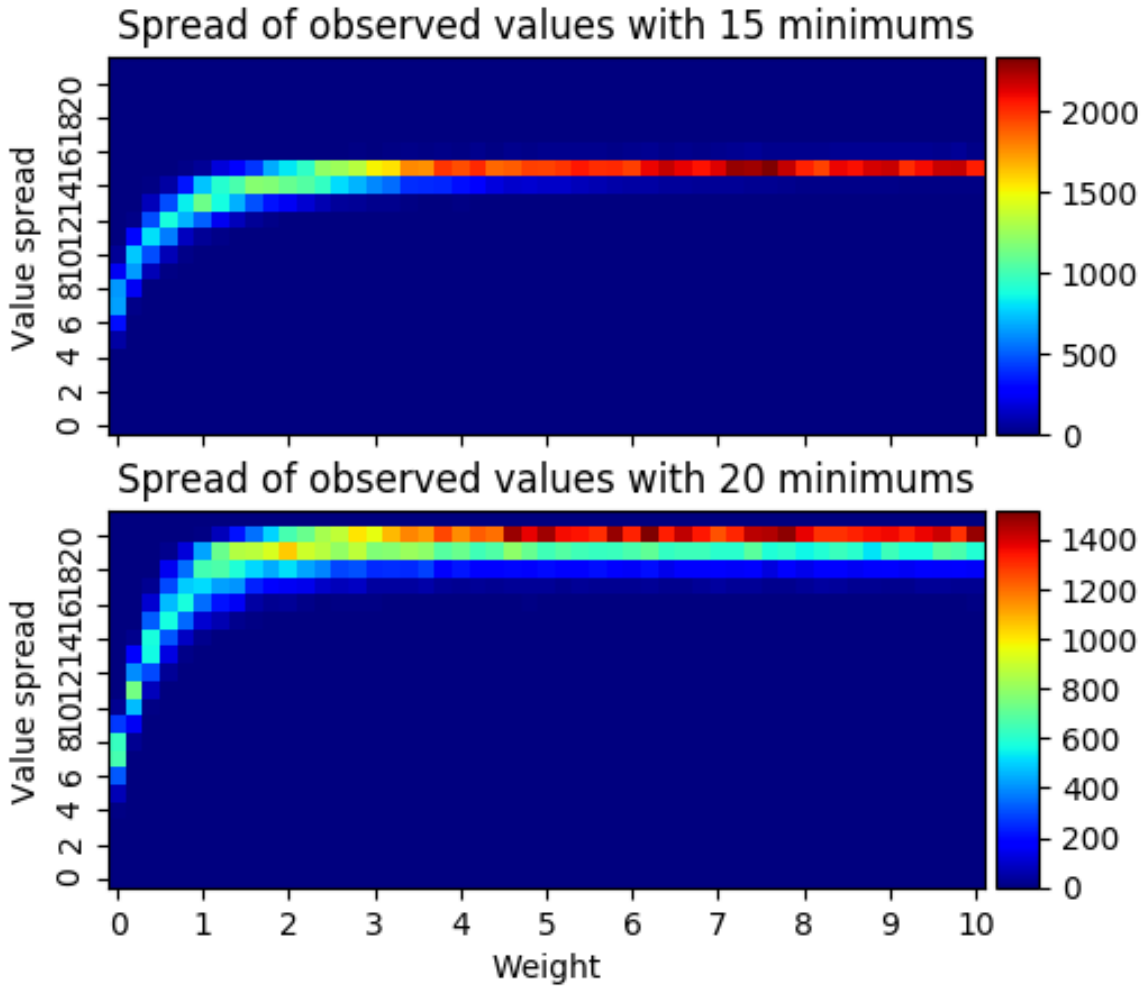


Figure 4.4: Count of Activities with category '1' respective to weight and minimums for three route

to lower profits. From the aforementioned description we see that in reality, we are searching for $\min weight : P(solutionisvalid|weight) = 1$. In the beginning of this subsection, we showed that for a given weight, the probability of how many Activities will be contained in a random solution, followed a normal distribution. As a result, what we are really looking for, is the smallest weight that will ensure that there will be many valid solutions to choose from. A fairly good approximation is using a linear function. Therefore the function we use is

$$P(w) = \begin{cases} 0 & w < \frac{b}{a} \\ aw - b & \frac{b}{a} \leq w \leq \frac{b+1}{a} \\ 1 & \frac{b+1}{a} < w \end{cases}$$

Having come up with a scoring function, now we need to choose a learning algorithm. Given that we have a way to quickly come up with random solutions given a weight, using ILS, utilizing the well known technique of Stochastic Gradient Descent is a good choice.

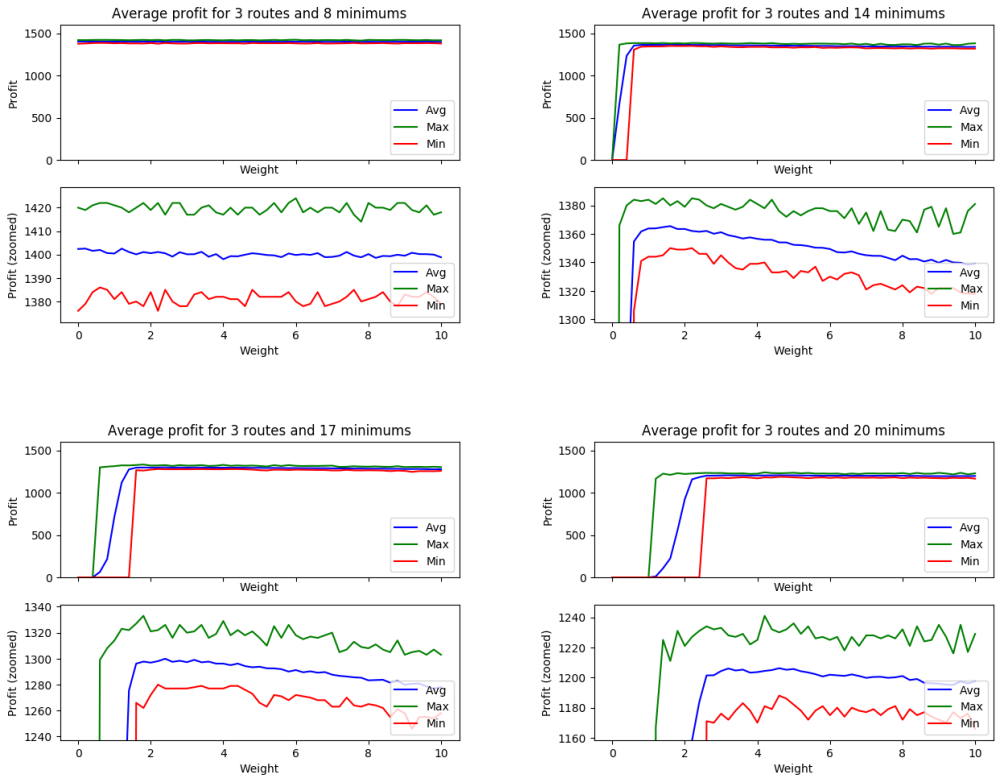


Figure 4.5: Profits relative to single weight for 8,14,17 and 20 minimums of category '1' for 3 routes

At first, we try a vanilla SGD algorithm, which given a new random solution with demands D produced with weight W updates as follows:

$$W_{new} = W + D * step \tag{4.1}$$

The variable $step$ is of the form $step = \exp(-\frac{\ln(0.05) * iteration}{estimated\ total\ iterations})$. With that update strategy in mind, it is clear that, in principle, W should converge to a value where almost all solutions are valid. Unfortunately, because W is taken into consideration only when the minimum for this category has not been met, the count of Activities with that category will seldom be over the set threshold. As a result, W converges to a much higher value than the desired one. The histograms of the final W for the instances with 3 routes with a demand of 13–20 minimums are shown below to provide an easy overlook of how much this function overshoots.

We can see that the converged value is so great that in the end it completely dominates any other component of the $Ratio$ function. As a result, achieving a good profit is no longer the priority of that function. Instead it just focuses on including required Activities without taking into consideration the profit. Instead we could just admit Activities only from the required categories. The previous function tends to converge to large values, which is detrimental to the final result. To avoid this, we will use regularization by modifying the previous function as follows.

$$W_{new} = W + (D - \lambda W) * step \tag{4.2}$$

This modification limits the value of W and helps get closer to the desired value. For 3 routes and demand of category 1 Activities in the range 13–20 we get the following distributions.

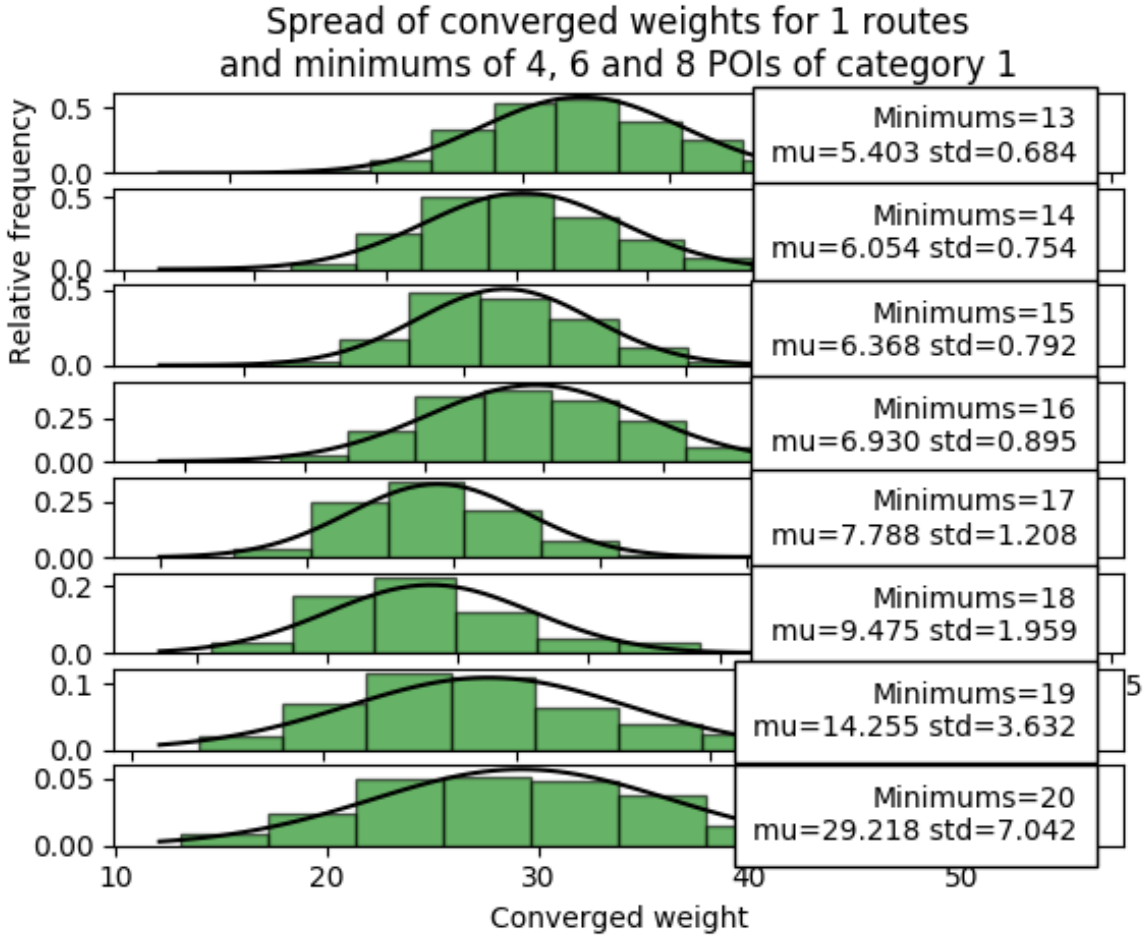


Figure 4.6: Spread of converged weights for different minimums

An easy takeaway from figure 4.7 is that adding regularization greatly helps towards limiting extreme values of W especially in demanding instances, where ensuring that all generated solutions are valid is impossible. In essence, regularization is a way to discount the probability of the generated solutions being valid. As W increases, so does our discount, leading to a lower fixed point. Another point of note is how regularization impacts the deviation of the final weights. As can easily be seen, without regularization there is a great spread of the produced values, ranging wildly around the average. On the other hand, using regularization, the standard deviation is significantly lower and, as expected, is larger for more limiting instances.

Finally, let's plot the takeaways from figure 4.5 and combine them with the above analysis to see how good our approach is.

From the above figures, we can see that with great confidence the algorithm will end up learning a very good value for W and therefore there is a high probability that many valid and highly competitive solutions will be produced, improving the overall quality of the final solution returned.

Finally, we tried adding momentum to our update function. The formula that was used is:

$$V_i = V_{i-1} * 0.7 + (D - 0.15 * W) \tag{4.3}$$

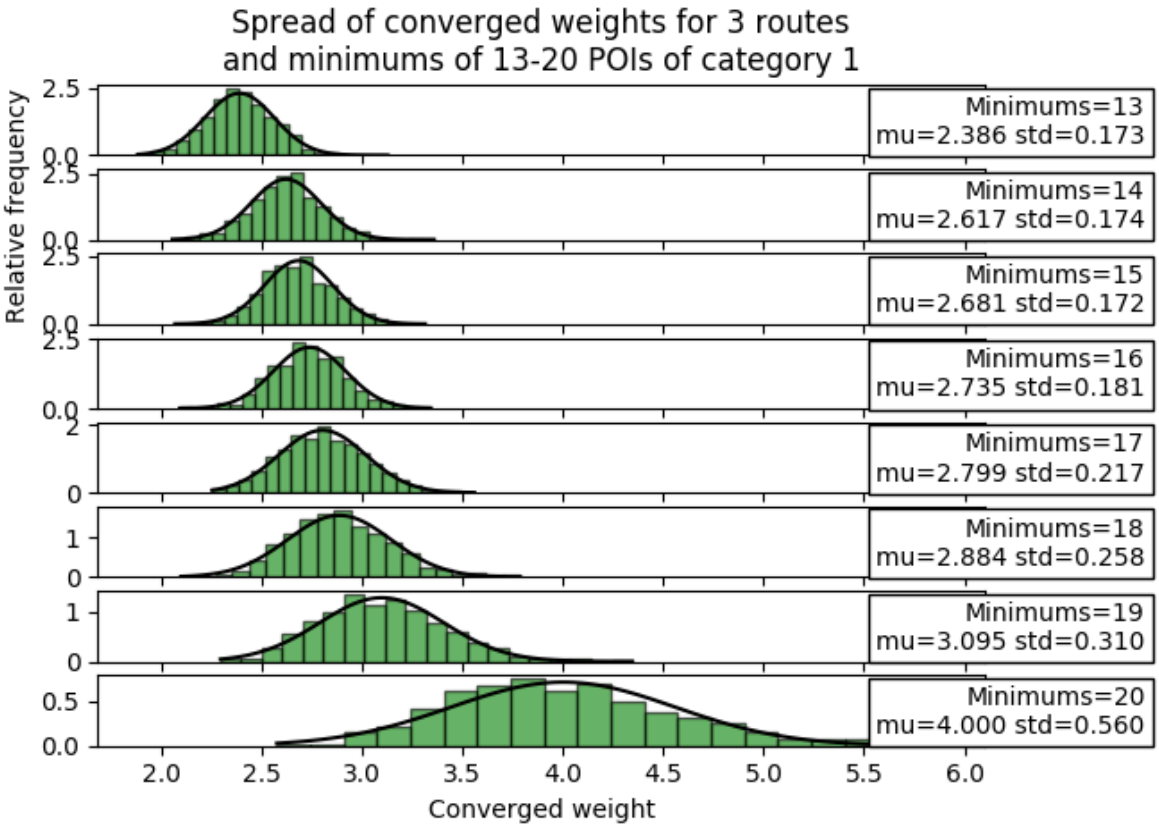


Figure 4.7: Spread of converged weights for different minimums with regularization

Minimums	No regularization		Regularization		With momentum	
	avg	std	avg	std	avg	std
13	5.403	0.684	2.386	0.173	2.408	0.264
14	6.054	0.754	2.617	0.174	2.627	0.271
15	6.368	0.792	2.681	0.172	2.700	0.245
16	6.930	0.895	2.735	0.181	2.732	0.283
17	7.788	1.208	2.799	0.217	2.808	0.292
18	9.475	1.959	2.884	0.258	2.924	0.363
19	14.255	3.632	3.095	0.310	3.122	0.453
20	29.218	7.042	4.000	0.560	4.040	0.748

Table 4.1: Comparison of update function in terms of avg and std for different minimums

$$W_i = W_{i-1} + step_i * V_i \tag{4.4}$$

Unfortunately, the result of this modification was the same normal distribution, but with a greater variance. The results of all 3 update functions are shown in 4.1. The first function without normalization greatly overshoots the sought value. By adding normalization, we get very close to the value we would like, ensuring that a good solution will be found. Lastly, trying momentum, we end up with good solutions, but the solutions are more spread out. For that reason, we choose to go with the second update function.

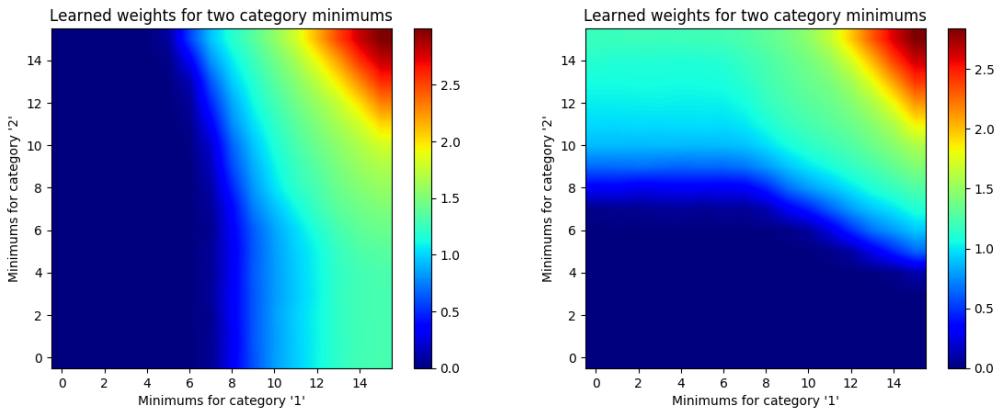


Figure 4.8: Inter-dependency of learned weight relative to minimums for two categories

Finally, we will show that the learned weights depend not only on the minimums for this category, but also from the minimums for other categories. This inter-category dependency can be seen in a 2D map in 4.8

The log function has been applied to the weights so that we can see a smoother transition and not just see a part which is red and the rest be blue. We can see that the learned weight for each category is a function of the minimums for all categories and not just of the one they represent. As a result, we see that the optimal solution is dependent not only on the minimums of each individual category, but as a function of the entire vector of m. Luckily, our approach is robust enough to be able to cope with that complexity.

