University of the Aegean
School of Engineering
Department of Financial and Management Engineering

Reverse logistics of solid waste and recyclable materials in urban areas

Chatziioannidis Dimitrios

Dissertation Committee:

Professor Ioannis Minis, Supervisor

Assistant Professor Vasileios Koutras, Thesis Committee Member

Assistant Professor Vasileios Zeimpekis, Thesis Committee Member

Chios, 2019

Acknowledgements

I would like to express my thanks to Professor Minis Ioannis for giving me the opportunity to complete the present diploma thesis. Also, I would like to thank my committee Dr. Vasileios Koutras, Assistant Professor at the Department of Financial & Management Engineering at the University of the Aegean and Dr. Vasileios Zeimpekis, Assistant Professor at the Department of Financial & Management Engineering at the University of the Aegean for their time and advising.

I am grateful to all members of the Design, Operations & Production Systems Lab (DeOPSys) of the Financial and Management Engineering (FME) Department of the University of the Aegean, and mainly to Mrs. Lemonia Amygdalou and Christina Arampantzi.

Furthermore, I am thankful to the municipality of Chios and specially to the cleaning and recycling services for their support and the availability of information.

Εκτενής Ελληνική Περίληψη
(Summary in Greek)

Εισαγωγή

Συλλογή και μεταφορά είναι κρίσιμες διαδικασίες διαχείρισης στερεών αποβλήτων οποιασδήποτε αστικής περιοχής. Σκοπός αυτών των διαδικασιών είναι η τακτική εκκένωση των απορριμμάτων από τους κάδους και η μεταφορά τους στους προβλεπόμενους χώρους εναπόθεσης. Η διαχείριση των αποβλήτων είναι πολύ σημαντική τόσο για την υγεία όσο και για την οικονομία κάθε κοινωνίας. Μέσω της κατάλληλης διαχείρισης προστατεύεται η δημόσια υγεία και παράλληλα ενισχύεται η ανακύκλωση απορριμμάτων.

Ως ανακύκλωση απορριμμάτων ορίζεται η επαναχρησιμοποίηση, είτε μερικώς είτε ολικώς των υλικών που δεν θεωρούνται πλέον αγαθά για τον άνθρωπο. Η ανακύκλωση είναι μέρος της διαχείρισης αστικών στερεών απορριμμάτων μέσω της οποίας αποφεύγεται η ταφή ή η καύση και αντικαθίσταται με την ομαλή ένταξη των απορριμμάτων στο περιβάλλον [16].

Τα αστικά απόβλητα διακρίνονται σε οικιακά και εμπορικά. Τα οικιακά είναι αυτά που προέρχονται από νοικοκυριά ενώ τα εμπορικά προέρχονται από εργοστάσια, σφαγεία, καταστήματα, γραφεία κ.α. Οι κύριες μέθοδοι διαχείρισης απορριμμάτων τόσο στην Ελλάδα όσο και στην Ευρωπαϊκή Ένωση είναι η αποτέφρωση, η ταφή και η ανακύκλωση [17]. Η Ελλάδα το 2014 ανακυκλώνει μόλις το 19% των στερεών απορριμμάτων και βρίσκεται στις τελευταίες θέσεις της ΕΕ ως προς τη συγκεκριμένη στατιστική. Από την άλλη η Γερμανία, η Αυστρία και το Βέλγιο ήταν στις πρώτες τρεις θέσεις με ποσοστό ανακύκλωσης 64%, 56% και 55% αντίστοιχα [1].

Η διαχείριση στερεών αποβλήτων στο σύνολο της αφορά στην αποθήκευση, στην συλλογή, στη μεταφορά και στην μεταφόρτωση. Η διαδικασία αποθήκευσης αφορά στην προσωρινή αποθήκευση σε ειδικά διαμορφωμένους κάδους από πολίτες, επιχειρήσεις, νοσοκομεία ή άλλους οργανισμούς. Υπάρχουν φυσικά και απορρίμματα τα οποία χρήζουν ειδικής μεταχείρισης καθώς κρίνονται υψηλής επικινδυνότητας όπως για παράδειγμα τα τοξικά απόβλητα ενός εργοστασίου ή τα απόβλητα ενός πλοίου κ.α.

Η διαδικασία της συλλογής σχετίζεται και με τον τρόπο διαχωρισμού των υλικών λόγω φυσικών, χημικών ή άλλων ιδιοτήτων. Η εν λόγω διαχείριση εκκινεί από τα ειδικά σημεία αποκομιδής μέχρι και τη στιγμή που θα φτάσουν τα απορρίμματα σε εξειδικευμένους χώρους επεξεργασίας. Οι μέθοδοι που χρησιμοποιούνται για τη συλλογή διαφέρουν από χώρα σε χώρα.

Στην Ελλάδα υπάρχουν κάδοι γενικών απορριμμάτων και απορριμμάτων ανακύκλωσης (μπλε κάδοι) κατά μήκος των οδικών δικτύων. Σε κάποιους δήμους οι κάδοι ανακύκλωσης είναι διαχωρισμένοι σε χαρτιού (κίτρινοι κάδοι), αλουμινίου (κόκκινοι κάδοι), πλαστικού (μπλε κάδοι), οργανικών υλικών (καφέ κάδοι) και γυαλιού (καμπανοειδείς κάδοι). Η μελέτη περίπτωσης της παρούσας διπλωματικής αφορά στην τελευταία αυτή παραλλαγή των κάδων ανακύκλωσης και συγκεκριμένα στη συλλογή των πλαστικών ανακυκλώσιμων υλικών.

Η διαδικασία μεταφόρτωσης ακολουθεί τη συλλογή και ολοκληρώνει τη διαχείριση των αστικών στερεών αποβλήτων. Αφού έχουν διαχωρισθεί κατάλληλα τα υλικά απορρίμματα ανάλογα με τον τύπο τους, καταλήγουν σε ειδικούς σταθμούς μεταφόρτωσης. Οι συγκεκριμένοι χώροι είναι είτε ανοικτά είτε κλειστά και ειδικά εξοπλισμένα κτίρια. Ο εξοπλισμός συσχετίζεται με ειδικά μηχανήματα συμπίεσης των υλικών, έτσι ώστε να μειώνεται ο όγκος τους όσο το δυνατό περισσότερο και η μεταφορά τους να γίνεται ευκολότερα και πιο αποτελεσματικά [18].

Η δραστηριότητα της μεταφοράς είναι ιδιαίτερα σημαντική για τη διαχείριση των αστικών αποβλήτων. Πρέπει να προβλεφθούν πολλοί παράγοντες, όπως να αποφεύγεται κυκλοφοριακή επιβάρυνση και να καθιερώνεται ορθή συχνότητα επίσκεψης στα σημεία συγκομιδής, έτσι ώστε να αποφεύγεται η υπερχείλιση των κάδων. Επίσης, η μεταφορά από και προς τα κέντρα επεξεργασίας, πρέπει να γίνεται έγκαιρα. Τέλος, το κόστος της μεταφοράς των αποβλήτων είναι εξίσου κρίσιμο μιας και οι επισκέψεις στους κάδους συλλογής είναι τακτικές.