Chapter 5

Comparisons

5.1 Results with previously Best Known

In this chapter we will use the parameters as they have been chosen in the previous chapter to produce results in order to compare with similar solutions. Our results were run on an Intel(R) Xeon(R) CPU E5−2630 v3 @ 2.40GHz with 62 GB of RAM, a big step up from the hardware used for results in previous papers. Solely for that reason we will not compare solution times in our analysis. For starters, we will compare ourselves with the ILS algorithm by Vansteenwegen et al. (2009b). Our algorithm found overwhelmingly better results, outperforming its predecessor in 217 out of 304 instances, producing the same result in 61 and underperforming in only 26 instances. More specifically, in the Solomon dataset (c*_100, r*_100, rc*_100) only 1−2 instances underperformed, while on the extended instances by Montemanni and Gambardella (c*_200, r*_200, rc*_200) based on Solomon, we noticed the majority of underperformances with 7 instances underperforming for 1 route. More importantly, for 3 routes there were 14 instances that produced the same results and all 27 instances produced the same, optimal results for 4 routes. In the Cordeau dataset (pr1−10) there is an average of 7.5 instances that are better with the rest being worse. Similarly, for the extended Cordeau dataset (pr11−20) there are on average 9.25/10 better instances. On average, our solutions are better by 1.43%. The minimum gap that was observed was over 10% in one of the Solomon instances. Conversely, the maximum gap was a little over 4% in one of the extended Cordeau instances. The biggest average gap in favor of our algorithm was also observed in the extended Cordeau dataset being 2.55%. The maximum average gap was in the Montemanni and Gambardella dataset which was based on the Solomon dataset. However that average was still negative, concluding that our algorithm was on average 0.67% better. We should note that this is also the dataset where all instances were the same for 4 routes, bringing down the average as all nodes were included in both solutions. It is natural that for fewer routes the gap values swing more wildly to more extreme values, as the absolute profit is smaller and therefore a mild change of let's say 10 points can mean an aggressive swing of more than 5%. For more routes where the absolute profit is maybe around 600, a solution that has a profit of 610 is only 1.64% better. We can see this tendency through the averages for each number of routes. Same as in Vansteenwegen et al. (2009b) we notice that on the one hand small instances like the original Solomon dataset with its small search space are easier to find better solutions, on the other hand, larger datasets such as the Montemanni have been studied less and also present themselves to find better results.

Another reason we do not find as good results for more routes is the size of the search space. While for more routes we necessarily spend more computing power, the size of the search space and the

possible solutions explodes exponentially, which means that the quality of our considerations continues to deteriorate further for each new route we need to consider. In the future this could be rectified by deciding dynamically when to stop searching for a better solution.
In short, our algorithm compares very favorably with previous attempts at solving this problem. We now present the results in tables A.14–A.17. Each table contains results for 1–4 routes. The first column contains the name of the topology. The next three depict the ILS result, the result we produced and the percent difference between them. A negative percentage means our algorithm is better. A gap of 0.00 means that the two solutions are equal. As we can see, our algorithm matches previous results and compares favorably with previous solutions.
In order not to bore the reader with long tables, those tables have been moved to appendix A and here we only include a summary for each dataset counting in how many instances we produced a better, worse or the same result and how much the minimum, maximum and average gap are.

Table 5.1: Aggregate results in the unconstrained setting

Name	routes	Worse	Same	Better	Max	Min	Average
righiniTOPTW2	1	1	8	20	0.68	-10.41	-2.10
	2	1	8	20	0.95	-6.36	-1.52
	3	2	0	27	0.14	-5.05	-1.62
	4	0	1	28	0.00	-4.08	-1.83
MontemanniTOPTW1	1	7	0	20	3.65	-3.57	-1.03
	2	4	1	22	2.34	-3.57	-0.98
	3	0	14	13	0.00	-2.86	-0.68
	4	0	27	0	0.00	0.00	0.00
righiniTOPTW3	1	1	2	7	1.91	-4.77	-1.71
	2	2	0	8	2.71	-5.10	-1.52
	3	4	0	6	1.29	-3.58	-1.15
	4	1	0	9	0.56	-2.84	-1.09
MontemanniTOPTW2	1	0	0	10	-0.60	-9.75	-5.15
	2	2	0	8	4.19	-7.66	-1.65
	3	0	0	10	-0.25	-5.76	-2.32
	4	1	0	9	2.70	-3.99	-1.08
General	1	9	10	57	3.65	-10.41	-2.50
	2	9	9	58	4.19	-7.66	-1.42
	3	6	14	56	1.29	-5.76	-1.44
	4	2	28	46	2.70	-4.08	-1.00
Entire Dataset	0	26	61	217	4.19	-10.41	-1.43

5.2 Increasingly difficult topologies

In this section we will present how our algorithm behaves as we ask for increasingly more minimums. We can see a steady decrease in profit as we move to more demanding scenarios. In scenarios where no solutions were returned, we have substituted with NaN values. In the rest of the cells, we present the average of the returned valid solutions. This means that while in some of the more exacting cases, not all runs produced a result, in order not to skew those results toward very small values, the result for each setting will be the average of only the non zero solutions. In this section we have also included the dataset used in Gavalas et al. (2015). In general, we can see that as we introduce more minimums, the returned result stays the same or diminishes as expected.

However, in 34 out of the total 96 instances one of the constrained instances produced results by at least one point on average than the unconstrained. Although this may seem strange, we can see that it is not. By introducing minimums, we change the priorities of the algorithm. In some cases, this prioritization leads to better results, while in others to worse. However, only in 2 topologies one of the constrained instances is better by 5 whole points on average.

In most cases, we see that in the unconstrained setting and the slightly constrained where we require only two Activities of the first category the average profit stays the same. On the other hand, as we require more and more Activities of that category we can see sudden falls in the achieved results. Some of these falls are so visible that indicate clearly that the produced solution would not be considered if we hadn't included the minimums. From results like these, it is clear that our algorithm works and produces the expected results.

Name	min=0	min=2	min=4	min=6	min=8
c101	320.00	319.90	309.50	290.00	251.82
c102	360.00	360.00	360.00	360.00	349.90
c103	394.20	394.40	387.00	371.04	NaN
c104	416.50	416.40	411.00	393.98	363.64
c105	340.00	340.00	330.00	330.00	275.62
c106	340.00	340.00	340.00	319.50	263.64
c107	365.60	365.40	368.20	370.00	344.20
c108	378.20	361.50	345.30	327.80	288.32
c109	380.00	380.00	380.00	377.50	358.31
c201	861.40	861.20	860.90	858.80	851.70
c202	908.40	908.20	911.40	910.50	912.10
c203	938.00	937.30	937.70	938.20	937.90
c204	963.70	962.90	963.10	963.70	963.70
c205	902.20	902.10	902.50	901.40	900.40
c206	918.90	919.30	918.10	919.10	919.10
c207	918.40	918.90	918.80	918.80	917.10
c208	932.30	933.50	933.40	934.40	933.00

Continued on next page

Name	min=0	min=2	min=4	min=6	min=8
pr01	299.06	298.49	296.96	292.79	251.85
pr02	381.39	381.90	381.78	382.70	382.76
pr03	378.11	376.78	378.56	379.61	378.99
pr04	450.86	452.05	451.20	450.22	451.54
pr05	550.46	550.91	549.91	549.49	553.08
pr06	496.73	493.60	495.13	495.05	493.77
pr07	280.66	282.48	281.40	281.90	258.44
pr08	445.33	444.82	443.89	447.24	445.78
pr09	443.47	446.01	447.40	438.68	428.81
pr10	528.14	529.79	528.72	528.03	528.67
pr11	324.91	325.47	325.18	324.78	318.54
pr12	424.65	423.83	424.36	425.31	424.59
pr13	440.30	440.91	442.33	441.01	434.81
pr14	499.54	502.32	499.17	497.40	499.49
pr15	635.68	635.03	639.20	636.50	639.07
pr16	553.15	555.12	553.86	553.92	558.12
pr17	346.21	346.09	345.77	346.16	345.18
pr18	449.06	448.66	447.80	448.59	438.57
pr19	461.78	460.65	456.71	455.91	455.68
pr20	583.63	580.95	580.94	576.28	556.41
r101	197.76	197.71	197.11	177.94	NaN
r102	279.63	280.35	280.93	274.47	269.73
r103	289.41	289.72	291.56	292.20	277.44
r104	300.32	300.56	300.81	281.83	264.59
r105	241.67	242.29	240.98	235.57	201.58
r106	285.98	286.15	281.05	268.46	255.68
r107	298.96	298.96	293.81	248.67	190.83
r108	304.62	303.79	294.07	285.51	269.08
r109	273.85	273.83	272.08	263.82	212.24
r110	280.78	280.64	268.04	249.39	235.16
r111	299.80	299.52	299.30	298.02	282.88
r112	293.17	293.43	287.58	276.09	230.00
r201	753.30	753.83	753.02	752.91	752.06
r202	874.62	874.34	876.15	873.75	874.60
r203	966.08	965.44	965.21	965.04	967.14
r204	1054.99	1056.17	1055.57	1055.41	1056.35
r205	867.50	867.09	867.02	866.34	869.27
r206	949.96	947.93	946.97	948.17	950.99

Continued on next page

Name	min=0	min=2	min=4	min=6	min=8
r207	1014.41	1017.21	1015.54	1013.73	1015.19
r208	1080.13	1081.40	1080.64	1080.52	1081.65
r209	888.76	887.20	887.75	889.22	887.53
r210	927.97	928.92	927.80	926.39	925.58
r211	990.80	995.14	991.60	991.29	988.82
rc101	218.03	218.09	216.62	NaN	NaN
rc102	249.96	247.89	246.69	220.43	NaN
rc103	254.59	255.89	258.77	244.17	195.22
rc104	277.47	273.53	275.99	273.30	265.29
rc105	237.96	238.48	237.98	235.54	215.00
rc106	242.13	239.11	234.62	219.24	NaN
rc107	269.62	269.34	269.48	256.48	239.88
rc108	283.76	291.24	277.47	265.54	NaN
rc201	742.76	743.34	743.43	749.08	749.95
rc202	876.57	874.85	873.19	873.61	877.47
rc203	950.29	951.41	948.70	950.52	950.85
rc204	1081.99	1080.82	1082.56	1081.18	1083.57
rc205	817.88	818.86	816.92	817.95	817.73
rc206	836.36	837.68	836.45	834.91	836.77
rc207	902.01	905.89	899.69	903.30	901.12
rc208	968.82	967.31	968.25	970.23	968.28
t1	512.89	512.89	504.67	491.00	455.06
t2	499.82	499.47	498.80	494.00	472.44
t3	532.00	532.00	531.96	516.53	478.19
t4	533.00	533.00	533.00	523.98	478.35
t5	495.38	495.36	491.01	477.63	463.00
t6	504.23	504.49	504.97	498.90	496.04
t7	535.00	535.00	533.50	516.44	502.00
t8	567.17	569.00	554.51	518.19	474.00
t9	534.66	534.96	526.65	506.88	469.98
t10	579.95	569.04	566.93	529.98	497.00
t11	525.45	525.84	525.61	524.20	506.53
t12	524.09	524.06	524.04	524.94	512.77
t13	529.00	528.00	513.00	474.17	416.62
t14	552.00	552.00	535.00	497.36	465.00
t15	520.65	517.94	494.50	452.84	NaN
t16	504.70	504.61	504.56	499.86	482.18
t17	537.00	537.00	529.64	504.96	474.50

Continued on next page

Name	min=0	min=2	min=4	min=6	min=8
t18	476.88	470.66	457.34	436.18	NaN
t19	531.70	531.89	527.60	516.78	NaN
t20	520.91	520.98	521.00	464.50	NaN

In general, the experimental results we obtained were very close to the ones we were expecting from the corresponding theoretical predictions. Furthermore, our algorithm has compared favourably with the state of the art on the known datasets. In short, our algorithm is a valuable addition to the bibliography and the ideas and analysis that have been used for its formulation can form the basis for future work in the field.

Finally, every algorithm needs a name. We have chosen the term Learning Randomized Weighted Iterated Local Search or LRWILS for short. The randomized part refers to the random factor that we have used to escape local minimums in the small instances and the weighted part refers to the new term we have added to the ILS algorithm in order to prioritize the correct Activities. Finally, while solving the instance, LRWILS finds the best possible weights for this instance.

Chapter 6

Conclusion

6.1 Synopsis

In this diploma thesis, we introduced a new variant of the Orienteering Problem inspired from the Tourist Trip Design Problem. Our motivation for formulating this new variant is making the setting of the TTDP more real and more practical with an eye towards real life application that can help actual tourists make a choice on how to structure their visits in a new town that they will be visiting for just a few days. Starting from the algorithm in Vansteenwegen et al. (2009b) we extended it with a new part in order to skew the algorithms insert choice towards including more Activities with the needed categories. On that same algorithm we have explored augmentations like including randomness, 2-OPT moves and the effects of the authors choice to use $profit^2/shift$ and how these augmentations affect the final result of the algorithm. We have concluded that adding randomness helps with exploring the search space, while the deterministic algorithm helps in exploiting good solutions. For small instances where we spend enough computing power, exploration helps us escape local optima. On the other hand, in large instances where there is not enough time to properly explore the search space, focusing on exploitation yields better results. Because of the time windows using 2-OPT moves is very difficult and although it produces better results, it needs a lot of time to be properly executed and therefore it is not cost efficient. Regarding the ratio function, $profit^2$ produces comparable results but usually includes fewer Activities in the final solution. In order to achieve the best possible solution we had to find an appropriate weight. We turned to learning for that and designed a Stochastic Gradient Descent algorithm to help us find that optimum value. We saw how normalization helped bound that value closer to the desired range and presented competitive results with other algorithms in the unconstrained setting and results in the respective constrained one.

6.2 Future work

There are many avenues to explore future work. In direct relevance to this problem is examining the case where each Activity can have more than one categories. In this work each Activity had only category. But in reality, an Activity can fall under many categories. Having many categories changes substantially the way that this problem is solved. One question is how the different weights should be combined with these categories.

Another avenue to explore is how should randomness be utilised. The algorithm could start being deterministic for high exploitation and over time relax to achieve better exploration. The schedule of this relaxation and how it relates to the size of the search space is another interesting question.
In this diploma thesis we have concerned ourselves with cold solving a single instance. However, solving multiple instances in the same topology could give us new ways to approach the problem. For instance, instead of using the SGD we could train a model to provide us with a a-priori weight to use taking into account all the input variables. Furthermore, solving many instances in parallel could pose its own challenges. Should instances be batched together, or having access to static information and each one being solved in a different CPU is the most efficient approach we can have?
As Moore's law has been ticking away for 35 years since the conception of the Orienteering Problem and as better and better heuristics have been devised to solve this problem, we have acquired the ability to solve instances for ever greater topologies. However all modern datasets are still around 100 nodes, a far cry from the real life datasets where there are a hundred Activities even in a small city and thousands in the big metropolises around the globe. Being able to scale these solutions to the thousands and millions is crucial for the ability to transform this from a theoretical paper to an engineered solution which actually helps people make more informed decisions.
Looking through the bibliography on this problem one notices that all algorithms perform a cold start. This means that when they start to solve a new instance of the problem they don't utilise any prior knowledge and experience. For most common algorithms, the different possible inputs that can be received are so many that no memoization is possible. However, utilizing Machine Learning one could suppose that after solving billions of different instances for a city a model could be created that would be able to efficiently solve any of these instances better and faster than any of the common algorithms. Even if someone doesn't resolve to Machine Learning, solving in parallel many instances on the same topology would be an interesting problem both in terms of engineering but also in terms of computer science.

Appendices

Appendix A

Tables

Table A.1: Benchmark OP and TOP instances

Reference	Problem	Number of test instances	Number of vertices (N)	Number of paths m
Tsiligirides (1984)	OP	18	32	1
		11	21	1
		20	33	1
Chao et al. (1993) and	OP	26	66	1
Chao et al. (1996b)		14	64	1
Fischetti et al. (1998)	OP	3*15	21-262	1
		3*44	47-400	1
		4*11	25-500	1
		5*15	21-301	1
Chao et al. (1996a)	TOP	3*18	32	2 to 4
		3*11	21	2 to 4
		3*20	33	2 to 4
		3*20	100	2 to 4
		3*26	66	2 to 4
		3*14	64	2 to 4
		3*20	102	2 to 4

Table A.14: Results comparison between the ILS and our own algorithm for m=1 route

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
c101	320	320	0.00	c201	840	**870**	-3.57
c102	360	360	0.00	c202	910	**930**	-2.20
c103	390	**400**	-2.56	c203	940	**960**	-2.13
c104	400	**420**	-5.00	c204	950	**970**	-2.11
c105	340	340	0.00	c205	900	**910**	-1.11
c106	340	340	0.00	c206	910	**930**	-2.20
c107	360	**370**	-2.78	c207	910	**930**	-2.20
c108	370	**380**	-2.70	c208	930	**950**	-2.15

Continued on next page

Table A.14: Results comparison between the ILS and our own algorithm for m=1 route

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
c109	380	380	0.00				
r101	182	**198**	-8.79	r201	788	777	1.40
r102	286	**289**	-1.05	r202	880	**903**	-2.61
r103	286	**291**	-1.75	r203	980	**988**	-0.82
r104	297	**303**	-2.02	r204	1073	**1078**	-0.47
r105	247	247	0.00	r205	931	897	3.65
r106	293	291	0.68	r206	996	989	0.70
r107	288	**299**	-3.82	r207	1038	**1040**	-0.19
r108	297	**306**	-3.03	r208	1069	**1096**	-2.53
r109	276	**277**	-0.36	r209	926	914	1.30
r110	281	**284**	-1.07	r210	958	949	0.94
r111	295	**300**	-1.69	r211	1023	**1032**	-0.88
r112	295	295	0.00				
rc101	219	219	0.00	rc201	780	768	1.54
rc102	259	**266**	-2.70	rc202	882	**910**	-3.17
rc103	265	**266**	-0.38	rc203	960	**992**	-3.33
rc104	297	**301**	-1.35	rc204	1117	**1122**	-0.45
rc105	221	**244**	-10.41	rc205	840	**852**	-1.43
rc106	239	**252**	-5.44	rc206	860	**877**	-1.98
rc107	274	**277**	-1.09	rc207	926	**949**	-2.48
rc108	288	**298**	-3.47	rc208	1037	1031	0.58
pr01	304	304	0.00	pr10	539	**546**	-1.30
pr02	385	**394**	-2.34	pr11	330	**336**	-1.82
pr03	384	**388**	-1.04	pr12	431	**436**	-1.16
pr04	447	**464**	-3.80	pr13	450	**464**	-3.11
pr05	576	565	1.91	pr14	482	**529**	-9.75
pr06	538	**556**	-3.35	pr15	638	**674**	-5.64
pr07	291	**298**	-2.41	pr16	559	**605**	-8.23
pr08	463	463	0.00	pr17	346	**360**	-4.05
pr09	461	**483**	-4.77	pr18	479	**521**	-8.77
pr10	539	**546**	-1.30	pr19	499	**502**	-0.60
pr11	330	**336**	-1.82	pr20	570	**618**	-8.42

Table A.15: Results comparison between the ILS and our own algorithm for m=2 routes

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
c101	590	590	0.00	c201	1400	**1450**	-3.57

Continued on next page

Table A.15: Results comparison between the ILS and our own algorithm for m=2 routes

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
c102	650	650	0.00	c202	1430	**1460**	-2.10
c103	700	**710**	-1.43	c203	1430	**1460**	-2.10
c104	750	**760**	-1.33	c204	1460	**1470**	-0.68
c105	640	640	0.00	c205	1450	**1470**	-1.38
c106	620	620	0.00	c206	1440	**1480**	-2.78
c107	670	670	0.00	c207	1450	**1480**	-2.07
c108	670	**680**	-1.49	c208	1460	**1490**	-2.05
c109	710	**720**	-1.41				
r101	330	**351**	-6.36	r201	1231	1215	1.30
r102	508	**509**	-0.20	r202	1270	**1308**	-2.99
r103	513	**518**	-0.97	r203	1377	**1378**	-0.07
r104	539	**555**	-2.97	r204	1440	**1443**	-0.21
r105	430	**443**	-3.02	r205	1338	**1356**	-1.35
r106	529	524	0.95	r206	1401	**1407**	-0.43
r107	529	529	0.00	r207	1428	**1445**	-1.19
r108	549	**558**	-1.64	r208	1458	1458	0.00
r109	498	**503**	-1.00	r209	1345	**1353**	-0.59
r110	515	**520**	-0.97	r210	1365	**1372**	-0.51
r111	535	**536**	-0.19	r211	1422	**1431**	-0.63
r112	515	**541**	-5.05				
rc101	427	427	0.00	rc201	1305	**1344**	-2.99
rc102	494	**507**	-2.63	rc202	1461	1455	0.41
rc103	519	**523**	-0.77	rc203	1573	1546	1.72
rc104	565	565	0.00	rc204	1656	**1674**	-1.09
rc105	459	**481**	-4.79	rc205	1381	**1398**	-1.23
rc106	458	**482**	-5.24	rc206	1495	1460	2.34
rc107	515	**526**	-2.14	rc207	1531	**1539**	-0.52
rc108	546	**553**	-1.28	rc208	1606	**1632**	-1.62
pr01	471	**495**	-5.10	pr11	542	**548**	-1.11
pr02	660	**680**	-3.03	pr12	727	**738**	-1.51
pr03	714	**723**	-1.26	pr13	757	**815**	-7.66
pr04	863	**885**	-2.55	pr14	925	**937**	-1.30
pr05	1011	**1028**	-1.68	pr15	1126	**1130**	-0.36
pr06	997	970	2.71	pr16	1110	1105	0.45
pr07	552	**556**	-0.72	pr17	624	**628**	-0.64
pr08	796	**806**	-1.26	pr18	877	**905**	-3.19
pr09	867	855	1.38	pr19	955	915	4.19
pr10	1004	**1041**	-3.69	pr20	1056	**1113**	-5.40

Table A.2: Exact algorithms on OP and TOP

Reference	Problem	Algorithm	Benchmark instances	Performance
Laporte and Martello (1990)	OP	Branch and Bound	Uniformly generated graphs of up to 20 vertices	0.8-1.0 of Optimum
Ramesh et al. (1992)	OP	Branch and Bound	Up to 150 vertices	
Leifer and Rosenwein (1994)	OP	Cutting Plane		
Fischetti et al. (1998)	OP	Branch and Cut	Fischetti et al. (1998)	
Gendreau et al. (1998a)	OP	Branch and Cut	Gendreau et al. (1998a)	
Feillet et al. (2005)	OP	Branch and Cut		
Boussier et al. (2007)	TOP	Branch & Price	Up to 100 customers	2 hours for 15 customers per path
Martinelli et al. (2011)	TOP	Branch-cut-and-price algorithm	Chao et al. (1996a)	No improvement on known dataset
Dang et al. (2013a)	TOP	Branch-and-cut algorithm	Chao et al. (1996a)	Improve 29 best known solutions
Keshtkaran et al. (2016)	TOP	Branch-and-price and Branch-and-cut-and-price algorithm	Chao et al. (1996a)	Improve 17 best known solutions

Table A.16: Results comparison between the ILS and our own algorithm for m=3 routes

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
c101	790	**810**	-2.53	c201	1750	**1800**	-2.86
c102	890	**910**	-2.25	c202	1750	**1790**	-2.29
c103	960	**970**	-1.04	c203	1760	**1780**	-1.14
c104	1010	**1020**	-0.99	c204	1780	**1800**	-1.12
c105	840	**870**	-3.57	c205	1770	**1800**	-1.69
c106	840	**870**	-3.57	c206	1770	**1810**	-2.26
c107	900	**910**	-1.11	c207	1810	1810	0.00
c108	900	**910**	-1.11	c208	1810	1810	0.00
c109	950	**970**	-2.11				
r101	481	**483**	-0.42	r201	1408	**1414**	-0.43
r102	685	**687**	-0.29	r202	1443	**1452**	-0.62
r103	720	**732**	-1.67	r203	1458	1458	0.00
r104	765	**784**	-2.48	r204	1458	1458	0.00
r105	609	**614**	-0.82	r205	1458	1458	0.00

Continued on next page

Table A.16: Results comparison between the ILS and our own algorithm for m=3 routes

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
r106	719	**722**	-0.42	r206	1458	1458	0.00
r107	747	**754**	-0.94	r207	1458	1458	0.00
r108	790	**796**	-0.76	r208	1458	1458	0.00
r109	699	698	0.14	r209	1458	1458	0.00
r110	711	**729**	-2.53	r210	1458	1458	0.00
r111	764	**770**	-0.79	r211	1458	1458	0.00
r112	758	757	0.13				
rc101	604	**621**	-2.81	rc201	1625	**1658**	-2.03
rc102	698	**709**	-1.58	rc202	1686	**1702**	-0.95
rc103	747	**753**	-0.80	rc203	1724	1724	0.00
rc104	822	**828**	-0.73	rc204	1724	1724	0.00
rc105	654	**687**	-5.05	rc205	1659	**1682**	-1.39
rc106	678	**688**	-1.47	rc206	1708	**1724**	-0.94
rc107	745	**758**	-1.74	rc207	1713	**1724**	-0.64
rc108	757	**785**	-3.70	rc208	1724	1724	0.00
pr01	598	**604**	-1.00	pr11	632	**641**	-1.42
pr02	899	895	0.44	pr12	902	**949**	-5.21
pr03	946	**973**	-2.85	pr13	1046	**1067**	-2.01
pr04	1195	**1222**	-2.26	pr14	1197	**1266**	-5.76
pr05	1356	**1388**	-2.36	pr15	1488	**1521**	-2.22
pr06	1376	1361	1.09	pr16	1478	**1493**	-1.01
pr07	713	**730**	-2.38	pr17	808	**810**	-0.25
pr08	1082	1081	0.09	pr18	1165	**1191**	-2.23
pr09	1144	**1185**	-3.58	pr19	1238	**1269**	-2.50
pr10	1473	1454	1.29	pr20	1514	**1523**	-0.59

Table A.17: Results comparison between the ILS and our own algorithm for m=4 routes

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
c101	1000	**1020**	-2.00	c201	1810	1810	0.00
c102	1090	**1130**	-3.67	c202	1810	1810	0.00
c103	1150	**1180**	-2.61	c203	1810	1810	0.00
c104	1220	**1240**	-1.64	c204	1810	1810	0.00
c105	1030	**1060**	-2.91	c205	1810	1810	0.00
c106	1040	**1060**	-1.92	c206	1810	1810	0.00
c107	1100	**1110**	-0.91	c207	1810	1810	0.00
c108	1100	**1110**	-0.91	c208	1810	1810	0.00

Continued on next page

Table A.17: Results comparison between the ILS and our own algorithm for m=4 routes

Name	ILS	profit	Gap (%)	Name	ILS	profit	Gap (%)
c109	1180	1180	0.00				
r101	601	**613**	-2.00	r201	1458	1458	0.00
r102	807	**820**	-1.61	r202	1458	1458	0.00
r103	878	**902**	-2.73	r203	1458	1458	0.00
r104	941	**967**	-2.76	r204	1458	1458	0.00
r105	735	**765**	-4.08	r205	1458	1458	0.00
r106	870	**877**	-0.80	r206	1458	1458	0.00
r107	927	**936**	-0.97	r207	1458	1458	0.00
r108	982	**989**	-0.71	r208	1458	1458	0.00
r109	866	**873**	-0.81	r209	1458	1458	0.00
r110	870	**895**	-2.87	r210	1458	1458	0.00
r111	935	**940**	-0.53	r211	1458	1458	0.00
r112	939	**960**	-2.24				
rc101	794	**796**	-0.25	rc201	1724	1724	0.00
rc102	881	**899**	-2.04	rc202	1724	1724	0.00
rc103	947	**961**	-1.48	rc203	1724	1724	0.00
rc104	1019	**1054**	-3.43	rc204	1724	1724	0.00
rc105	841	**857**	-1.90	rc205	1724	1724	0.00
rc106	874	**890**	-1.83	rc206	1724	1724	0.00
rc107	951	**972**	-2.21	rc207	1724	1724	0.00
rc108	998	**1010**	-1.20	rc208	1724	1724	0.00
pr01	644	**654**	-1.55	pr11	654	**657**	-0.46
pr02	1014	**1029**	-1.48	pr12	1041	**1074**	-3.17
pr03	1162	**1163**	-0.09	pr13	1263	**1289**	-2.06
pr04	1452	**1471**	-1.31	pr14	1528	**1536**	-0.52
pr05	1665	**1687**	-1.32	pr15	1818	**1826**	-0.44
pr06	1696	**1723**	-1.59	pr16	1889	1838	2.70
pr07	840	**841**	-0.12	pr17	889	**898**	-1.01
pr08	1267	**1303**	-2.84	pr18	1352	**1406**	-3.99
pr09	1460	**1477**	-1.16	pr19	1560	**1572**	-0.77
pr10	1782	1772	0.56	pr20	1846	**1866**	-1.08

Table A.3: Heuristic algorithms on OP and TOP

Reference	Problem	Algorithm
Golden et al. (1987)	OP	Centre-of-gravity
Ramesh and Brown (1991)	OP	Four-phase heuristic
Chao et al. (1996b) & Chao et al. (1996a)	TOP & OP	Five-step heuristic
Gendreau et al. (1998b)	OP	Tabu search
Archetti et al. (2007)	TOP	Tabu search Slow and Fast Variable Neighborhood Search
Ke et al. (2008)	TOP	Ant Colony Optimization
Vansteenwegen et al. (2009a)	TOP	Guided Local Search
Vansteenwegen et al. (2009c)	TOP	Skewed Variable Neighborhood Seach
Souffriau et al. (2010)	TOP	GRASP with Path Relinking
Sevkli and Sevilgen (2010)	OP	Strengthened Particle Swarm Optimization
ŞEVKLİ and SEVİLGEN (2010)	OP	Discrete Strengthened Particle Swarm Optimization
Muthuswamy and Lam (2011a)	OP	Discrete Particle Swarm Optimization
Chekuri et al. (2012)	OP	Approximation algorithms
Liang et al. (2013)	OP	Multi-Level Variable Neighborhood Search
Campos et al. (2014)	OP	Greedy Randomized Adaptive Search Procedure and Path Relinking
Marinakis et al. (2015)	OP	Memetic-Greedy Randomized Adaptive Search Procedure
Bouly et al. (2010)	TOP	Memetic Algorithm
Muthuswamy and Lam (2011b)	TOP	Discrete Particle Swarm Optimization
Dang et al. (2011)	TOP	Particle Swarm Optimization-based Memetic Algorithm
Dang et al. (2013b)	TOP	Particle Swarm Optimized inspired Algorithm
Vincent and Lin (2014)	TOP	Multi-start Simulated Annealing
Ferreira et al. (2014)	TOP	Genetic Algorithm
Ke et al. (2016)	TOP	Pareto mimic algorithm

Table A.4: Papers on (T)OPTW

Reference	Problem	Algorithm
Kantor and Rosenwein (1992)	OPTW	Score over insertion heuristic
Righini and Salani (2009)	OPTW	Bi-directional dynamic programming algorithm
Mansini et al. (2006)	OPTW	Granular Variable Neighborhood Search
Montemanni and Gambardella (2009)	TOPTW	Ant Colony Optimization
Tricoire et al. (2010)	TOPTW	Variable Neighborhood Search
Vansteenwegen et al. (2009b)	TOPTW	Iterated Local Search
Labadie et al. (2011)	OPTW & TOPTW	Hybrid Greedy Randomised Adaptive Search Procedure and Evolutionary Local Search
Gambardella et al. (2012)	OPTW & TOPTW	Enchanced Ant Colony System
Lin and Vincent (2012)	OPTW & TOPTW	Fast SA and Slow SA
Labadie et al. (2012)	OPTW & TOPTW	LP-based Granular Variable Neighborhood Search
Souffriau et al. (2013)	OPTW & TOPTW	Hybrid Greedy Randomized Adaptive Search Procedure and Iterated Local Search
Gavalas et al. (2013)	OPTW & TOPTW	Cluster Search Cluster Ratio and Cluster Search Cluster Routes
Hu and Lim (2014)	OPTW & TOPTW	Hybrid Local Seach and Simulated Annealing
Cura (2014)	OPTW & TOPTW	Artificial Bee Colony
Duque et al. (2015)	OPTW	Pulse algorithm
Gunawan et al. (2015a)	OPTW	Iterated Local Search
Gunawan et al. (2015b)	OPTW & TOPTW	Hybrid Simulated Annealing and Iterated Local Search
Gunawan et al. (2015c)	OPTW & TOPTW	Well-Tuned Iterated Local Search

Table A.5: Papers on TTDP

Reference	Problem	Algorithm	Application
Fomin and Lingas (2002)	TDOP	2+ε Approximation Algorithm	Moving Targets Interception
Li (2012)	TDOP	Dynamic Programming	Transportation Network
Verbeeck et al. (2014)	TDOP	Ant Colony System	-
Gunawan et al. (2014)	TDOP	Iterated Local Search	Theme Park Navigation Problem
Garcia et al. (2010)	TDOPTW	Hybrid Iterated Local Searchg	Personalized Electronic Tourist Guides
Abbaspour and Samadzadegan (2011)	TDOPTW	Adapted Genetic Algorithm	Tourist Trip Design Problem
Garcia et al. (2013)	TDOPTW	Hybrid Iterated Local Search	Personalised Electronic Tourist guides
Gavalas et al. (2014b)	TDOPTW	Time Dependent CSCRoutes the SlackCSCRoutes	Tourist Trip Design Problem

Table A.6: Papers on SOP

Reference	Problem	Characteristic	Algorithm	Application
Ilhan et al. (2008)	OPSP	Stochastic profits / scores	Exact Solution Algorithm and bi-objective Genetic Algorithm	Logistic Problem
Campbell et al. (2011)	OPSTS	Stochastic travel and service times	Variable Neighborhood Search	Logistic Problem
Papapanagiotou et al. (2014)	OPSTS	Stochastic travel and service times	Monte Carlo sampling and Hybrid Monte Carlo sampling and an analytical solution	Logistic Problem
Lau et al. (2012)	DSOP	Stochastic time-dependent travel times	Hybrid Variable Neighborhood Search and Simulated Annealing	Theme park navigation problem
Varakantham and Kumar (2013)	DSOP	Stochastic time-dependent travel times	Mixed Integer Linear Programming - Sample Average Approximation	Theme park navigation problem
Evers et al. (2014)	OPSW	Stochastic travel and service times	Sample Average Approximation and OPSW heuristic	Logistic problem
Zhang et al. (2014)	SOPTW	Stochastic waiting time	Variable Neighborhood Search	Sales representative planning problem
Verbeeck et al. (2016)	TD-OPSWTW	Stochastic travel times	Local Search	Logistic problem
Dolinskaya et al. (2018)	OPST	Adaptive Stochastic travel times	Adaptive Variable Neighborhood Search	Search and Rescue

Table A.7: Papers on GOP

Reference	Problem	Algorithm	Application
Geem et al. (2005)	GOP	Harmony Search	Tourist trip design problem
Wang et al. (2008)	GOP	Genetic Algorithm	Tourist trip design problem
Silberholz and Golden (2010)	GOP	Two-parameter iterative algorith	Tourist trip design problem
Pietz and Royset (2013)	GOP-RDR	Hybrid Brnach-and-bound and four phase heuristic	Smuggler search problem