Τα απορριμματοφόρα που χρησιμοποιούνται στην Ελλάδα είναι είτε κλειστού είτε ανοικτού τύπου. Τα οχήματα κλειστού τύπου είναι πιο σύγχρονα και έχουν μηχανισμούς αυτόματης συμπίεσης των υλικών και μηχανισμό ανύψωσης κάδων. Στην άλλη κατηγορία ανήκουν οχήματα που συνήθως διαχειρίζονται διαφορετικού όγκου και βάρους απόβλητα. Στη μελέτη περίπτωσης της παρούσας διπλωματικής, εξετάζονται οχήματα με μηχανισμό πρέσας και μηχανισμό ανύψωσης.

Ορισμός Προβλήματος

Σκοπός της παρούσας διπλωματικής εργασίας είναι να μελετηθεί η διαδικασία της συλλογής αστικών απορριμμάτων (απορρίμματα σε δημόσιους κάδους) ενός δήμου και της μεταφοράς τους στα κατάλληλα σημεία εναπόθεσης. Πιο συγκεκριμένα, επιλύεται το Πρόβλημα Συλλογής και Μεταφοράς Αστικών Στερεών Αποβλήτων (ΠΣΜΑΣΑ). Στο πρόβλημα αυτό η ποσότητα των υλικών που συλλέγεται από ένα κάδο διαφοροποιείται από μέρα σε μέρα μέσα σε μια συγκεκριμένη χρονική περίοδο. Το γεγονός αυτής της μεταβαλλόμενης ζήτησης ανά ημέρα

στους κάδους, δεν μας επιτρέπει να εξετάσουμε το πρόβλημα ως ένα κλασσικό πρόβλημα δρομολόγησης ή Vehicle Routing Problem (VRP). Το μοντέλο της εργασίας είναι εμπνευσμένο από το Periodic VRP (PVRP) και έχει σαν στόχο να εκκενώνεται το σύνολο των κάδων μια περιοχής εντός ορισμένης περιόδου, ελαχιστοποιώντας το χιλιομετρικό κόστος. Επιβάλλεται να εξυπηρετηθούν όλα τα σημεία αποκομιδής από τα διαθέσιμα απορριμματοφόρα όσες φορές χρειαστεί, έτσι ώστε να μην υπάρξει οποιαδήποτε υπερχείλιση αποβλήτων σε δημόσιο χώρο.

Δεδομένα Προβλήματος

Για την επίλυση του προβλήματος απαιτούνται συγκεκριμένες πληροφορίες οι οποίες χρησιμοποιούνται ως είσοδοι του μοντέλου Periodic-VRP. Αυτά τα δεδομένα σχετίζονται με το οδικό δίκτυο, τον στόλο οχημάτων, τα ωράρια των οδηγών, τη ζήτηση στους κόμβους και τα χαρακτηριστικά των μέσων αποθήκευσης. Πιο συγκεκριμένα, απαιτούνται οι γεωγραφικές συντεταγμένες όλων των σημείων που υπάρχουν κάδοι απορριμμάτων, καθώς και να προσδιοριστεί ο αριθμός των κάδων ανά σημείο. Στην περίπτωση μας οι κάδοι είναι ίδιου τύπου και περιέχουν τον ίδιο τύπο υλικών. Η τοποθεσία από όπου ξεκινούν τα δρομολόγια χρησιμοποιείται και ως χώρος εναπόθεσης των υλικών που συλλέχθηκαν. Τέλος, απαιτούνται δύο ακόμα είσοδοι, η μία είσοδος αφορά τις ημέρες της περιόδου μέσα στην οποία πρέπει να εκκενωθούν οι κάδοι και η άλλη αφορά την ποσότητα υλικών αποβλήτων στους κάδους (κόμβους ου προβλήματος). Πιο συγκεκριμένα πρέπει να προσδιορισθεί η ποσότητα των αποβλήτων που συγκεντρώνεται στους εκάστοτε κάδους του κάθε σημείου, έτσι ώστε να υπολογιστεί η συχνότητα επίσκεψης από το όχημα προς τα σημεία συγκομιδής. Στη μελέτη περίπτωσης της συγκεκριμένης διπλωματικής εργασίας η συχνότητα επίσκεψης λαμβάνεται ως δεδομένο από τον δήμο Χίου.

Προσεγγιστική Επίλυση Προβλήματος

Για την επίλυση του συγκεκριμένου προβλήματος χρησιμοποιούνται δύο ευρετικοί αλγόριθμοι. Ο πρώτος εξ αυτών, Heuristic Algorithm 1 (HA1), ανήκει στην κατηγορία των αλγορίθμων εξοικονόμησης και αποτελεί παραλλαγή του γνωστού αλγορίθμου Clarke & Wright προσαρμοσμένη στο PVRP. Ο δεύτερος, Heuristic Algorithm 2 (HA2) , ανήκει στην κατηγορία των αλγορίθμων δύο φάσεων (Two Phase Algorithms)

Συγκεκριμένα, ο αλγόριθμος HA1 κατασκευάζει δρομολόγια κατ’ ευθείαν με βάση τους πίνακες εξοικονόμησης του C&W χωρίς να λαμβάνει υπόψιν τη συχνότητα επίσκεψης. Ο αλγόριθμος HA2 πρώτα ομαδοποιεί τους κόμβους με τη μεγαλύτερη συχνότητα επίσκεψης και

τους τοποθετεί μαζί στα δρομολόγια έτσι ώστε να ελαχιστοποιήσει το συνολικό αριθμό των δρομολογίων. Κατόπιν οι κόμβοι μοιράζονται στις ημέρες της περιόδου ξεκινώντας από αυτούς με την μεγαλύτερη συχνότητα επίσκεψης. Τέλος, για κάθε ημέρα εκτελείται ο C&W σαν ένα κλασσικό πρόβλημα δρομολόγησης. Στην περίπτωση που οι κόμβοι του γραφήματος με ίδια συχνότητα επίσκεψης βρίσκονται κοντά ο ένας στον άλλο, αναμένεται να αποδίδει καλύτερα ο αλγόριθμος HA2. Διαφορετικά αν οι κόμβοι με την ίδια συχνότητα επίσκεψης βρίσκονται μακριά ο ένας με τον άλλον τότε αποτελεσματικότερος αναμένεται να είναι ο HA1. Οι συγκεκριμένοι αλγόριθμοι δημιουργούν το πρόγραμμα των δρομολογίων που πρέπει να ακολουθήσουν τα απορριμματοφόρα μιας υπηρεσίας καθαρισμού (π.χ. Δήμος καθαριότητας) προκειμένου να αδειάσουν όλους τους κάδους απορριμμάτων μια περιοχής. Σκοπός είναι να συλλέξουν τα απόβλητα αυτών των κάδων και να τα μεταφέρουν στο αρχικό σημείο του δρομολογίου (depot), με αντικειμενικό στόχο την ελαχιστοποίηση του χιλιομετρικού κόστους.