Table A.8: Comparison of $profit$ and $profit^2$ ratio functions for 1 route

Set	Instance	$Profit$			$Profit^2$			Percentage difference		
		$Profit$	Time	Activities	$Profit$	Time	Activities	$Profit$	Time	Activities
gavalas	t1	439.35	0.04	9.12	418.96	0.03	7.7	-4.64%	-38.29%	-15.57%
	t2	425.35	0.04	10.03	420.01	0.03	7.84	-1.26%	-30.78%	-21.83%
	t3	463.0	0.03	11.0	430.14	0.03	7.81	-7.10%	-24.70%	-29.00%
	t4	837.18	0.10	25.55	838.58	0.08	24.64	+0.17%	-15.40%	-3.56%
	t5	428.0	0.04	10.0	406.1	0.02	7.02	-5.12%	-39.40%	-29.80%
	t6	902.46	0.13	28.63	885.19	0.11	25.74	-1.91%	-12.29%	-10.09%
	t7	897.38	0.13	27.8	894.54	0.11	25.29	-0.32%	-10.62%	-9.03%
	t8	898.61	0.11	26.89	854.22	0.10	23.91	-4.94%	-2.06%	-11.08%
	t9	457.96	0.04	9.45	440.0	0.02	7.0	-3.92%	-45.04%	-25.93%
	t10	492.3	0.03	9.59	489.57	0.03	7.98	-0.55%	-25.44%	-16.79%
	t11	456.42	0.06	13.58	421.09	0.03	8.2	-7.74%	-55.36%	-39.62%
	t12	460.0	0.04	10.0	436.71	0.03	8.22	-5.06%	-32.38%	-17.80%
	t13	453.0	0.03	9.0	434.53	0.03	8.01	-4.08%	-21.03%	-11.00%
	t14	475.74	0.04	8.99	467.67	0.02	7.11	-1.70%	-34.94%	-20.91%
	t15	444.94	0.04	10.0	427.43	0.03	7.88	-3.94%	-35.07%	-21.20%
	t16	428.59	0.04	11.19	396.33	0.02	7.03	-7.53%	-48.92%	-37.18%
	t17	460.0	0.04	9.0	456.0	0.02	7.0	-0.87%	-35.75%	-22.22%
	t18	404.98	0.04	9.5	387.89	0.02	7.06	-4.22%	-48.84%	-25.68%
	t19	464.0	0.03	10.0	451.1	0.03	8.7	-2.78%	-23.36%	-13.00%
	t20	449.99	0.03	8.0	450.0	0.02	8.0	+0.00%	-20.85%	+0.00%
MontemanniTOPTW1	c201	800.0	0.07	32.0	786.8	0.06	30.73	-1.65%	-13.36%	-3.97%
	c202	852.9	0.11	31.77	857.3	0.10	31.43	+0.52%	-5.83%	-1.07%
	c203	911.3	0.14	32.86	906.9	0.13	32.38	-0.48%	-7.56%	-1.46%
	c204	940.1	0.21	33.15	936.6	0.19	32.75	-0.37%	-9.91%	-1.21%
	c205	868.2	0.08	31.33	864.7	0.08	30.38	-0.40%	+1.09%	-3.03%
	c206	897.5	0.10	32.13	893.3	0.10	31.24	-0.47%	+6.22%	-2.77%
	c207	893.4	0.14	33.33	882.4	0.11	30.61	-1.23%	-19.34%	-8.16%
	c208	911.6	0.12	32.71	913.3	0.10	31.67	+0.19%	-18.71%	-3.18%
	r201	739.9	0.11	39.18	762.1	0.10	38.4	+3.00%	-10.20%	-1.99%
	r202	862.99	0.16	48.07	871.71	0.16	46.33	+1.01%	-2.41%	-3.62%
	r203	954.87	0.25	53.42	963.81	0.25	50.23	+0.94%	-1.13%	-5.97%
	r204	1043.27	0.39	56.63	1051.08	0.36	54.83	+0.75%	-7.37%	-3.18%
	r205	854.12	0.17	45.54	884.37	0.17	43.84	+3.54%	+0.87%	-3.73%
	r206	937.82	0.25	51.62	967.43	0.22	49.41	+3.16%	-9.10%	-4.28%
	r207	1003.23	0.33	54.84	1014.13	0.27	51.95	+1.09%	-17.69%	-5.27%
	r208	1070.68	0.44	57.85	1073.44	0.41	55.82	+0.26%	-7.82%	-3.51%
	r209	879.31	0.20	47.87	897.53	0.21	46.08	+2.07%	+6.08%	-3.74%
	r210	915.15	0.22	50.5	929.22	0.21	48.42	+1.54%	-3.34%	-4.12%
	r211	980.89	0.30	52.15	997.38	0.29	50.16	+1.68%	-0.38%	-3.82%
	rc201	734.7	0.11	35.56	763.4	0.08	35.19	+3.91%	-23.06%	-1.04%
	rc202	861.95	0.14	43.27	879.28	0.14	40.7	+2.01%	-6.01%	-5.94%
	rc203	937.1	0.21	48.71	938.49	0.19	44.36	+0.15%	-11.92%	-8.93%
	rc204	1068.25	0.37	52.79	1079.34	0.30	47.43	+1.04%	-20.23%	-10.15%
	rc205	802.42	0.14	39.85	819.47	0.13	38.18	+2.12%	-6.27%	-4.19%
	rc206	823.88	0.15	40.95	847.32	0.13	37.49	+2.85%	-16.62%	-8.45%
	rc207	887.75	0.19	44.21	909.28	0.18	41.91	+2.43%	-1.34%	-5.20%
	rc208	954.07	0.24	46.83	988.4	0.23	43.55	+3.60%	-4.23%	-7.00%

Table A.9: Comparison of $profit$ and $profit^2$ ratio functions for 1 route

Set	Instance	$Profit$			$Profit^2$			Percentage difference		
		$Profit$	Time	Activities	$Profit$	Time	Activities	$Profit$	Time	Activities
MontemanniTOPTW2	pr11	319.3	0.05	22.21	325.21	0.02	22.1	+1.85%	-46.66%	-0.50%
	pr12	420.67	0.09	25.82	420.17	0.09	24.72	-0.12%	-0.51%	-4.26%
	pr13	429.83	0.12	26.81	439.59	0.07	26.71	+2.27%	-45.49%	-0.37%
	pr14	488.84	0.16	30.04	498.76	0.10	28.61	+2.03%	-39.76%	-4.76%
	pr15	638.07	0.27	38.51	639.95	0.20	36.97	+0.29%	-24.92%	-4.00%
	pr16	542.84	0.23	32.44	556.85	0.17	31.36	+2.58%	-23.97%	-3.33%
	pr17	335.8	0.06	21.07	343.9	0.03	20.92	+2.41%	-50.81%	-0.71%
	pr18	437.32	0.11	26.35	447.82	0.07	25.43	+2.40%	-39.49%	-3.49%
	pr19	449.61	0.15	29.16	478.63	0.15	29.46	+6.45%	+0.31%	+1.03%
	pr20	578.51	0.23	35.41	588.74	0.19	34.42	+1.77%	-20.78%	-2.80%
righiniTOPTW1	50_c101	160.0	0.01	8.0	140.0	0.01	7.0	-12.50%	-4.99%	-12.50%
	50_c102	230.0	0.02	11.0	230.0	0.02	11.0	+0.00%	+1.07%	+0.00%
	50_c103	270.0	0.03	11.99	270.0	0.02	11.06	+0.00%	-25.55%	-7.76%
	50_c104	290.0	0.03	12.0	290.0	0.03	11.66	+0.00%	-12.11%	-2.83%
	50_c105	240.0	0.01	12.0	228.6	0.01	11.43	-4.75%	-3.82%	-4.75%
	50_c106	180.0	0.01	10.0	180.0	0.01	10.0	+0.00%	+6.34%	+0.00%
	50_c107	270.0	0.02	12.0	270.0	0.02	12.0	+0.00%	+2.08%	+0.00%
	50_c108	280.0	0.02	12.0	280.0	0.02	12.0	+0.00%	+4.01%	+0.00%
	50_c109	310.0	0.02	12.0	310.0	0.02	12.0	+0.00%	+3.03%	+0.00%
	50_r101	118.0	0.01	7.0	114.0	0.01	6.0	-3.39%	-3.52%	-14.29%
	50_r102	178.22	0.01	10.74	178.52	0.01	10.8	+0.17%	+5.60%	+0.56%
	50_r103	201.0	0.02	11.0	200.31	0.02	10.77	-0.34%	-7.49%	-2.09%
	50_r104	212.14	0.02	11.01	216.0	0.02	11.0	+1.82%	-2.73%	-0.09%
	50_r105	146.08	0.01	7.82	135.65	0.01	7.0	-7.14%	-18.39%	-10.49%
	50_r106	190.7	0.02	10.1	191.12	0.01	10.02	+0.22%	-8.49%	-0.79%
	50_r107	206.96	0.02	11.0	205.88	0.02	10.92	-0.52%	+5.03%	-0.73%
	50_r108	212.45	0.02	11.0	216.0	0.02	11.0	+1.67%	-22.06%	+0.00%
	50_r109	188.0	0.01	10.0	188.0	0.01	10.0	+0.00%	-6.84%	+0.00%
	50_r110	189.86	0.02	10.76	192.71	0.02	10.71	+1.50%	+1.26%	-0.46%
	50_r111	206.98	0.02	11.0	205.62	0.02	10.9	-0.66%	-15.92%	-0.91%
	50_r112	210.19	0.02	11.01	210.09	0.02	11.0	-0.05%	-16.87%	-0.09%
	50_rc101	170.0	0.01	8.19	170.0	0.01	8.04	+0.00%	-3.58%	-1.83%
	50_rc102	217.0	0.01	10.13	220.0	0.01	10.0	+1.38%	-13.12%	-1.28%
	50_rc103	206.2	0.01	11.2	204.9	0.01	9.17	-0.63%	-6.80%	-18.12%
	50_rc104	200.0	0.02	12.0	242.6	0.02	10.95	+21.30%	+19.86%	-8.75%
	50_rc105	185.5	0.01	10.42	190.0	0.01	10.0	+2.43%	-4.06%	-4.03%
	50_rc106	196.1	0.01	10.17	190.3	0.01	10.92	-2.96%	-4.87%	+7.37%
	50_rc107	210.0	0.01	10.0	219.8	0.01	10.01	+4.67%	+15.93%	+0.10%
	50_rc108	202.6	0.02	11.82	225.8	0.01	10.58	+11.45%	-6.21%	-10.49%

Table A.10: Comparison of $profit$ and $profit^2$ ratio functions for 1 route

Set	Instance	$Profit$			$Profit^2$			Percentage difference		
		$Profit$	Time	Activities	$Profit$	Time	Activities	$Profit$	Time	Activities
righiniTOPTW2	c101	180.0	0.01	7.0	180.0	0.01	7.0	+0.00%	-5.17%	+0.00%
	c102	270.0	0.03	11.0	261.2	0.02	10.12	-3.26%	-28.88%	-8.00%
	c103	348.2	0.05	11.97	340.1	0.04	11.01	-2.33%	-31.45%	-8.02%
	c104	370.0	0.06	12.0	370.0	0.06	12.0	+0.00%	-1.24%	+0.00%
	c105	272.9	0.02	10.83	260.0	0.02	10.0	-4.73%	-3.98%	-7.66%
	c106	290.0	0.02	10.0	290.0	0.02	10.0	+0.00%	-1.01%	+0.00%
	c107	319.3	0.02	11.96	318.3	0.02	11.83	-0.31%	+3.51%	-1.09%
	c108	330.0	0.03	12.0	330.0	0.03	12.0	+0.00%	-4.00%	+0.00%
	c109	350.0	0.04	12.0	350.0	0.03	12.0	+0.00%	-3.56%	+0.00%
	r101	174.0	0.01	10.0	143.0	0.01	7.0	-17.82%	-16.05%	-30.00%
	r102	268.9	0.04	12.84	274.65	0.03	12.06	+2.14%	-26.55%	-6.07%
	r103	279.57	0.04	13.88	275.01	0.04	12.01	-1.63%	-11.33%	-13.47%
	r104	287.38	0.05	14.12	285.67	0.05	13.09	-0.60%	-4.33%	-7.29%
	r105	218.5	0.02	11.92	211.69	0.02	11.17	-3.12%	-0.12%	-6.29%
	r106	265.79	0.04	13.07	274.64	0.03	12.01	+3.33%	-20.10%	-8.11%
	r107	280.94	0.04	14.0	279.08	0.04	13.15	-0.66%	+0.20%	-6.07%
	r108	288.47	0.04	14.01	291.64	0.04	14.0	+1.10%	+0.65%	-0.07%
	r109	261.81	0.03	13.39	263.42	0.03	13.53	+0.61%	+6.95%	+1.05%
	r110	266.88	0.03	13.04	268.0	0.03	13.0	+0.42%	-14.50%	-0.31%
	r111	279.03	0.04	13.97	277.65	0.04	13.61	-0.49%	+2.29%	-2.58%
	r112	279.84	0.04	14.0	277.2	0.04	12.7	-0.94%	-10.58%	-9.29%
	rc101	195.64	0.02	10.21	195.13	0.01	10.0	-0.26%	-16.20%	-2.06%
	rc102	239.5	0.03	12.09	237.82	0.02	11.05	-0.70%	-24.21%	-8.60%
	rc103	239.16	0.03	12.02	242.56	0.03	11.46	+1.42%	-20.89%	-4.66%
	rc104	256.98	0.04	12.81	265.81	0.03	11.07	+3.44%	-21.87%	-13.58%
	rc105	213.27	0.02	11.91	213.51	0.02	11.56	+0.11%	-7.39%	-2.94%
	rc106	232.0	0.02	12.44	228.87	0.02	11.89	-1.35%	+1.74%	-4.42%
	rc107	260.31	0.03	13.01	261.56	0.03	12.49	+0.48%	-9.39%	-4.00%
	rc108	266.64	0.03	13.14	277.27	0.03	11.99	+3.99%	-14.21%	-8.75%
righiniTOPTW3	pr01	298.62	0.03	21.77	301.63	0.04	21.69	+1.01%	+13.25%	-0.37%
	pr02	368.75	0.06	22.48	372.1	0.05	22.17	+0.91%	-9.79%	-1.38%
	pr03	364.62	0.07	22.27	374.62	0.06	22.02	+2.74%	-1.10%	-1.12%
	pr04	443.53	0.11	25.96	445.28	0.08	25.13	+0.39%	-24.07%	-3.20%
	pr05	538.38	0.17	32.84	552.93	0.17	31.9	+2.70%	-3.17%	-2.86%
	pr06	494.37	0.12	28.65	522.9	0.12	27.5	+5.77%	-0.28%	-4.01%
	pr07	269.56	0.04	17.77	280.25	0.03	17.49	+3.97%	-12.39%	-1.58%
	pr08	431.01	0.08	25.09	436.1	0.07	25.02	+1.18%	-7.97%	-0.28%
	pr09	433.87	0.12	26.64	454.82	0.10	26.43	+4.83%	-13.44%	-0.79%
	pr10	515.41	0.16	29.41	525.67	0.15	28.8	+1.99%	-5.44%	-2.07%

Table A.11: Comparison of $profit$ and $profit^2$ ratio functions for 3 routes

Set	Instance	$Profit$			$Profit^2$			Percentage difference		
		$Profit$	Time	Activities	$Profit$	Time	Activities	$Profit$	Time	Activities
gavalas	t1	1238.08	0.17	30.27	1215.35	0.12	24.25	-1.84%	-29.86%	-19.89%
	t2	1213.0	0.17	29.67	1178.68	0.11	23.26	-2.83%	-35.98%	-21.60%
	t3	1225.57	0.17	29.69	1202.34	0.13	24.0	-1.90%	-24.69%	-19.16%
	t4	1772.8	0.20	47.93	1790.48	0.17	46.61	+1.00%	-18.06%	-2.75%
	t5	1177.85	0.15	29.17	1159.73	0.11	22.76	-1.54%	-25.66%	-21.97%
	t6	1993.82	0.26	60.72	2030.18	0.21	56.84	+1.82%	-20.26%	-6.39%
	t7	2078.71	0.27	62.16	2095.43	0.21	60.17	+0.80%	-23.20%	-3.20%
	t8	2051.33	0.28	60.66	2016.14	0.19	57.15	-1.72%	-31.90%	-5.79%
	t9	1251.37	0.18	31.41	1213.45	0.12	24.72	-3.03%	-35.20%	-21.30%
	t10	1343.15	0.15	28.26	1334.73	0.12	25.45	-0.63%	-20.57%	-9.94%
	t11	1239.87	0.21	33.79	1193.89	0.12	25.51	-3.71%	-44.13%	-24.50%
	t12	1215.17	0.18	29.04	1178.98	0.12	24.6	-2.98%	-31.94%	-15.29%
	t13	1217.28	0.18	30.11	1193.39	0.13	25.03	-1.96%	-27.76%	-16.87%
	t14	1283.86	0.16	27.57	1269.74	0.12	23.78	-1.10%	-25.96%	-13.75%
	t15	1220.65	0.16	26.81	1206.56	0.12	22.96	-1.15%	-28.49%	-14.36%
	t16	1186.03	0.18	30.21	1147.4	0.12	24.24	-3.26%	-34.17%	-19.76%
	t17	1222.37	0.16	26.73	1198.06	0.11	23.31	-1.99%	-34.71%	-12.79%
	t18	1136.13	0.16	26.88	1135.16	0.12	22.2	-0.09%	-27.57%	-17.41%
	t19	1251.18	0.17	29.07	1238.05	0.12	24.24	-1.05%	-29.45%	-16.62%
	t20	1235.83	0.15	25.82	1211.51	0.11	23.08	-1.97%	-26.52%	-10.61%
MontemanniTOPTW1	c201	1733.0	0.32	98.33	1716.7	0.31	96.69	-0.94%	-3.25%	-1.67%
	c202	1704.1	0.35	95.59	1699.9	0.33	95.1	-0.25%	-6.16%	-0.51%
	c203	1713.2	0.38	96.39	1708.5	0.38	95.94	-0.27%	-0.07%	-0.47%
	c204	1748.9	0.47	99.9	1744.2	0.45	99.42	-0.27%	-3.83%	-0.48%
	c205	1745.1	0.35	99.53	1725.6	0.37	97.56	-1.12%	+3.69%	-1.98%
	c206	1757.6	0.39	100.76	1741.0	0.40	99.12	-0.94%	+2.81%	-1.63%
	c207	1760.0	0.42	101.01	1748.2	0.42	99.82	-0.67%	-0.81%	-1.18%
	c208	1784.5	0.44	103.46	1771.9	0.43	102.19	-0.71%	-2.40%	-1.23%
	r201	1382.5	0.36	96.48	1383.27	0.37	95.95	+0.06%	+1.35%	-0.55%
	r202	1420.74	0.38	101.26	1423.52	0.39	101.26	+0.20%	+1.76%	+0.00%
	r203	1457.72	0.42	105.91	1457.72	0.44	105.92	+0.00%	+4.62%	+0.01%
	r204	1458.0	0.42	106.0	1458.0	0.43	106.0	+0.00%	+1.42%	+0.00%
	r205	1457.49	0.39	105.81	1456.94	0.43	105.68	-0.04%	+9.46%	-0.12%
	r206	1458.0	0.36	106.0	1458.0	0.38	106.0	+0.00%	+5.93%	+0.00%
	r207	1458.0	0.38	106.0	1458.0	0.40	106.0	+0.00%	+4.99%	+0.00%
	r208	1458.0	0.46	106.0	1458.0	0.47	106.0	+0.00%	+2.15%	+0.00%
	r209	1457.96	0.41	105.96	1457.72	0.42	105.8	-0.02%	+3.64%	-0.15%
	r210	1458.0	0.39	106.0	1458.0	0.39	106.0	+0.00%	+0.67%	+0.00%
	r211	1458.0	0.41	106.0	1458.0	0.41	106.0	+0.00%	+2.11%	+0.00%
	rc201	1604.09	0.33	92.38	1605.77	0.34	91.84	+0.10%	+3.67%	-0.58%
	rc202	1668.09	0.37	99.55	1666.95	0.40	99.02	-0.07%	+7.52%	-0.53%
	rc203	1712.98	0.42	104.51	1712.66	0.42	104.46	-0.02%	+0.07%	-0.05%
	rc204	1724.0	0.44	106.0	1724.0	0.44	106.0	+0.00%	-0.22%	+0.00%
	rc205	1627.73	0.37	96.34	1633.19	0.39	95.79	+0.34%	+4.96%	-0.57%
	rc206	1704.08	0.44	102.86	1702.08	0.42	102.21	-0.12%	-4.54%	-0.63%
	rc207	1706.36	0.43	103.34	1705.66	0.46	102.99	-0.04%	+7.75%	-0.34%
	rc208	1724.0	0.41	106.0	1724.0	0.43	106.0	+0.00%	+4.02%	+0.00%