Η επίσκεψη στους κόμβους πρέπει να γίνεται τόσες φορές όσες ορίζει η συχνότητα επίσκεψης των σημείων συγκομιδής και η οποία προκύπτει με βάση την ποσότητα απορριμμάτων στους κάδους του εκάστοτε σημείου. Η συχνότητα επίσκεψης για κάθε σημείο πραγματοποιείται εντός μίας συγκεκριμένης περιόδου, η οποία έχει οριστεί στη μία εβδομάδα. Οι περιορισμοί της χωρητικότητας και της βάρδιας των οχημάτων γίνεται βάσει δύο συγκεκριμένων παραδοχών.

Παραδοχή 1. Η χωρητικότητα του οχήματος μετριέται σε κάδους και υπολογίζεται ως εξής: **χωρητικότητα οχήματος = n κάδοι * μέσο περιεχόμενου κάδου**

Παραδοχή 2. Η βάρδια των οχημάτων μετριέται σε κάδους ανά όχημα και υπολογίζεται ως εξής: **βάρδια οχήματος = m κάδοι * μέσο χρόνο συλλογής/εκκένωσης κάδου**

Κάθε απορριμματοφόρο από το στόλο έχει την ίδια χωρητικότητα και την ίδια βάρδια. Επομένως, όταν το όχημα φτάσει αυτό το όριο της παραδοχής 1 οφείλει να επιστρέψει στο αμαξοστάσιο (depot) ανεξάρτητα από την πληρότητα διαθέσιμου χώρου προσωρινής αποθήκευσης και μεταφοράς των απορριμμάτων (καρότσα). Για παράδειγμα, αν ένα όχημα έχει χωρητικότητα 150 κάδους και η ημερήσια βάρδια έχει ορισθεί επίσης στους 150 κάδους, τότε κάθε φορά που το όχημα θα φθάνει την μέγιστη πληρότητά του, τότε θα επιστρέφει στο χώρο ομαδοποίησης και συλλογής των απορριμμάτων (depot) αδειάζοντας τα απόβλητα που συγκέντρωσε, ολοκληρώνοντας παράλληλα την ημερήσια βάρδια. Πρέπει να τονισθεί σε αυτό το σημείο ότι εάν είναι απαραίτητο ένα όχημα μπορεί να πραγματοποιήσει πολλές βάρδιες σε μία ημέρα.

Οι εφαρμογές των αλγορίθμων πραγματοποιήθηκαν σε τρεις φάσεις: σε ένα απλό παράδειγμα που αποτελείται από εννιά κόμβους όπου η επίλυση γίνεται δια χειρός ώστε να

κατανοηθούν τα βήματα των αλγορίθμων. Στη συνέχεια, γίνεται εφαρμογή σε τυχαία παραδείγματα μεγάλου μεγέθους με χρήση γεννήτριας δεδομένων για να δημιουργηθεί μια πιο ρεαλιστική περίπτωση. Τέλος, γίνεται εφαρμογή σε πραγματικό παράδειγμα, στο δήμο της Χίου η οποία αφορά τη συλλογή των ανακυκλώσιμων πλαστικών υλικών. Αφού εντοπιστούν οι τελικές λύσεις από τους αλγορίθμους, χρησιμοποιούνται οι μέθοδοι βελτιστοποίησης Improvement Heuristic (2-opt, VRP-exchange) που αναλύονται στο 4^{ο} κεφάλαιο για περεταίρω μείωση του κόστους.

Τα αποτελέσματα έδειξαν ότι ο αλγόριθμος HA1 αποδίδει καλύτερα όσον αφορά το χιλιομετρικό κόστος στις 7 από τις 8 εφαρμογές που πραγματοποιήθηκαν. Η μόνη περίπτωση στην οποία ο αλγόριθμος HA2 παράγει αποτελέσματα με καλύτερο κόστος είναι η πρώτη εφαρμογή που ο συνολικός αριθμός κόμβων είναι χαμηλός (50 κόμβοι). Όσο αυξάνεται ο αριθμός των κόμβων σταδιακά αποδίδει καλύτερα ο HA1.

Μελέτη Περίπτωσης

Στη συγκεκριμένη διπλωματική εργασία πραγματοποιείται εφαρμογή των αλγορίθμων σε πραγματικό πρόβλημα (practical case). Αναλυτικότερα, μελετήθηκε η περίπτωση μεταφοράς πλαστικών ανακυκλώσιμων αποβλήτων του δήμου Χίου, όπου η δρομολόγηση των απορριμματοφόρων γίνεται εμπειρικά. Οι κάδοι απορριμμάτων εκτείνονται σε όλο το νησί, όμως σε καθημερινή βάση η διαδικασία εκκένωσης τους από τα απορριμματοφόρα γίνεται στην πρωτεύουσα του νησιού η οποία χαρακτηρίζεται από υψηλή παραγωγή απορριμμάτων. Οι υπόλοιπες περιοχές χαρακτηρίζονται από χαμηλή παραγωγή αποβλήτων και ικανοποιούνται περιστασιακά. Σημειώνεται ότι η μελέτη περίπτωσης της διπλωματικής εργασίας επικεντρώνεται μόνο στις περιοχές υψηλής παραγωγικότητας.

Τα πρώτα βήματα για το μοντέλο του PVRP είναι ο υπολογισμός της συχνότητας των επισκέψεων στους κόμβους καθώς και η περίοδος μέσα στην οποία πρέπει να ικανοποιηθούν οι συχνότητες επίσκεψης. Για τον υπολογισμό της συχνότητας απαιτούνται ιστορικά στοιχεία τα οποία αντιπροσωπεύουν τη ζήτηση στους κόμβους. Με αυτόν τον τρόπο μπορούμε να υπολογίσουμε τη συχνότητα επίσκεψης στα σημεία συγκομιδής στη χρονική περίοδο που ορίσαμε στις 6 ημέρες.

Η διαλογή των ανακυκλώσιμων υλικών γίνεται ξεχωριστά ανάλογα με το είδος του υλικού (χαρτί, πλαστικό γυαλί, αλουμίνιο). Τόσο οι γεωγραφικές θέσεις όσο και ο αριθμός των κάδων ανά σημείο διαφέρουν σε κάθε υλικό. Ο αριθμός των κόμβων στη μελέτη περίπτωσης της διπλωματικής

αποτελείται από 116 σημεία και ο αριθμός των κάδων με τα πλαστικά υλικά επί αυτών των σημείων είναι 171.

Ο στόλος των οχημάτων αποτελείται από τρία απορριμματοφόρα τύπου πρέσας, με χωρητικότητα 8, 12 και 16 κυβικών λίτρων αντίστοιχα. Τα οχήματα είναι σχεδιασμένα για να μεταφέρουν χαρτί, πλαστικό και αλουμίνιο, δεν μπορούν όμως να μεταφέρουν γυάλινα απόβλητα. Κάθε όχημα εξυπηρετεί αποκλειστικά ένα ανακυκλώσιμο υλικό σε μία εβδομάδα. Ο δήμος Χίου έχει αναθέσει τη συλλογή του γυαλιού σε ιδιώτες καθώς δεν έχει αγοράσει όχημα κατάλληλο για αυτή τη περίσταση. Η μεταφορά γυαλιού, χαρτιού αλλά και αυτή του αλουμινίου δεν έχουν υπολογιστεί στην παρούσα διπλωματική. Η μελέτη της μεταφοράς τους όμως, είναι παρόμοια με αυτή του πλαστικού που αναλύεται στα παρακάτω κεφάλαια.