Table A.12: Comparison of $profit$ and $profit^2$ ratio functions for 3 routes

Set	Instance	$Profit$			$Profit^2$			Percentage difference		
		$Profit$	Time	Activities	$Profit$	Time	Activities	$Profit$	Time	Activities
MontemanniTOPTW2	pr11	617.77	0.10	48.82	622.91	0.07	48.43	+0.83%	-23.75%	-0.80%
	pr12	897.14	0.26	62.58	903.27	0.28	61.79	+0.68%	+8.42%	-1.26%
	pr13	1016.2	0.48	72.2	1043.0	0.39	70.71	+2.64%	-18.73%	-2.06%
	pr14	1187.18	0.77	78.18	1202.57	0.67	74.73	+1.30%	-12.98%	-4.41%
	pr15	1418.75	1.28	88.2	1447.12	1.24	86.55	+2.00%	-3.30%	-1.87%
	pr16	1406.51	1.54	86.55	1431.34	1.40	84.26	+1.77%	-9.20%	-2.65%
	pr17	774.28	0.19	56.33	780.0	0.14	55.06	+0.74%	-29.45%	-2.25%
	pr18	1116.69	0.51	70.67	1138.52	0.43	69.24	+1.95%	-15.07%	-2.02%
	pr19	1188.02	1.03	82.35	1232.03	0.92	79.39	+3.70%	-10.07%	-3.59%
	pr20	1435.57	1.46	90.19	1476.64	1.40	87.06	+2.86%	-4.37%	-3.47%
righiniTOPTW1	50_c101	410.0	0.02	23.0	410.0	0.02	23.0	+0.00%	-0.97%	+0.00%
	50_c102	514.2	0.05	30.58	514.9	0.04	30.57	+0.14%	-4.69%	-0.03%
	50_c103	592.8	0.07	34.48	590.5	0.06	33.01	-0.39%	-12.46%	-4.26%
	50_c104	640.0	0.10	36.84	640.0	0.11	36.5	+0.00%	+4.57%	-0.92%
	50_c105	582.5	0.04	32.33	582.7	0.04	32.27	+0.03%	-4.39%	-0.19%
	50_c106	478.5	0.03	26.88	477.0	0.03	26.7	-0.31%	-6.19%	-0.67%
	50_c107	629.5	0.06	35.99	628.8	0.06	35.9	-0.11%	+7.51%	-0.25%
	50_c108	648.4	0.06	36.47	645.3	0.07	36.35	-0.48%	+3.84%	-0.33%
	50_c109	684.7	0.09	38.81	683.9	0.10	38.78	-0.12%	+4.20%	-0.08%
	50_r101	300.63	0.03	19.5	297.8	0.03	18.91	-0.94%	-3.80%	-3.03%
	50_r102	419.97	0.04	27.87	418.45	0.04	27.55	-0.36%	-7.42%	-1.15%
	50_r103	460.22	0.06	29.95	462.91	0.05	29.57	+0.58%	-14.08%	-1.27%
	50_r104	518.94	0.09	33.35	518.95	0.08	33.21	+0.00%	-5.56%	-0.42%
	50_r105	379.54	0.04	24.24	387.93	0.03	24.01	+2.21%	-9.44%	-0.95%
	50_r106	459.99	0.05	30.97	467.93	0.05	30.87	+1.73%	-9.38%	-0.32%
	50_r107	497.2	0.06	32.17	498.38	0.06	31.52	+0.24%	-10.32%	-2.02%
	50_r108	529.64	0.09	34.08	530.73	0.08	33.9	+0.21%	-12.15%	-0.53%
	50_r109	458.03	0.05	28.6	464.25	0.06	28.04	+1.36%	+5.28%	-1.96%
	50_r110	482.58	0.07	30.31	486.07	0.06	30.06	+0.72%	-9.31%	-0.82%
	50_r111	500.49	0.08	31.48	497.69	0.06	31.59	-0.56%	-26.02%	+0.35%
	50_r112	538.51	0.09	33.04	536.26	0.09	32.97	-0.42%	-8.07%	-0.21%
	50_rc101	508.7	0.03	28.91	500.0	0.03	27.0	-1.71%	+3.27%	-6.61%
	50_rc102	559.7	0.04	32.09	598.7	0.04	30.01	+6.97%	+16.85%	-6.48%
	50_rc103	603.9	0.04	32.77	622.4	0.05	30.36	+3.06%	+27.65%	-7.35%
	50_rc104	625.5	0.06	34.94	654.4	0.06	31.6	+4.62%	+11.08%	-9.56%
	50_rc105	520.3	0.03	30.97	521.8	0.03	27.45	+0.29%	+6.96%	-11.37%
	50_rc106	565.7	0.05	32.83	577.1	0.05	31.61	+2.02%	-0.16%	-3.72%
	50_rc107	588.0	0.05	34.28	626.9	0.05	34.57	+6.62%	+6.92%	+0.85%
	50_rc108	609.1	0.06	35.47	616.8	0.06	31.57	+1.26%	+0.74%	-11.00%

Table A.13: Comparison of $profit$ and $profit^2$ ratio functions for 3 routes

Set	Instance	$Profit$			$Profit^2$			Percentage difference		
		$Profit$	Time	Activities	$Profit$	Time	Activities	$Profit$	Time	Activities
righiniTOPTW2	c101	500.0	0.05	23.0	500.0	0.04	23.0	+0.00%	-16.73%	+0.00%
	c102	652.6	0.10	31.53	655.2	0.10	31.37	+0.40%	-5.73%	-0.51%
	c103	824.1	0.18	34.14	818.3	0.14	33.46	-0.70%	-19.04%	-1.99%
	c104	895.8	0.23	36.24	891.1	0.19	35.52	-0.52%	-15.66%	-1.99%
	c105	732.8	0.07	33.11	732.6	0.08	32.51	-0.03%	+21.28%	-1.81%
	c106	733.7	0.08	32.39	722.9	0.08	32.29	-1.47%	+7.28%	-0.31%
	c107	806.9	0.10	35.88	806.4	0.10	35.57	-0.06%	+0.15%	-0.86%
	c108	829.6	0.11	36.24	829.6	0.12	36.28	+0.00%	+6.72%	+0.11%
	c109	881.3	0.16	36.78	880.3	0.17	36.37	-0.11%	+6.26%	-1.11%
	r101	425.44	0.06	24.91	421.01	0.05	23.02	-1.04%	-13.27%	-7.59%
	r102	608.56	0.10	34.46	614.42	0.10	32.89	+0.96%	+3.08%	-4.56%
	r103	668.61	0.16	37.13	665.2	0.15	36.04	-0.51%	-6.11%	-2.94%
	r104	729.28	0.22	40.41	730.53	0.20	39.1	+0.17%	-8.07%	-3.24%
	r105	540.43	0.08	31.53	555.45	0.07	30.34	+2.78%	-13.29%	-3.77%
	r106	654.49	0.13	36.43	670.9	0.11	35.8	+2.51%	-14.10%	-1.73%
	r107	709.04	0.14	38.92	709.3	0.13	37.57	+0.04%	-7.31%	-3.47%
	r108	741.04	0.22	39.73	749.69	0.20	39.06	+1.17%	-6.91%	-1.69%
	r109	644.19	0.11	35.68	653.28	0.12	35.21	+1.41%	+10.52%	-1.32%
	r110	681.03	0.14	37.7	697.05	0.14	37.52	+2.35%	-0.77%	-0.48%
	r111	723.13	0.16	38.67	715.26	0.17	37.69	-1.09%	+6.96%	-2.53%
	r112	701.16	0.18	39.34	729.55	0.18	37.4	+4.05%	+2.68%	-4.93%
	rc101	561.74	0.06	31.99	562.26	0.07	31.33	+0.09%	+34.13%	-2.06%
	rc102	643.72	0.10	35.49	658.12	0.09	34.51	+2.24%	-6.99%	-2.76%
	rc103	673.6	0.14	36.22	697.07	0.11	33.76	+3.48%	-18.64%	-6.79%
	rc104	749.18	0.16	39.48	782.05	0.14	36.38	+4.39%	-10.03%	-7.85%
	rc105	600.17	0.10	34.58	601.73	0.08	32.14	+0.26%	-24.51%	-7.06%
	rc106	639.97	0.09	34.96	644.93	0.09	34.12	+0.78%	-1.64%	-2.40%
	rc107	709.45	0.12	37.6	711.08	0.11	34.83	+0.23%	-9.49%	-7.37%
	rc108	726.17	0.14	38.78	744.3	0.14	36.46	+2.50%	-2.67%	-5.98%
righiniTOPTW3	pr01	572.31	0.09	45.82	576.62	0.08	45.45	+0.75%	-5.49%	-0.81%
	pr02	845.79	0.19	58.3	858.99	0.20	57.76	+1.56%	+3.63%	-0.93%
	pr03	906.6	0.30	61.82	931.31	0.30	60.43	+2.73%	-1.10%	-2.25%
	pr04	1135.13	0.54	72.3	1155.7	0.47	70.33	+1.81%	-12.15%	-2.72%
	pr05	1308.12	0.92	82.79	1327.66	0.86	79.59	+1.49%	-6.47%	-3.87%
	pr06	1289.62	0.95	79.28	1319.55	0.92	77.07	+2.32%	-2.86%	-2.79%
	pr07	700.26	0.13	50.67	705.34	0.11	50.26	+0.73%	-20.65%	-0.81%
	pr08	1023.4	0.34	64.43	1033.11	0.31	63.21	+0.95%	-10.29%	-1.89%
	pr09	1099.23	0.63	74.29	1126.75	0.57	72.05	+2.50%	-9.14%	-3.02%
	pr10	1353.36	1.05	83.83	1380.22	1.01	81.75	+1.98%	-4.26%	-2.48%

Appendix B

Figures

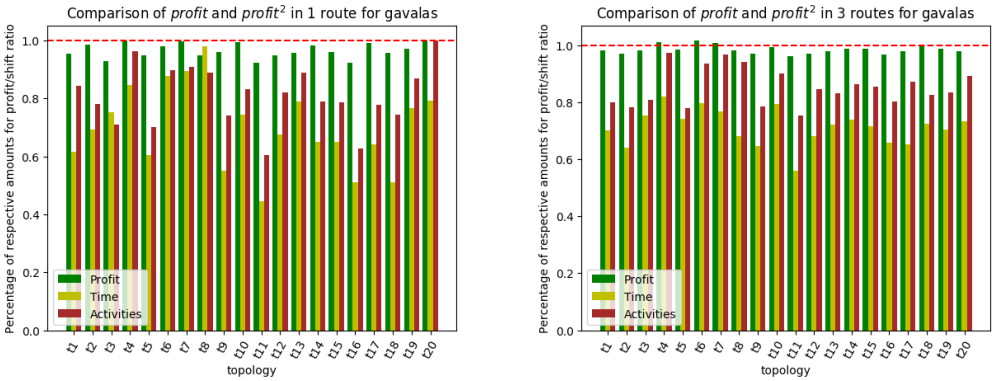


Figure B.1: Relative results for $profit$ and $profit^2$ for Gavalas dataset

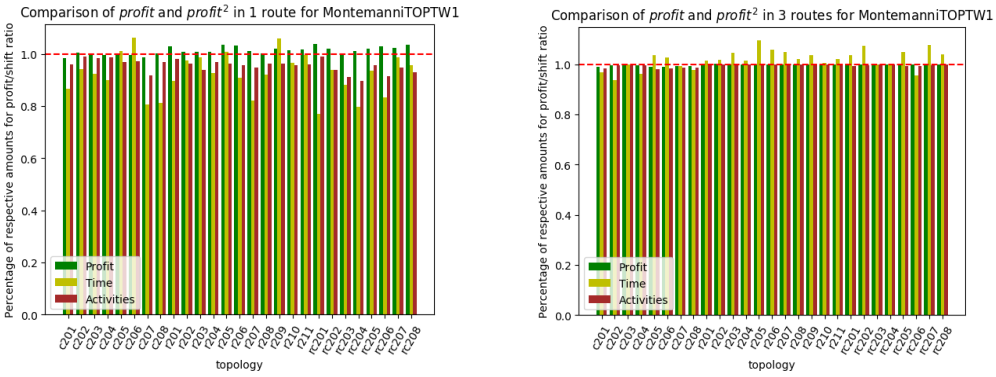


Figure B.2: Relative results for $profit$ and $profit^2$ for MontemanniTOPTW1 dataset

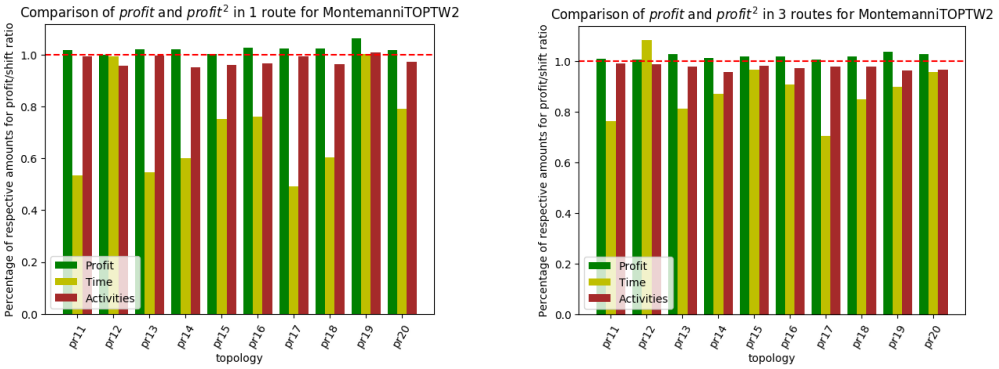


Figure B.3: Relative results for $profit$ and $profit^2$ for MontemanniTOPTW2 dataset

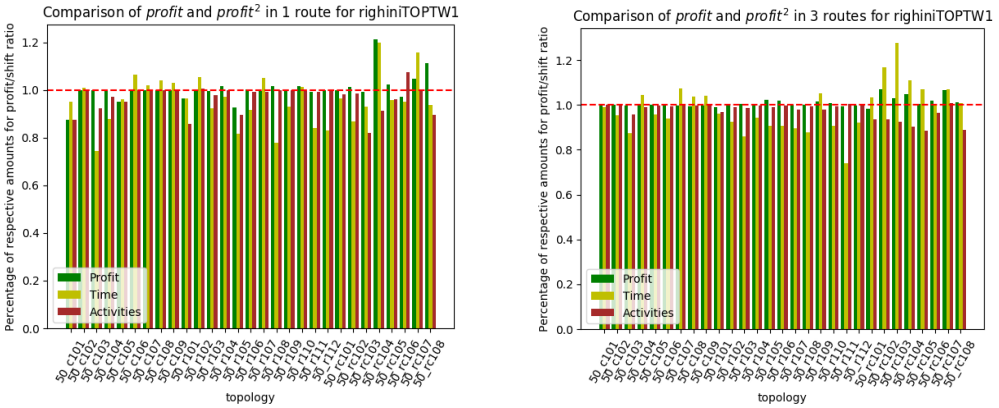


Figure B.4: Relative results for $profit$ and $profit^2$ for righiniTOPTW1 dataset

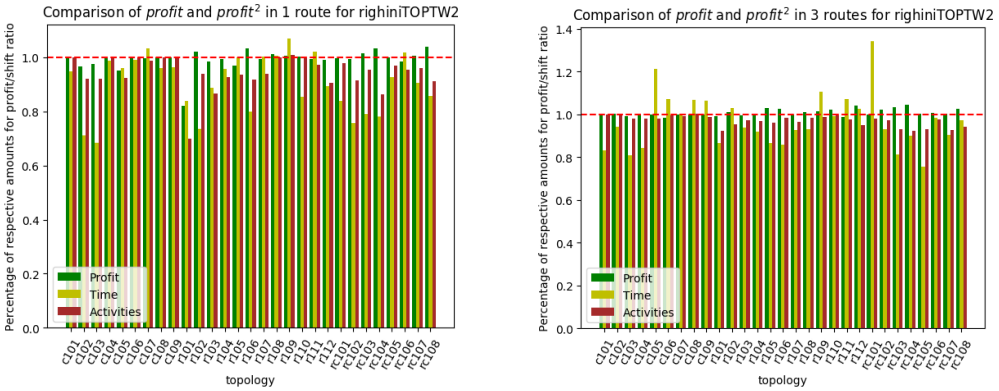


Figure B.5: Relative results for $profit$ and $profit^2$ for righiniTOPTW2 dataset