Καθότι η μελέτη περίπτωσης αφορά 116 σημεία με συνολικά 171 κάδους επιλέχτηκε να εκτελεστεί ο αλγόριθμος HA1. Τόσο η εφαρμογή του HA1, όσο και η μετέπειτα εφαρμογή των αλγορίθμων περαιτέρω βελτιστοποίησης μείωσαν το χιλιομετρικό κόστος της μελέτης περίπτωσης του δήμου Χίου.

Abstract

The collection and transportation of waste from an urban area is a significant topic for any municipality worldwide. If this process works under an orderly and efficient way, it will have positive effects such as environmental protection, public hygiene, fuel savings for the fleet of collection trucks and financial savings for the municipalities. A major issue of this operation is the total distance that the collection trucks need to cover in order to empty the bins of waste. For instance, in the USA it is estimated that more than 60,000 collection trucks [20] are used to execute routes of related operations on an annual basis. Moreover, the average annual distance for every vehicle is estimated to be 40,000 kilometers [21]. Actually, this means that in USA more than 2,400,000,000 kilometers are covered annually to collect the accumulated tons of waste. This thesis proposes a solution for this important logistics problem by minimizing the covered distance (cost) for a fleet of collection trucks.

The following chapters present and propose methods for the collection and transportation of waste, which is a subprocess of Solid Waste Management (SWM). The approach followed in this thesis has been inspired by the Periodic Vehicle Routing Problem (PVRP). The objective is to develop an appropriate model for a fleet of vehicles, which empties the bins of an urban area as necessary during a whole period of *m* days, minimizing the distance cost.

To solve the PVRP mathematical model, two heuristic approaches have been developed: Heuristic Algorithm 1 (HA1) and Heuristic Algorithm 2 (HA2). HA1 involves simultaneously deciding which customers (bins of waste in our problem) will be served (emptied) in each day of the week, which vehicle will be used, and in which sequence. HA1 is a variant of the savings algorithm Clarke & Wright adapted to the characteristics of the PVRP model. More specifically, heuristic HA1 executes the Clarke & Wright method as many times as needed throughout the period until the number of visits (frequency) in each node is met.

Algorithm HA2 first decides which customers (bins of waste in our problem) will be served (emptied) in each day of the week, by which vehicle and in which sequence. The characteristic of this algorithm is that it uses (empirical) classifications which minimize the total number of routes. More extensively, the algorithm first classifies the nodes according to their frequency and then assigns the nodes to days using PVRP. Secondly, Clarke & Wright is executed in each day of the period in order to route the assigned nodes.

In our model, algorithms HA1 and HA2 must follow three constraints which are associated with the schedule of visits, the capacity of the vehicle and the daily truck work-shift.

The schedule of visits is based on three parameters. First, a node must be visited as many times as its visiting frequency defines through the days of the set period. Secondly, every point cannot be visited more than once in a single day. Finally, if a point has frequency of more than one, it cannot not be visited in two consecutive days according to the Well Balance Schedule scenario (WBS). The WBS scenario is analyzed in the following paragraphs and its aim is to prevent waste overflowing from bins.

The constraints of capacity and time are made by the following two assumptions: Assumption 1: The capacity of a vehicle is measured in number of bins and computed as truck's capacity = n bins * average waste amount per bin.

Assumption 2: The shift of a vehicle is measured in a limited number of bins per shift and computed as truck's work-shift = m bins * average evacuation time per bin.

The developed algorithms are initially shown in two small instances for a better understanding of the mathematical model. Eight applications with a range of 150 to 1400 nodes are then implemented 50 times each to compare both heuristics. In the last chapter, our PVRP inspired model is developed and solved for a practical case concerning the municipality of Chios. Finally, algorithms HA1 and HA2 are used in combination with the appropriate local search improvement heuristics for further reduction of the initial solution.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
HA1	Heuristic Algorithm 1
HA2	Heuristic Algorithm 2
IH	Improvements Heuristics
VRP	Vehicle Routing Problem
PVRP	Periodic Vehicle Routing Problem
MoC	Municipality of Chios
C&W	Clarke and Wright
assum1	Assumption 1
assum2	Assumption 2

1. Introduction

1.1 Scope of the thesis

Waste is defined as any material or product that is no longer fit for human use. Urban waste is divided into four main categories namely: industrial (waste from factories), non-residential urban (construction and hospital waste), rural (farm waste), and residential-urban (housekeeping litter). The amount of global waste produced in 2010 was 1.3 billion tons and by 2025 this is expected to grow to 2.2 billion tons [8].

The basic methods of waste management in Greece and the European Union are incineration, burial and recycling. In 2014 Greece recycled only 19% of total solid waste and was ranked low in the European Union. On the other hand, Germany, Austria and Belgium held the top three positions with a recycling rate of 64%, 56% and 55% respectively [1].

Solid waste management consists of four main phases; waste generation, storage at source (e.g. recycling bins), collection and transport by waste trucks, and treatment in a transfer station or a disposal site [3]. This work deals with the process of urban waste transport and includes a case study related to the collection of plastic material by the municipality of Chios. Waste transport is defined as the process of collecting and transferring waste materials from the collection points to the appropriate processing centers [3].

Solid waste management is a problem of high importance to every society on a global scale. Consequently, transporting millions of tons of waste through public roads under safe conditions is an important part of the process. Several studies have been conducted on this subject, which vary according to the characteristics of each problem addressed. Some examples include waste collection with time windows [4], evacuation of garbage bins with stochastic demand [5**],** as well as waste collection with intermediate facilities and a heterogeneous fleet of vehicles [6].

Efficient management of waste transport requires frequent visits in each collection point as well as proper design of the timetables to avoid traffic congestion. This thesis deals with the development of a mathematical model for the reverse logistics problem of waste collection and transport. The purpose of the problem under consideration is to evacuate all waste collection bins within a certain time period and with minimum distance cost.

The approach in this thesis varies from relevant literature since it is inspired from periodic vehicle routing. In an urban area the total number of bins may be large, and the need of waste collection may differ from day to day. For example, in the first day of a week the bin may be full, the next day less than full, etc. A typical approach is to visit all bins (nodes) daily, but may not constitute an efficient approach, since it may entail high transport costs. The idea of PVRP is to assess how often each node must be visited during a period of M days without waste overflow and unnecessary routing costs. In this study the frequency of visits at the pick-up points is taken as given. Visits are executed in a one-week period which consists of six working days. Another problem input is the fleet of vehicles, with each vehicle undertaking a certain type of material (e.g. glass). The model of the thesis includes capacity and timing constraints which are based on two assumptions. Under the first assumption, a vehicle can carry the content of a certain number of bins and the truck's capacity can be computed by the expression *truck's capacity = n bins * average waste amount per bin.* Under the second assumption, a vehicle can collect a specific number of bins through a whole work-shift and this can be calculated by truck's *work-shift = m bins * average evacuation and transport time per bin.*

To solve the proposed mathematical model, two heuristic approaches have been developed, namely HA1 (heuristic approach 1) and HA2 (heuristic approach 2). These approaches were firstly used in two simple examples. Then they were applied in multiple, more complex examples produced by an appropriate generator, and the results were compared. Finally, the two heuristics have been applied on a practical case study representing recyclable material collection in the municipality of Chios.

1.2 Problem Description

This thesis models and solves the problem of Solid Waste Collection in urban areas inspired from the Periodic Vehicle Routing (PVRP).

The model addresses the demand per bin by respecting the frequency of emptying each bin within a certain period. The objective is to minimize the total distance traveled. Moreover, all necessary constraints of the problem are respected. These constraints have to do with the frequency of visiting each bin as well as other issues. In this thesis the connection between the problem in hand and the PVRP was made under a simple observation; the amount of waste in a bin varies. In this study the frequency per node is known. Scheduling restrictions in addition to the frequency of visiting nodes are analyzed in the following chapter.

The recycling department of Chios undertakes the collection of the following materials: plastic, paper, aluminum. All the collected units must reach the treatment center, which is located close to the depot. Furthermore, it is noted that those materials must be delivered to the center separately. For that reason, the transfer procedures are divided into three independent cases. For example, if a waste truck collects plastic, it cannot collect a different material like paper, at the same route. For example, the collection of paper is separated from the collection of plastic and aluminum. In this thesis the collection of plastic refuse bins of the municipality of Chios is considered. The number of bins is more than 100 plastic bins located in the capital as well as in surrounding districts. Geographical areas that were not considered were assumed to have negligible amounts of waste compared to the districts analyzed

1.3 Literature Review

This thesis approach is based on [9], in which the author describes the mathematical model of the Periodic-VRP**.** More specifically, the work in [9] is related to a logistics company, which is responsible for transporting toxic and nontoxic industrial waste from slaughterhouses, supermarkets and factories. Each company has separate and specially outfitted places of storage to accumulate amounts of waste. Furthermore, the levels of waste depend on the daily production which means that the daily waste amount is changing. It is not necessary to visit all nodes daily and therefore a well-organized schedule has been planned by the logistics enterprise. The idea is that some nodes in which the need for garbage collection is greater will be visited more frequently while other nodes less frequently. According to the author, the difference in the frequency of visits is what distinguishes the classic VRP from the Periodic-VRP method. The paper's aim is to create as many routes as needed so that the accumulation of industrial waste will not exceed the available place of storage. Finally, the purpose of that paper is to minimize the distance cost of the routes.

In [11] an explanation of Periodic-VRP and its differences with the periodic travel salesman problem (PTSP) are given. The PTSP is a modification of the classic TSP (travelling salesman problem) and a sub-category of the PVRP. The only similarity with the PVRP is that in each occasion, the problem's customers have an unstable demand. Also, the author explains that there are three differences between those two methods. The first refers to the available number of vehicles; in PTSP only one vehicle is available while several vehicles are available in PVRP. The other two differences have to do with the vehicle's timetables and total capacity. In PTSP there is no constraint on capacity or in time availability. In PVRP each vehicle may have limited capacity, limited time per day, or both at the same time.

In [10] one of the most popular algorithms of VRP solution, Clarke & Wright saving's method, is analyzed. This algorithm is related to the capacitated routing problem. The author in [12] mentions that the savings heuristic may be coded into two different ways which produce different results. Specifically, the first method, the "sequential" one, is easier to codify as it creates one route in each iteration. In this scenario when a route is completed, no change or intervention can be made to it and the algorithm starts to build the next feasible route. In comparison to the "sequential" variant, the "parallel" one can create and merge multiple routes. This second variant is more complex to codify but results in improved solutions.

In [13] there is a two-fold analysis: in the first part a vehicle routing problem with time windows is analyzed and in the second part there is an extended analysis of improvement heuristics. The author explains that the idea of local search is to improve an initial feasible solution for a VRP. This is achieved through appropriate changes among the stops of an existing route. These changes can be performed among the nodes of a single route or among those of multiple routes. Also, the author in [13] categorizes all intra and inter route solutions.

Reference [14] deals with the Travelling Salesman Problem (TSP) and some metaheuristic methods. Furthermore, a definition of the TSP and an explanatory analysis are given. Reference [15] takes a closer look at the Two-Phase Algorithms for solving the VRP. The first one is the Cluster First - Route Second, which classifies all nodes of a graph into clusters and then creates routes, one per cluster. Specifically, at the first node clusters are created and then for each cluster a TSP is solved. Some of the most widely known algorithms of that approach are the sweep heuristic, the heuristic of Fisher and Jaikumar etc. The second sub-category is the Route First - Cluster Second. The procedure is to connect the nodes of a graph and to create a cyclic route as a large TSP. After that, the route breaks into a few smaller routes in such way that the requirements of all constraints are satisfied.

From the existing literature, the problem that is closer to the waste collection PVRP is discussed in [9].

1.4 Thesis Structure

In chapter 2, the heuristic algorithms (HA1 & HA2) for the waste transport PVRP are set out with the corresponding steps. Chapter 3 presents two simple examples, in which HA1 and HA2 are applied and the results are compared. Chapter 4 analyzes the two categories of improvement heuristics. In chapter 5, the application of the municipality of Chios is solved by the HA1 to find

an initial feasible solution. Afterward, two methods from the local search algorithms, called 2 – opt and Exchange operator are implemented to improve the initial solution of HA1 further. 2 – opt and Exchange operator are analyzed in chapter 4, as well as additional local search algorithms are described in appendix (Appendix K).

2. Heuristic approaches for waste collection

Mathematical problems like the one under consideration are characterized by high complexity [9]. For that reason, heuristic methods such as Heuristic Algorithm 1 (HA1) and Heuristic Algorithm 2 (HA2) of this chapter result in close to optimal solutions within reasonable computation time.

Approach HA1 and approach HA2 are proposed in this chapter to solve the waste collection PVRP. The aim of HA1 and HA2 is to determine the vehicles' routes to empty the designated bins as many times as necessary within the defined period. The objective function is to minimize the total distance. The routes are implemented by a fleet of a certain capacity. A vehicle starts from the depot and after serving a certain number of bins, delivers the aggregated waste to the disposal sites. As mentioned above, the disposal site is assumed to be the depot.

In the following paragraphs all the needed criteria for the feasible insertion of nodes into routes will be analyzed. Thereafter, these criteria will be included in the steps of the proposed algorithms in order to clarify how the two heuristics are executed.

2.1 Capacity and Time (Constraints)

Definition

Capacity and time constraints are included in the model. The fleet consists of waste trucks with certain capacity and every waste truck has a specific work-shift for each single day of the period. The two restrictions of the model are defined by two assumptions. More specifically, the capacity of the waste truck is defined by the number of bins that can be collected in a single trip.:

- Assumption 1 (assum1): truck's capacity = n bins * average waste amount per bin

Also, the daily work-shift of a waste truck is estimated as the number of bins the vehicle can collect in a single work-shift. This restriction is defined by assumption 2:

- Assumption 2 (assum2): truck's work-shift = m bins * average emptying and transport time per bin

Furthermore,

- If only assumption 2 is reached:

the vehicle must return to the depot to complete its work shift even if the cabin is not fully loaded.