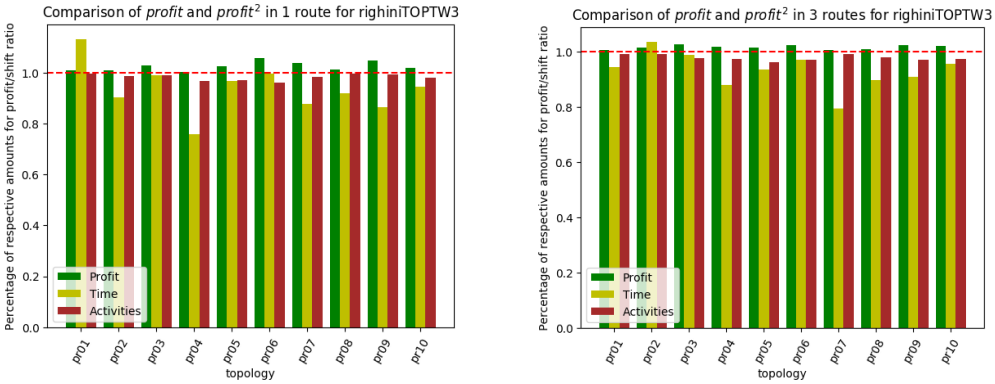


Figure B.6: Relative results for $profit$ and $profit^2$ for righiniTOPTW3 dataset

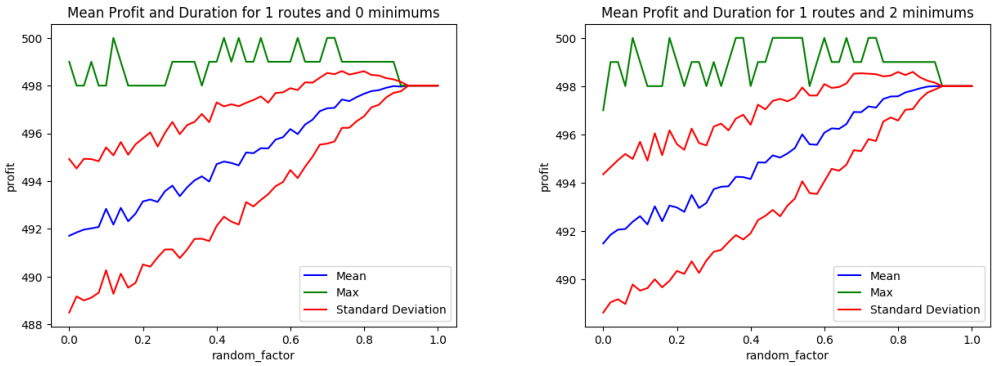


Figure B.7: Mean Profit relative to random factor

Figure B.8: Mean Profit relative to random factor

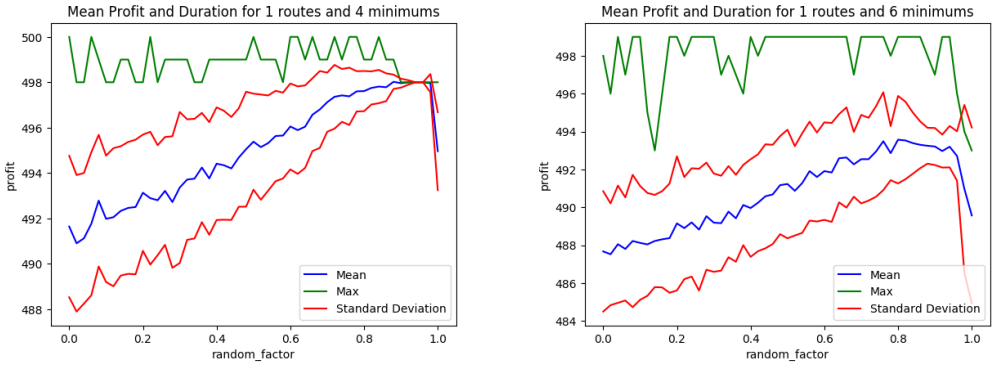


Figure B.9: Mean Profit relative to random factor

Figure B.10: Mean Profit relative to random factor

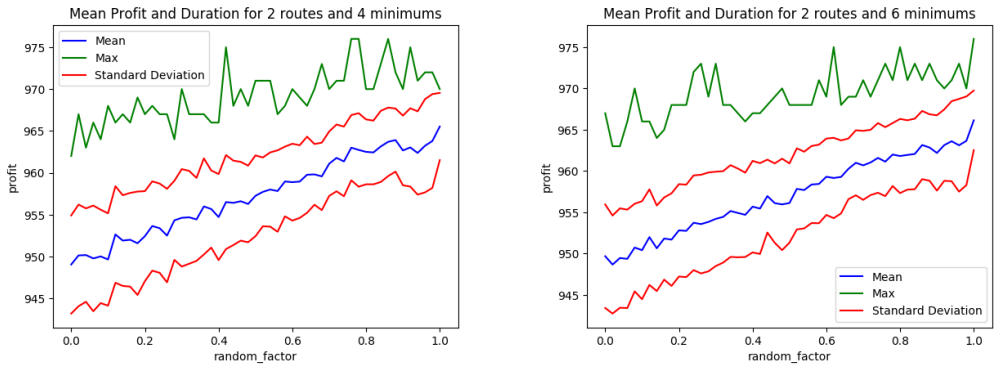


Figure B.11: Mean Profit relative to random factor

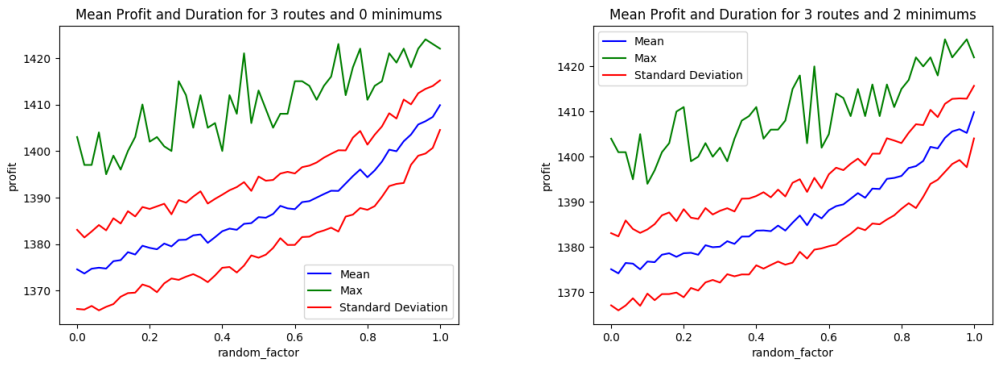


Figure B.12: Mean Profit relative to random factor

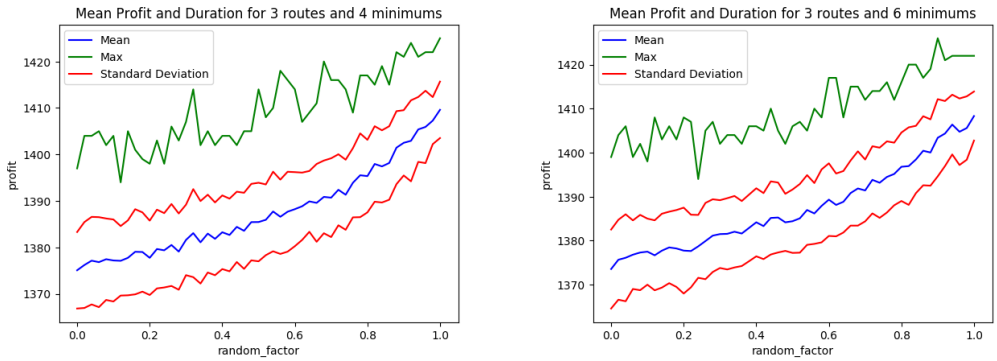


Figure B.13: Mean Profit relative to random factor

Figure B.14: Relative results for including $2-opt$ moves and inserting for Gavalas dataset

Figure B.15: Relative results for including $2-opt$ moves and inserting for MontemanniTOPTW1 dataset

Figure B.16: Relative results for including $2-opt$ moves and inserting for MontemanniTOPTW2 dataset

Figure B.17: Relative results for including $2-opt$ moves and inserting for righiniTOPTW1 dataset

Figure B.18: Relative results for including $2-opt$ moves and inserting for righiniTOPTW2 dataset

Figure B.19: Relative results for including $2-opt$ moves and inserting for righiniTOPTW3 dataset

Figure B.20: Relative results for including $2-opt$ moves and replacing for Gavalas dataset

Figure B.21: Relative results for including $2-opt$ moves and replacing for MontemanniTOPTW1 dataset

Figure B.22: Relative results for including $2-opt$ moves and replacing for MontemanniTOPTW2 dataset

Figure B.23: Relative results for including $2-opt$ moves and replacing for righiniTOPTW1 dataset

Figure B.24: Relative results for including $2-opt$ moves and replacing for righiniTOPTW2 dataset

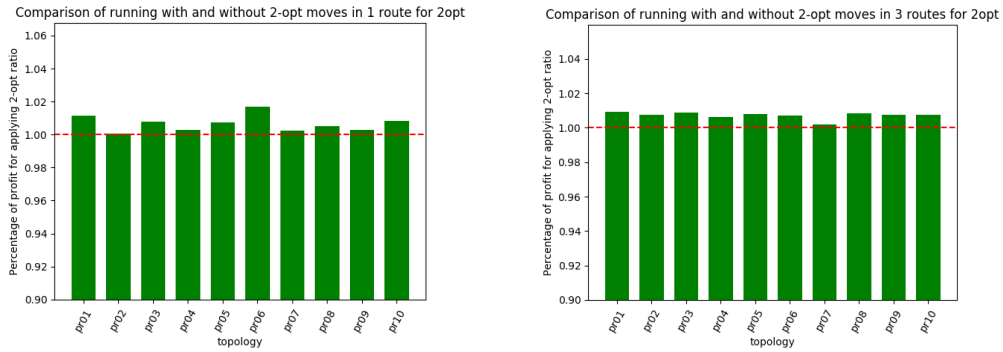
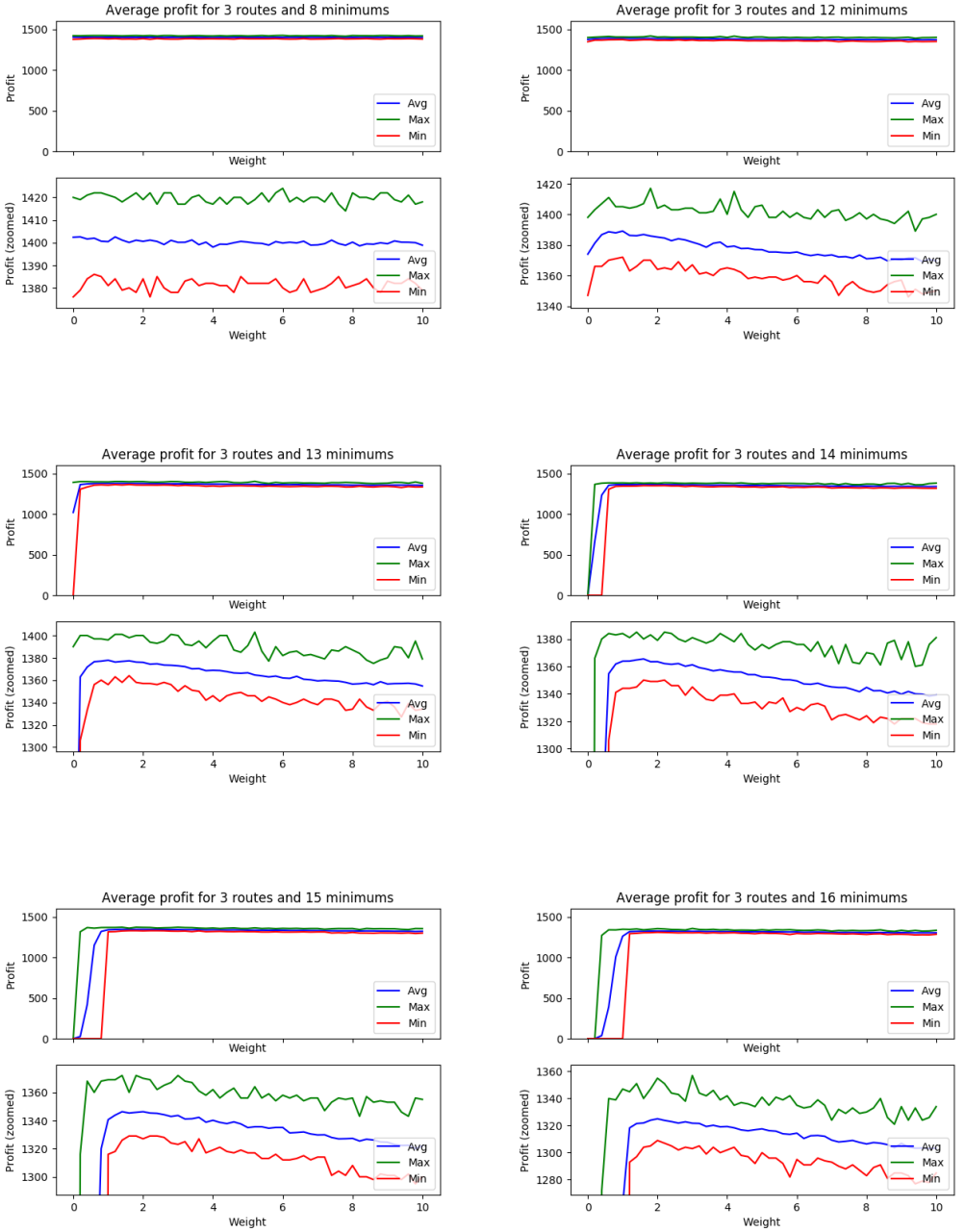


Figure B.25: Relative results for including $2-opt$ moves and replacing for righiniTOPTW3 dataset

Figure B.26: Profits relative to single weight for 8–20 minimums of category 1 for 3 routes