- If only assumption 1 is reached:
  the vehicle must return to the depot to unload the aggregated waste amount. In this case the collection truck will continue its work during the day until assumption 2 is reached.

2.2 Well Balance Schedule (WBS Constraint)

Definition

The model of this thesis is inspired by the PVRP, as every node can be visited multiple times through a certain period. An important characteristic is that the geographical points of our model consist of one or more refuse bins which can overflow at any time. Therefore, to prevent such a situation where possible and enable a more uniform visiting schedule, the WBS constraint does not allow visiting the same nodes in two consecutive days. (Note: the time period must always be greater than the visiting frequency for the PVRP method).

WBS's Characteristics

To compute the WBS constraint we need the following variables:

- Frequency of the node
- Remaining unmet days of the period
- Recorded schedule of visits

In addition to the above variables, two new variables, index1 and index2, are needed to compute the WBS scenario:

- $index1(node_i)$ → index1 takes 1 if node i has been visited in the immediately preceding day and 0 otherwise
- $index2(node_i)$ → index2 is equal to $remaining\ days\ -frequency(node_i)$

Condition for the two variables:

$$if\ index2(node_i) > 0\ \boldsymbol{AND}\ index1(node_i) = 1$$

The node cannot be inserted in the current day

$$if\ index2(node_i) \leq 0\ \boldsymbol{AND}\ index1(node_i) = 1$$

The node can be inserted in the current day

Example for the understanding of the constraint

Let's suppose a hypothetic schedule for node i. The last visit of node i was on day 2 and we are currently considering its insertion on day 3. Node i has frequency = 2, the period consists of 5 days and the remaining days are 3. The schedule has been recorded in the following table.

Days we have met		*Remaining days (unmet days = 3)*		
Day 1	Day 2	Day 3	Day 4	Day 5
	i			

Solution

Node i has been visited in the immediately preceding day (Day = 2), so index1 of node i is equal to 1

- $index1(node_i) = 1$

The remaining days are shown in the schedule (unmet days = 3)

- $index2(node_i) = remaining\ days - \ frequency(node_i) = 3 - 1 = 2$

$$if\ index2(node_i) > 0\ \boldsymbol{AND}\ index1(node_i) = 1$$

Node i must not be inserted in the current (day = 3)

So, if we initially visit node i on day 2, then we can't visit it again on day 3, but we can visit it on day 4 or day 5. By the same logic, if we initially visit node i on day 1, then we can't visit it again on day 2, but we can visit it on day 3, 4 or day 5 and so on.

Prior to presenting the proposed heuristics (HA1 and HA2) we overview the Clarke and Wright Saving algorithm, which is used in both heuristics.

2.3 Clarke & Wright (Construction Method)

The Clarke and Wright (C&W) savings algorithm was published in 1964 and is used to build initial solutions for Vehicle Routing Problems (VRP) [12].

Since this algorithm is a heuristic algorithm an optimal solution cannot be provided with certainty. The Clarke & Wright is implemented on problems with one depot from which goods

must be delivered in specific quantities to customers. The transferring of those quantities requires a fleet of vehicles, where each vehicle has a certain capacity. Every vehicle must travel a route by starting and ending at the depot and the objective function is to minimize the total distance. The concept of "savings" is expressed by merging two different routes $(i.e. 0-i-0 \; and \; 0-j-0)$ into one $(i.e. 0-i-j-0)$ for all points of a distance matrix by the formula $s_{ij} = s_{i0} - s_{j0} + s_{ij}$.

Consider the following example:

There is one depot (depot = 0), one vehicle of 100 units capacity and 5 points of certain demand. Table2.1 is the distance matrix, while Table2.2 provides the customer demand.

Table 2. 1 Table of random distances in km between five points and one depot

	0	1	2	3	4	5
0	0	28	31	20	25	34
1	28	0	21	29	26	20
2	31	21	0	38	20	32
3	20	29	38	0	30	27
4	25	26	20	30	0	25
5	34	20	32	27	25	0

Table 2. 2 Table with demands in units of waste for the points of matrix 2.1

Customer	**Demand**
1	37
2	35
3	30
4	25
5	32

The savings are calculated in the following matrix (Table 2.3)

Table 2. 3 Table of savings for the points of matrix 2.1

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	38	19	27	42
2	0	38	0	13	36	33
3	0	19	13	0	15	27
4	0	27	36	15	0	34
5	0	42	33	27	34	0

The point pairs from Table 2.3 in descending order will be:

1-5
1-2
2-4
4-5
2-5
1-4
3-5
1-3
3-4
2-3

In the parallel version nodes 1 and 5 are combined first. The next pair of the list is the 1-2 pair which can be combined with the 0-1-5-0 as node 1 is external to the path, but the combination 0-2-1-5-0 exceeds the vehicle's capacity and is ultimately rejected. Consequently, points 2 and 4 are combined in the second, parallel phase and the constructed routes will be 0-1-5-0 and 0-2-4-0. The next available saving pair is 4-5. Both 4 and 5 are external nodes in their routes but the routes can't be combined as 0-1-5-2-4-0, because this combination exceeds the vehicle's capacity. Finally, points 3 and 5 are combined, as 5 is an external node of route 0-1-5-0 and the combination 0-1-5-3-0 does not exceed the vehicle's capacity. In this way the algorithm constructs the routes 0-1-5-3-0 and 0-2-4-0 with total transportation cost amounting to 171 km.

2.4 Proposed Heuristic HA1

In heuristic HA1 the pairs with highest saving values are chosen first from the saving list of C&W to be routed. The insertion of nodes into the routes to be completed must meet the constraints of capacity, time and WBS. The process is repeated for each day of the period until the frequency of all points is exhausted. The description of the steps is made in short context, while a more detailed explanation is given in the last pages of the appendix (Appendix N). The implementation of HA1 in MATLAB is presented analytically in appendix (Appendix A, Appendix C, Appendix D, Appendix E).

The steps of HA1 to solve are the following:

Step 1. Set the M days of the period P

Step 2. Define the number of vehicles in a vector veh and the capacity of their cabin cap (constraint of capacity)

Step 3. Define the frequency of each node f_i

Step 4. Designate the total number of refuse bins in every point

Step 5. Set how many bins can a vehicle carry in a single day (constraint of work-shift)

Step 6. Compute the savings list $savings$

Step 7. **IF** all the days of the period have been met **OR** is the first repetition of the algorithm

SELECT day 1

GO to **step 8**

ELSE

SELECT the next available day

GO to **step 9**

Step 8. **IF** you have used all the vehicles from the fleet **OR** is the first repetition of day 1

SELECT the 1st vehicle of the fleet

GO to **step 9**

ELSE

SELECT the next available vehicle

GO to **step 9**

Step 9. **IF** the frequencies of all points aren't exhausted

Go to **step 10**

Elseif the frequencies of all points are exhausted

END of the algorithm

Step 10. **IF** the list of savings has NOT been scanned until the end

SELECT the next available point pair from the savings list ($savings$)

GO to **step 11**

ELSE

GO to **step 7**

Step 11. **IF** the selected pair from savings can be inserted as a new route or can be combined according to C&W steps

GO to **step 12**

ELSE

GO to **step 10**

Step 12. **IF** capacity & time constraints are followed (assum1 and assum2)

GO to **step 13**

ELSE

GO to **step 10**

Step 13. **IF** WBS constraint is followed

UPDATE the frequency of points of saving's point pair

UPDATE the capacity of the truck of the current work-shift

UPDATE the current path

UPDATE the schedule

GO to **step 10**

ELSE

GO to **step 10**

2.5 Proposed Heuristic HA2

Algorithm HA2 is similar with HA1 as it uses the same construction method (C&W) to build initial solutions. Where HA2 differs significantly, is that it first assigns the nodes to days and after that creates initial solutions. On the other hand, heuristic HA1 builds the routes and assign the nodes to days at the same time. The constraints for both algorithms are the same. The steps of HA2 are described extensively in the chapters of the appendix (Appendix O).). The implementation of HA2 in MATLAB is presented analytically in appendix (Appendix B, Appendix C, Appendix D, Appendix F, Appendix G, Appendix H).

The steps of HA2 to solve are the following:

Step 1. Divide the nodes of the problem into classes according to their frequency

Step 2. Select the cluster with the highest frequencies and sort the nodes in it in ascending order according to their distance with the depot

Step 3. Select cluster with the second highest frequency value. Sort the nodes in it, according to their distance from the last point of the cluster with the immediately smaller value of frequencies. Repeat that for the rest of the clusters.

Step 4. Merge the clusters side by side in one list beginning with nodes of clusters with highest frequency $cluster\_list$

Step 5. Set the M days of the period P

Step 6. Define the number of vehicles in a vector veh and the capacity of their cabin cap

Step 7. Define the frequency of each node f_i

Step 8. Designate the total number of refuse bins in every point

Step 9. Set how many bins can a vehicle carry in a single day (daily work-shift)

Step 10. **IF** all the days of the period have been met **OR** is the first repetition of the algorithm

SELECT day 1

GO to **step 11**

ELSE

SELECT the next available day

GO to **step 12**

Step 11. **IF** you have used all the vehicles from the fleet **OR** is the first repetition of day 1

SELECT the 1st vehicle of the fleet

GO to **step 12**

ELSE

SELECT the next available vehicle

GO to **step 12**

Step 12. **IF** the frequencies of all points aren't exhausted

Go to **step 13**

Elseif the frequencies of all points are exhausted

GO to **step 16**

Step 13. **IF** the list $cluster\_list$ has NOT been scanned until the end

SELECT the next available point from the $cluster\_list$

GO to **step 14**

ELSE

GO to **step 10**

Step 14. **IF** capacity & time constraints are followed (assum1 and assum2)

GO to **step 15**

ELSE

GO to **step 13**

Step 15. **IF** WBS constraint is followed

UPDATE the frequency of points of saving's point pair

UPDATE the capacity of the truck of the current work-shift

UPDATE the schedule

GO to **step 13**

ELSE

GO to **step 13**

EXECUTE C&W for each single day of the schedule

3. Application of HA1 and HA2 and comparison between the two heuristic

This chapter presents the application of the algorithms with two simple examples which have been solved manually, in order to clarify how the two heuristics work. HA1 and HA2 are executed in the first example (application 1) and then implemented again in the second example (application 2). The only difference between the two simple instances is the frequency per node, while all other problem parameters remain constant.

The following inputs are needed for the problem construction:

- Number of vehicles
- Number of nodes
- Number of days of the period
- Total number of bins
- Vehicle capacities in bins (assumption 1)
- Limited number of visits for the vehicles (assumption 2)
- Number of bins in each point
- Node frequency
- Coordinates of nodes
- Coordinates of depot
- Distance matrix

3.1 Application 1

The Euclidean distance method is implemented for nine random (x,y) points to build the distance matrix. The origin (0,0) and the nine points are the following: (0 0), (2 30), (5 29), (5 27), (10 13), (11 12), (12 11), (15 26), (16 25) and (17 23). Using the formula of Euclidean distance

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)},$$

the following distance matrix is produced:

Table 3. 1 Euclidean distance matrix between all (x,y) points

	0	1	2	3	4	5	6	7	8	9
0	0.00	30.07	29.43	27.46	16.40	16.28	16.28	30.02	29.68	28.60
1	30.07	0.00	3.16	4.24	18.79	20.12	21.47	13.60	14.87	16.55
2	29.43	3.16	0.00	2.00	16.76	18.03	19.31	10.44	11.70	13.42
3	27.46	4.24	2.00	0.00	14.87	16.16	17.46	10.05	11.18	12.65
4	16.40	18.79	16.76	14.87	0.00	1.41	2.83	13.93	13.42	12.21
5	16.28	20.12	18.03	16.16	1.41	0.00	1.41	14.56	13.93	12.53
6	16.28	21.47	19.31	17.46	2.83	1.41	0.00	15.30	14.56	13.00
7	30.02	13.60	10.44	10.05	13.93	14.56	15.30	0.00	1.41	3.61
8	29.68	14.87	11.70	11.18	13.42	13.93	14.56	1.41	0.00	2.24
9	28.60	16.55	13.42	12.65	12.21	12.53	13.00	3.61	2.24	0.00

In application 1 there are two available vehicles and the maximum load for each vehicle is 6 bins (assumption 1). The daily work-shift for each vehicle is also 6 bins (assumption 2). There is one depot in $(0,0)$ and nine nodes consisting of one or two bins $d = [1,1,1,1,1,1,2,2,2]$. Also, those nodes have a frequency between 1 and 4 visits in a set period of six working days $f = [1,4,4,4,3,3,3,1,1]$.

Table 3. 2 The number of bins in each node and the number of visits needed in each node (frequency)

Node	**Number of bins in each node**	**Frequency**
1	2	4
2	2	4
3	2	4
4	1	3
5	1	3
6	1	3
7	1	1
8	1	1
9	1	1

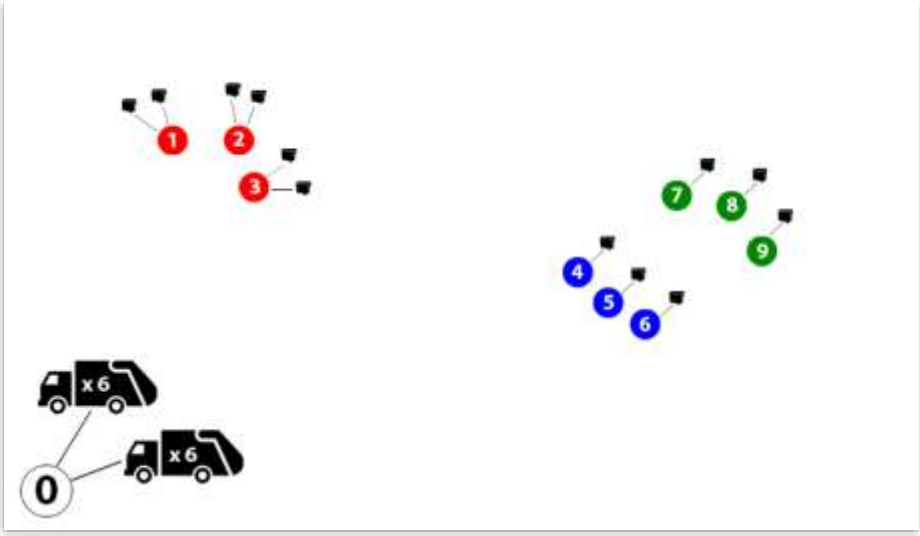
According to Clarke & Wright saving's formula which is $s_{ij} = s_{i0} - s_{j0} + s_{ij}$ the following values are calculated (Table 3.3).