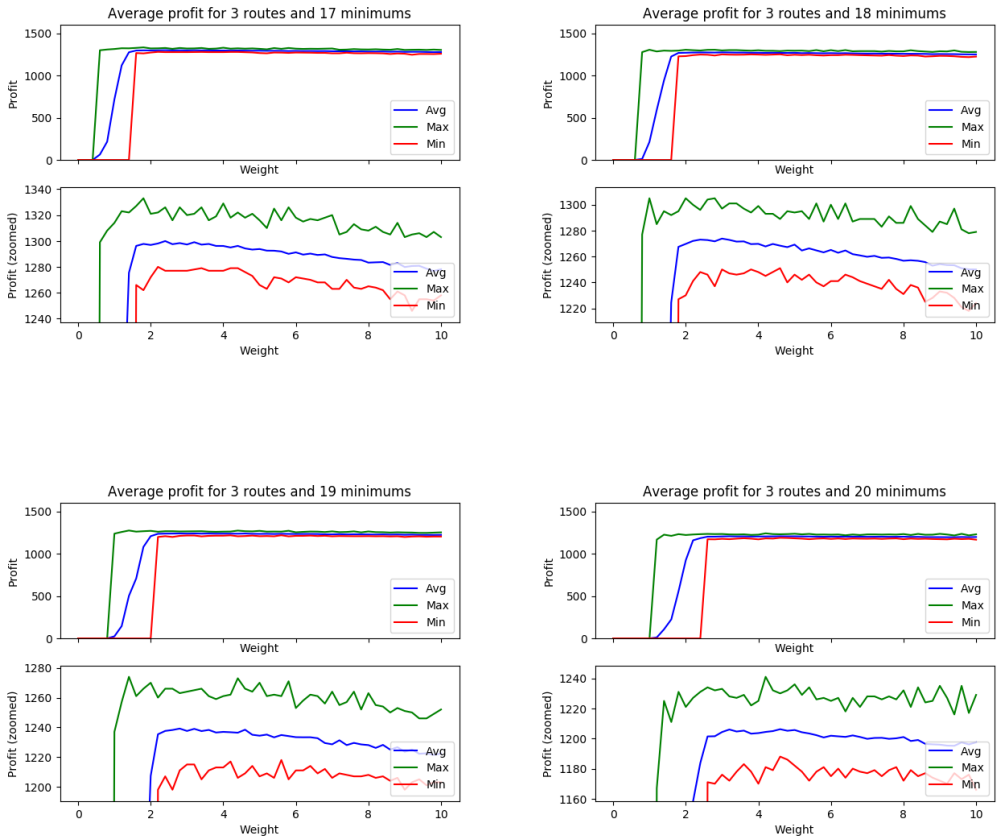
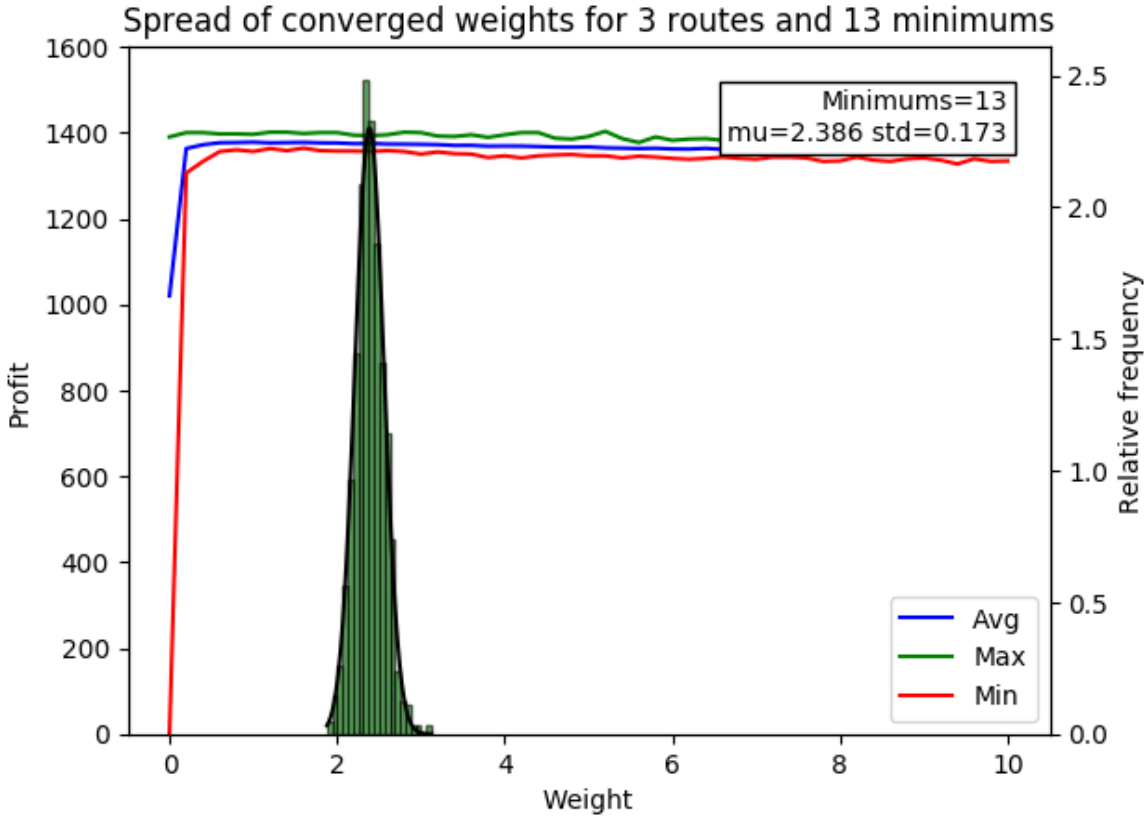
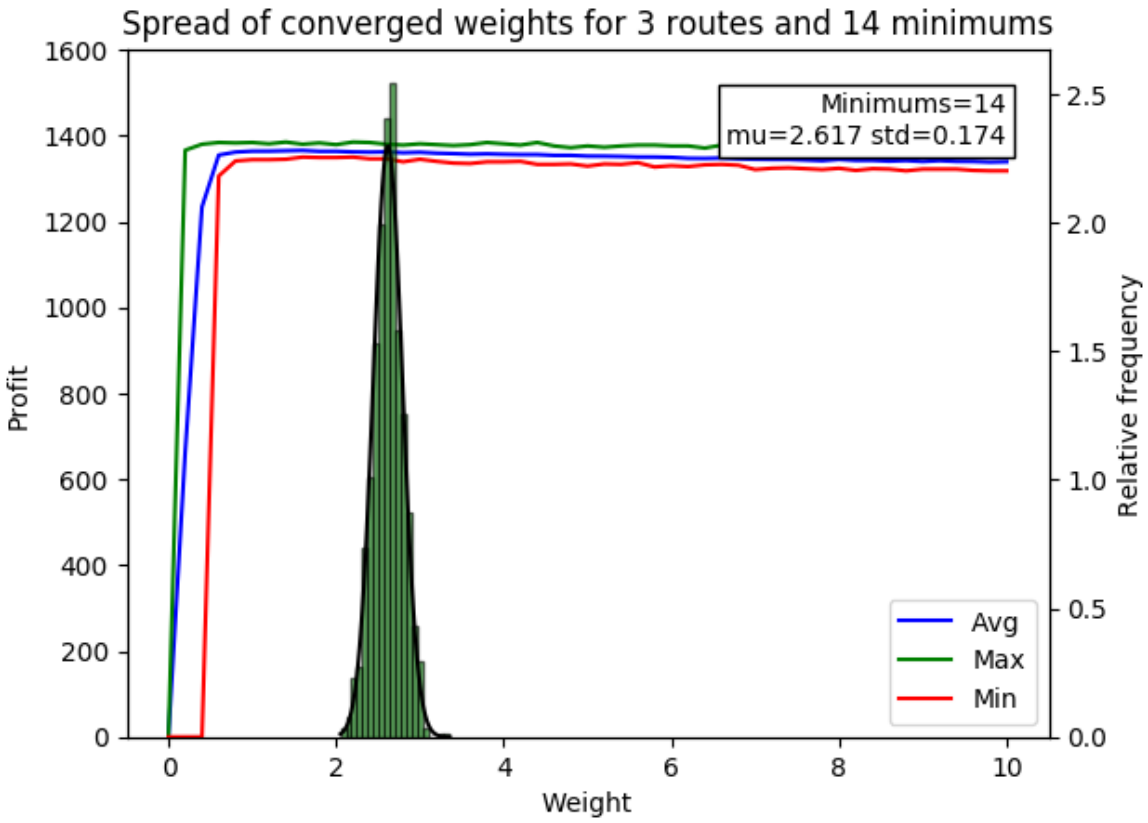
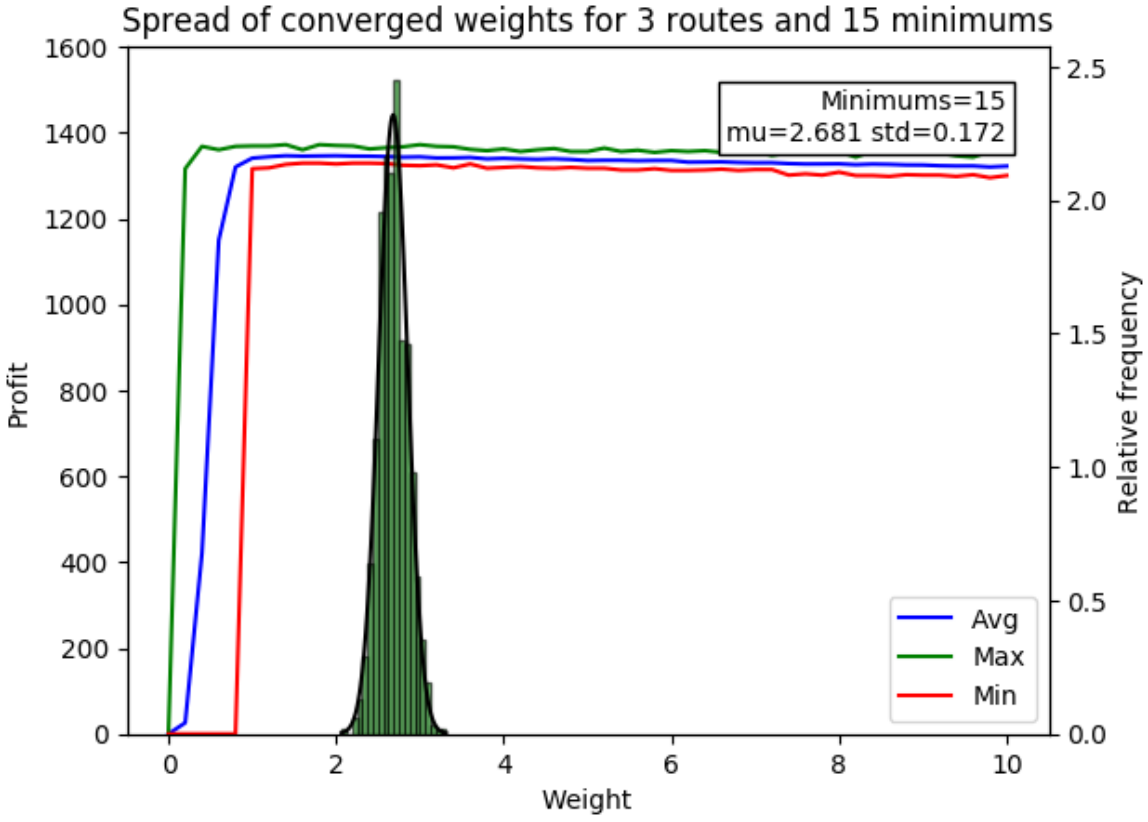
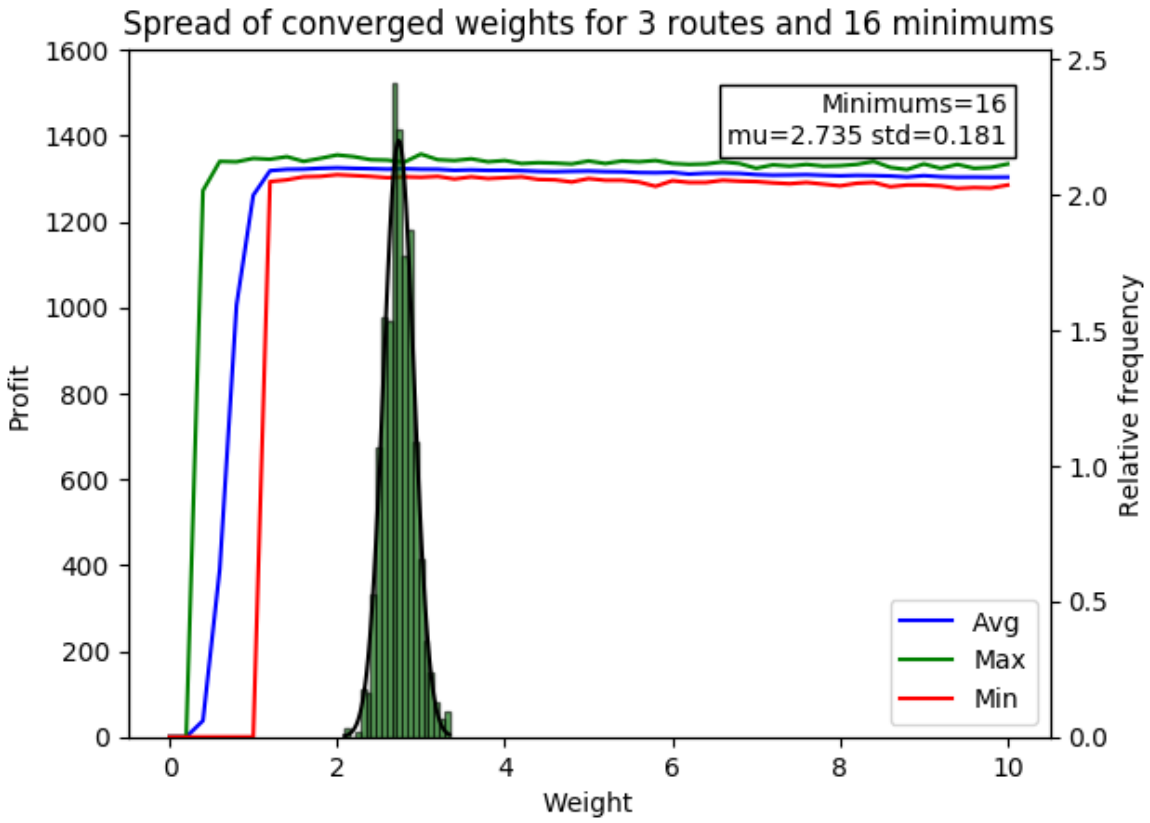


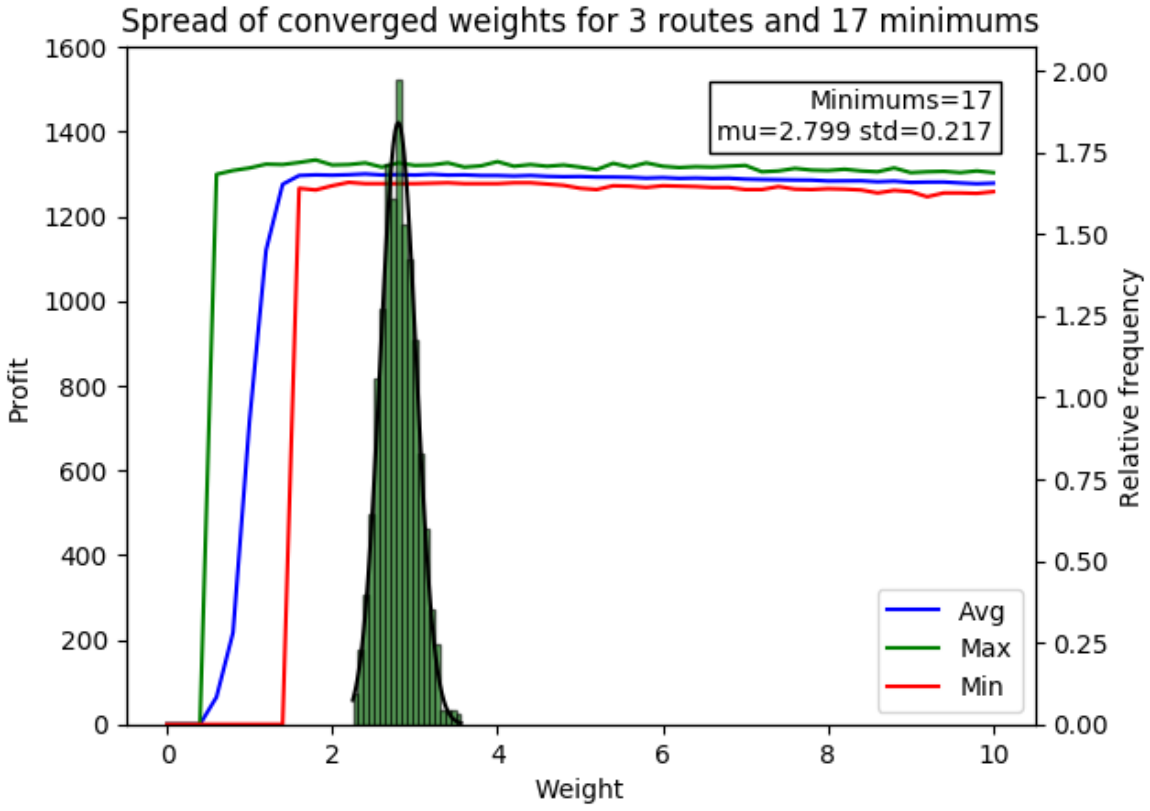
Figure B.27: Profits relative to learned weight for 13–20 minimums of category '1' for 3 routes

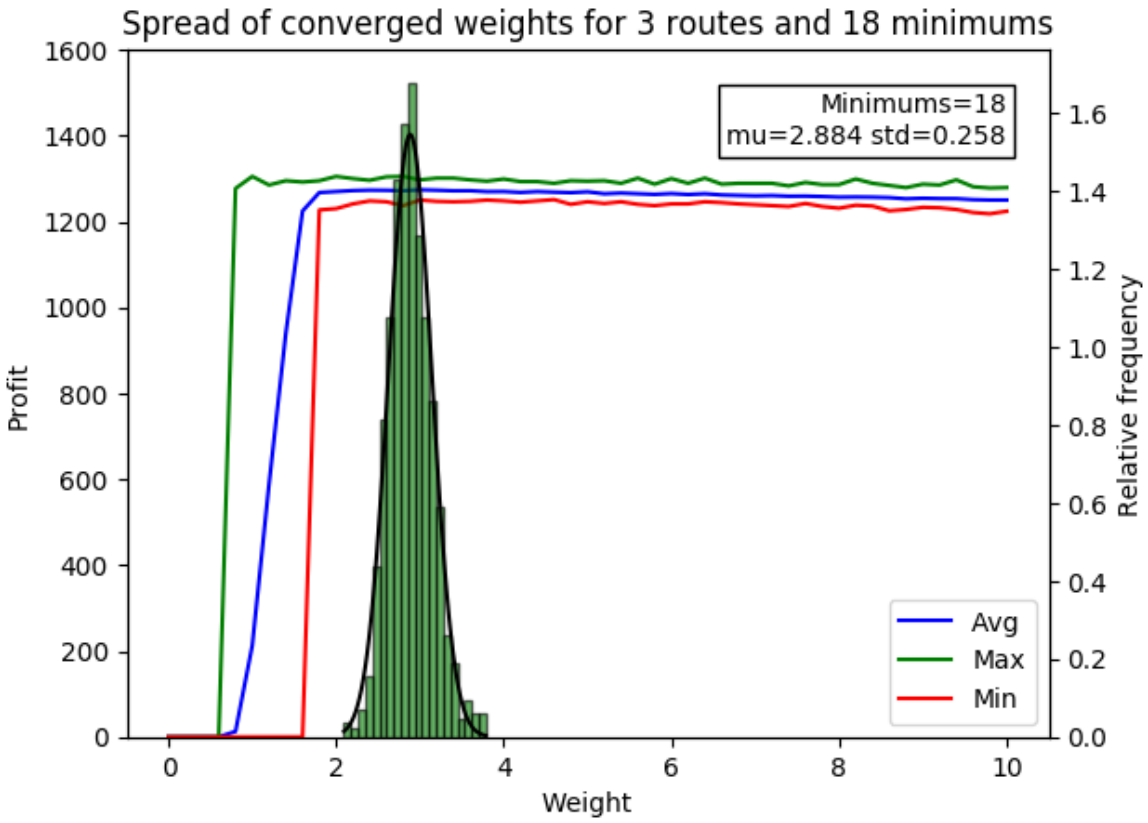


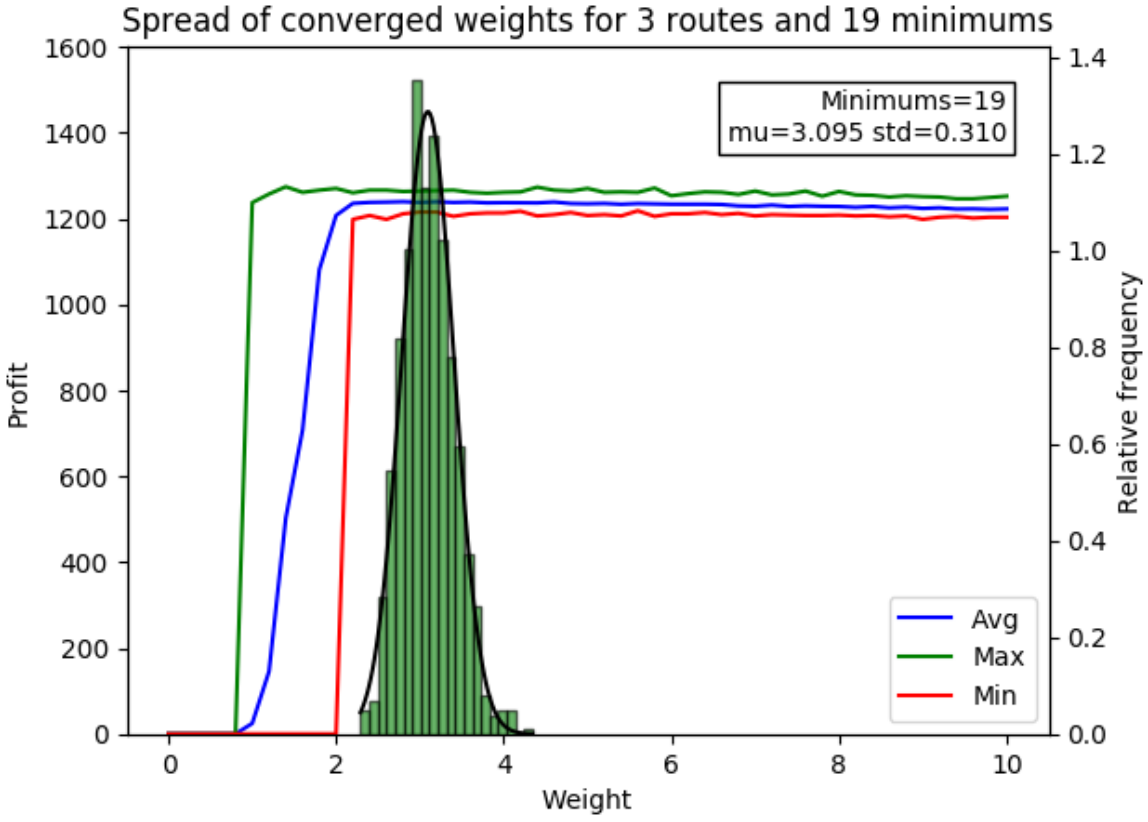

Spread of converged weights for 3 routes and 14 minimums
Minimums=14
mu=2.617 std=0.174
Avg
Max
Min
Profit
Relative frequency
Weight
0
200
400
600
800
1000
1200
1400
1600
0.0
0.5
1.0
1.5
2.0
2.5
0
2
4
6
8
10

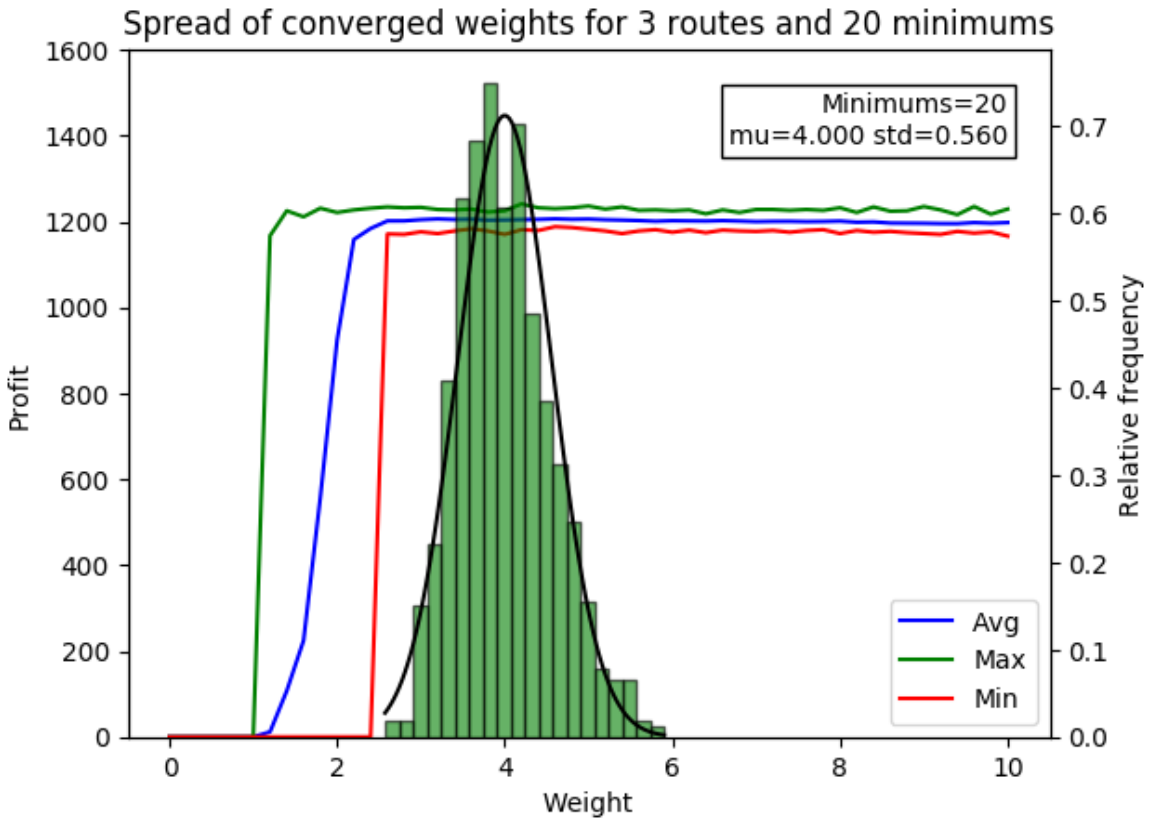

Spread of converged weights for 3 routes and 15 minimums
Minimums=15
mu=2.681 std=0.172
Avg
Max
Min
Profit
Relative frequency
Weight
0
200
400
600
800
1000
1200
1400
1600
0.0
0.5
1.0
1.5
2.0
2.5
0
2
4
6
8
10


Spread of converged weights for 3 routes and 16 minimums
Minimums=16
mu=2.735 std=0.181
Avg
Max
Min
Profit
Relative frequency
Weight
0
200
400
600
800
1000
1200
1400
1600
0.0
0.5
1.0
1.5
2.0
2.5
0
2
4
6
8
10


Spread of converged weights for 3 routes and 17 minimums
Minimums=17
mu=2.799 std=0.217
Avg
Max
Min
Profit
Relative frequency
Weight
0
200
400
600
800
1000
1200
1400
1600
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0
2
4
6
8
10


Spread of converged weights for 3 routes and 18 minimums
Minimums=18
mu=2.884 std=0.258
Avg
Max
Min
Profit
Relative frequency
Weight


Spread of converged weights for 3 routes and 19 minimums
Minimums=19
mu=3.095 std=0.310
Avg
Max
Min
Profit
Relative frequency
Weight


Spread of converged weights for 3 routes and 20 minimums
Minimums=20
mu=4.000 std=0.560
Avg
Max
Min
Profit
Relative frequency
Weight
0
200
400
600
800
1000
1200
1400
1600
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
2
4
6
8
10

Bibliography

Rahim A Abbaspour and Farhad Samadzadegan. Time-dependent personal tour planning and scheduling in metropolises. *Expert Systems with Applications*, 38(10):12439–12452, 2011.

Claudia Archetti, Alain Hertz, and Maria Grazia Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, 2007.

Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim. A memetic algorithm for the team orienteering problem. *4or*, 8(1):49–70, 2010.

Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4or*, 5(3):211–230, 2007.

Ann M Campbell, Michel Gendreau, and Barrett W Thomas. The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186(1):61–81, 2011.

Vicente Campos, Rafael Martí, Jesús Sánchez-Oro, and Abraham Duarte. Grasp with path relinking for the orienteering problem. *Journal of the Operational Research Society*, 65(12):1800–1813, 2014.

I Chao et al. Algorithms and solutions to multi-level vehicle routing problems. 1993.

I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996a.

I-Ming Chao, Bruce L Golden, and Edward A Wasil. A fast and effective heuristic for the orienteering problem. *European journal of operational research*, 88(3):475–489, 1996b.

Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8(3):23, 2012.

Tunchan Cura. An artificial bee colony algorithm approach for the team orienteering problem with time windows. *Computers & Industrial Engineering*, 74:270–290, 2014.

Duc-Cuong Dang, Rym Nesrine Guibadj, and Aziz Moukrim. A pso-based memetic algorithm for the team orienteering problem. In *European Conference on the Applications of Evolutionary Computation*, pages 471–480. Springer, 2011.

Duc-Cuong Dang, Racha El-Hajj, and Aziz Moukrim. A branch-and-cut algorithm for solving the team orienteering problem. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 332–339. Springer, 2013a.

Duc-Cuong Dang, Rym Nesrine Guibadj, and Aziz Moukrim. An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332–344, 2013b.

Irina Dolinskaya, Zhenyu Edwin Shi, and Karen Smilowitz. Adaptive orienteering problem with stochastic travel times. *Transportation Research Part E: Logistics and Transportation Review*, 109: 1–19, 2018.

Daniel Duque, Leonardo Lozano, and Andrés L Medaglia. Solving the orienteering problem with time windows via the pulse framework. *Computers & Operations Research*, 54:168–176, 2015.

Lanah Evers, Kristiaan Glorie, Suzanne Van Der Ster, Ana Isabel Barros, and Herman Monsuur. A two-stage approach to the orienteering problem with stochastic weights. *Computers & Operations Research*, 43:248–260, 2014.

Dominique Feillet, Pierre Dejax, and Michel Gendreau. Traveling salesman problems with profits. *Transportation science*, 39(2):188–205, 2005.

João Ferreira, Artur Quintas, José A Oliveira, Guilherme AB Pereira, and Luis Dias. Solving the team orienteering problem: Developing a solution tool using a genetic algorithm approach. In *Soft Computing in Industrial Applications*, pages 365–375. Springer, 2014.

Matteo Fischetti, Juan Jose Salazar Gonzalez, and Paolo Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998.

Fedor V Fomin and Andrzej Lingas. Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, 83(2):57–62, 2002.

Luca Maria Gambardella, Roberto Montemanni, and Dennis Weyland. An enhanced ant colony system for the sequential ordering problem. In *Operations Research Proceedings 2011*, pages 355–360. Springer, 2012.

Ander Garcia, Olatz Arbelaitz, Pieter Vansteenwegen, Wouter Souffriau, and Maria Teresa Linaza. Hybrid approach for the public transportation time dependent orienteering problem with time windows. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 151–158. Springer, 2010.

Ander Garcia, Pieter Vansteenwegen, Olatz Arbelaitz, Wouter Souffriau, and Maria Teresa Linaza. Integrating public transportation in personalised electronic tourist guides. *Computers & Operations Research*, 40(3):758–774, 2013.

Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Yiannis Tasoulas. Cluster-based heuristics for the team orienteering problem with time windows. In *International Symposium on Experimental Algorithms*, pages 390–401. Springer, 2013.

Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328, 2014a.

Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. Efficient heuristics for the time dependent team orienteering problem with time windows. In *International Conference on Applied Algorithms*, pages 152–163. Springer, 2014b.

Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, 62:36–50, 2015.

Zong Woo Geem, Chung-Li Tseng, and Yongjin Park. Harmony search for generalized orienteering problem: best touring in china. pages 741–750, 2005.

Michel Gendreau, Gilbert Laporte, and Frederic Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32(4):263–273, 1998a.

Michel Gendreau, Gilbert Laporte, and Frédéric Semet. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106(2-3):539–545, 1998b.

Bruce Golden, Larry Levy, and Roy Dahl. Two generalizations of the traveling salesman problem. *Omega*, 9(4):439–441, 1981.

Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval research logistics*, 34(3):307–318, 1987.

Aldy Gunawan, Zhi Yuan, and Hoong Chuin Lau. A mathematical model and metaheuristics for time dependent orienteering problem. PATAT, 2014.

Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. An iterated local search algorithm for solving the orienteering problem with time windows. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 61–73. Springer, 2015a.

Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. Sails: hybrid algorithm for the team orienteering problem with time windows. 2015b.

Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. Well-tuned ils for extended team orienteering problem with time windows. In *LARC Technical Report Series*. Singapore Management University, 2015c.

Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

Qian Hu and Andrew Lim. An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2):276–286, 2014.

Taylan Ilhan, Seyed MR Iravani, and Mark S Daskin. The orienteering problem with stochastic profits. *Iie Transactions*, 40(4):406–421, 2008.

Marisa G Kantor and Moshe B Rosenwein. The orienteering problem with time windows. *Journal of the Operational Research Society*, 43(6):629–635, 1992.

Liangjun Ke, Claudia Archetti, and Zuren Feng. Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3):648–665, 2008.

Liangjun Ke, Laipeng Zhai, Jing Li, and Felix TS Chan. Pareto mimic algorithm: An approach to the team orienteering problem. *Omega*, 61:155–166, 2016.

Morteza Keshtkaran, Koorush Ziarati, Andrea Bettinelli, and Daniele Vigo. Enhanced exact solution methods for the team orienteering problem. *International Journal of Production Research*, 54(2): 591–601, 2016.

Nacima Labadie, Jan Melechovskỳ, and Roberto Wolfler Calvo. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics*, 17(6):729–753, 2011.

Nacima Labadie, Renata Mansini, Jan Melechovskỳ, and Roberto Wolfler Calvo. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15–27, 2012.

Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete applied mathematics*, 26(2-3):193–207, 1990.

Hoong Chuin Lau, William Yeoh, Pradeep Varakantham, Duc Thien Nguyen, and Huaxing Chen. Dynamic stochastic orienteering problems for risk-aware applications. *arXiv preprint arXiv:1210.4874*, 2012.

Adrienne C Leifer and Moshe B Rosenwein. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research*, 73(3):517–523, 1994.

Jin Li. Research on team orienteering problem with dynamic travel times. *JSW*, 7(2):249–255, 2012.

Yun-Chia Liang, Sadan Kulturel-Konak, and Min-Hua Lo. A multiple-level variable neighborhood search approach to the orienteering problem. *Journal of Industrial and Production Engineering*, 30 (4):238–247, 2013.

Shih-Wei Lin and F Yu Vincent. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94–107, 2012.

R Mansini, M Wolfler Pelizzari, and R Wolfler Calvo. A granular variable neighborhood search for the tour orienteering problem with time windows. Technical report, Technical Report of the Department of Electronics for Automation, University of Brescia, 2006.

Yannis Marinakis, Michael Politis, Magdalene Marinaki, and Nikolaos Matsatsinis. A memetic-grasp algorithm for the solution of the orienteering problem. In *Modelling, Computation and Optimization in Information Systems and Management Sciences*, pages 105–116. Springer, 2015.

Rafael Martinelli, Diego Pecin, Marcus Poggi, and Humberto Longo. A branch-cut-and-price algorithm for the capacitated arc routing problem. In *International Symposium on Experimental Algorithms*, pages 315–326. Springer, 2011.

Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

Roberto Montemanni and Luca Maria Gambardella. An ant colony system for team orienteering problems with time windows. *Foundation Of Computing And Decision Sciences*, 34(4):287, 2009.

Shanthi Muthuswamy and Sarah Lam. Discrete particle swarm optimization for the orienteering problem. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 18(2), 2011a.

Shanthi Muthuswamy and Sarah S Lam. Discrete particle swarm optimization for the team orienteering problem. *Memetic Computing*, 3(4):287–303, 2011b.

V Papapanagiotou, R Montemanni, and LM Gambardella. Objective function evaluation methods for the orienteering problem with stochastic travel and service times. *Journal of applied Operational research*, 6(1):16–29, 2014.

Jesse Pietz and Johannes O Royset. Generalized orienteering problem with resource dependent rewards. *Naval Research Logistics (NRL)*, 60(4):294–312, 2013.

R Ramesh, Yong-Seok Yoon, and Mark H Karwan. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4(2):155–165, 1992.

Ram Ramesh and Kathleen M Brown. An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2):151–165, 1991.

Giovanni Righini and Matteo Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191–1203, 2009.

AİŞE ZÜLAL ŞEVKLİ and FATİH ERDOĞAN SEVİLGEN. Stpso: Strengthened particle swarm optimization. *Turkish Journal Of Electrical Engineering & Computer Sciences*, 18(6):1095–1114, 2010.

Zülal Sevkli and F Erdogan Sevilgen. Discrete particle swarm optimization for the orienteering problem. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

John Silberholz and Bruce Golden. The effective application of a new approach to the generalized orienteering problem. *Journal of Heuristics*, 16(3):393–415, 2010.

Wouter Souffriau, Pieter Vansteenwegen, Joris Vertommen, Greet Vanden Berghe, and Dirk Van Oudheusden. A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, 22(10):964–985, 2008.

Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. A path relinking approach for the team orienteering problem. *Computers & operations research*, 37(11): 1853–1859, 2010.

Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1):53–63, 2013.

Kadri Sylejmani, Jiirgen Dorn, and Nysret Musliu. A tabu search approach for multi constrained team orienteering problem and its application in touristic trip planning. In *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*, pages 300–305. IEEE, 2012.

Tommy Thomadsen and Thomas K Stidsen. The quadratic selective travelling salesman problem. Technical report, 2003.

Fabien Tricoire, Martin Romauch, Karl F Doerner, and Richard F Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37 (2):351–367, 2010.

Theodore Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809, 1984.

Pieter Vansteenwegen and Dirk Van Oudheusden. The mobile tourist guide: an or opportunity. *OR insight*, 20(3):21–27, 2007.

Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European journal of operational research*, 196(1):118–127, 2009a.

Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009b.

Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Metaheuristics for tourist trip planning. In *Metaheuristics in the service industry*, pages 15–31. Springer, 2009c.

Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

Pradeep Varakantham and Akshat Kumar. Optimization approaches for solving chance constrained stochastic orienteering problems. In *International Conference on Algorithmic DecisionTheory*, pages 387–398. Springer, 2013.

Cédric Verbeeck, Kenneth Sörensen, E-H Aghezzaf, and Pieter Vansteenwegen. A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, 236(2):419–432, 2014.

Cédric Verbeeck, Pieter Vansteenwegen, and E-H Aghezzaf. Solving the stochastic time-dependent orienteering problem with time windows. *European Journal of Operational Research*, 255(3):699–718, 2016.

F Yu Vincent and Shih-Wei Lin. Multi-start simulated annealing heuristic for the location routing problem with simultaneous pickup and delivery. *Applied soft computing*, 24:284–290, 2014.

Xia Wang, Bruce L Golden, and Edward A Wasil. Using a genetic algorithm to solve the generalized orienteering problem. In *The vehicle routing problem: latest advances and new challenges*, pages 263–274. Springer, 2008.

Wolfgang Wörndl. Solving tourist trip design problems from a user's perspective. *Mensch und Computer 2016–Workshopband*, 2016.

Shu Zhang, Jeffrey W Ohlmann, and Barrett W Thomas. A priori orienteering with time windows and stochastic wait times at customers. *European Journal of Operational Research*, 239(1):70–79, 2014.