Table 3. 3 Clarke & Wright saving matrix between all (x,y) points of Table3.1

	0	1	2	3	4	5	6	7	8	9
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	56.33	53.28	27.68	26.22	24.87	46.48	44.88	42.11
2	0.00	56.33	0.00	54.89	29.07	27.68	26.39	49.00	47.40	44.61
3	0.00	53.28	54.89	0.00	28.99	27.58	26.27	47.43	45.96	43.41
4	0.00	27.68	29.07	28.99	0.00	31.27	29.85	32.49	32.67	32.80
5	0.00	26.22	27.68	27.58	31.27	0.00	31.14	31.74	32.03	32.35
6	0.00	24.87	26.39	26.27	29.85	31.14	0.00	31.00	31.40	31.88
7	0.00	46.48	49.00	47.43	32.49	31.74	31.00	0.00	58.28	55.01
8	0.00	44.88	47.40	45.96	32.67	32.03	31.40	58.28	0.00	56.05
9	0.00	42.11	44.61	43.41	32.80	32.35	31.88	55.01	56.05	0.00

Figure 3.1 shows the nodes of Table3.1

Figure 3. 1 Graph illustration of nine nodes consisting of 12 bins, one depot = 0 and two waste trucks of six bins capacify



Red Nodes: frequency = 4, Blue Nodes: frequency = 3, Green Nodes: frequency = 4,

Black Nodes: bins

HA1 in Application 1

The implementation of HA1 takes as priority the pair of nodes with the highest saving from Table3.3. Initially one of the two available vehicles with capacity equal to six is chosen to be routed at the first day. Node number 7 and node number 8 is the highest saving pair and therefore considered first as $route1 = [\,0-7-8-0\,]$.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 7 – 8 – 0	2	2	2

The next pair with the highest saving is node 1 and node 2 $[\,1-2]$. This pair cannot be connected directly to route 1, as there is no common neighbor. However, according to the thesis' Clarke & Wright variation, the parallel building of routes is permitted. Consequently, neighbors $[\,1-2]$ is the second route of day 1.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 7 – 8 – 0	2	2	6
Route 2	1		0 – 1 – 2 – 0	4	4	

By the last insertion (route2) the total demand became six which actually reaches the maximum limit of bins that the waste truck can collect in a single day according to assum2. The use of parallel variation of C&W algorithm permits us to combine the [1 – 2] and the [7 - 8] into one route. Also, by this move the total demand does not exceed the truck's capacity. Thus, route 2 is deleted and the final solution for the first truck on day 1 is [0 – 1 – 2 – 7 – 8 – 0].

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 1 – 2 – 7 – 8 – 0	6	6	6

After that, the frequency of nodes [1, 2, 3, 4, 5, 6, 7, 8, 9] is updated and becomes equal to [3, 3, 4, 3, 3, 3, 0, 0, 1]. Nodes 7 and 8 will not be visited any longer due to their zero frequency, while nodes 1 and 2 will be visited three more times.

Once the constraint of work-shift is reached, we move on day 2 in order to follow the same process. More specifically, the Table of savings (Table 3.3) will be scanned again from the beginning in the same way as in day 1. At the top of Table3.3 is the [7 – 8], but this pair cannot be selected because of zero frequency. The next pair with the largest saving is [1 – 2]. To achieve a more uniform schedule, visiting nodes in two consecutive days is avoided if that is possible. To compute that we need to follow the **WBS scenario**:

Node 1 has been visited in the immediately preceding day, so index1 of node 1 is equal to 1

- $index1(node_1) = 1$

In the following computation, remaining days is the total number of unmet days of the period. The frequency refers to the current frequency of a node.

- $index2(node_1) = remaining\ days - \ frequency(node_i) = 5 - 3 = 2$

$$if\ index2(node_1) > 0\ \boldsymbol{AND}\ index1(node_1) = 1$$

Node 1 cannot be inserted

Node 2 has been visited in the immediately preceding day, so index1 of node 2 is equal to 1

- $index2(node_2) = 1$

In the following computation remaining days is the total number of the unmet days of the period. The frequency refers to the current frequency of a node.

$$index1(node_2) = remaining\ days - frequency(node_i) = 5 - 3 = 2$$

$$if\ index2(node_2) > 0\ \boldsymbol{AND}\ index1(node_2) = 1$$

Node 2 cannot be inserted

In this case both conditions are met, so the insertion of $[1 - 2]$ is invalid. Continuing in day 2 we go to the next highest saving pair in Table3.3 which is $[8 - 9]$. According to our restrictions, route $[0 - \ 8 - 9 - 0]$ is not feasible as one of the two points has frequency equal to zero. The next highest pair of nodes in Table3.3 is $[7 - 9]$, which cannot be inserted as the frequency of node 7 is zero. The next unmet pair is $[2 - 3]$ which cannot be inserted as node 2 doesn't meet the WBS scenario. The pair $[1 - 3]$ follows, but node 1 also doesn't meet the WBS scenario. The next highest pairs in Table3.3 which are $[7 - 2]$, $[7 - 3]$, $[2 - 8]$, $[3 - 8]$, [1 – 8]

are rejected because the frequency of points 7 and 8 is equal to zero. The next available pair is the [2 – 9], but node 2 doesn't meet the WBS scenario. After [2 – 9] in Table3.3 is the pair [3 – 9], which doesn't violate the constraint of frequency or the WBS scenario. Consequently, the first route of day 2 is $route\ 2 = [\ 0 - 3 - 9 - 0\ ]$.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 2	1	2	0 – 3 – 9 – 0	3	3	3

The next pair is the $[1 - 9]$ which cannot be inserted as node 1 doesn't meet the WBS scenario. After $[1 - 9]$, the pair $[4 - 9]$ follows. Node 4 is then added next to node 9 in route 2 as the route $[\ 0 - 3 - 9 - 4 - \ 0\ ]$ doesn't exceed the truck's capacity or the daily work-shift.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 2	1	2	0 – 3 – 9 – 4 – 0	4	4	4

The pairs of $[4 - 8]$ and $[4 - 7]$ cannot be combined due to the WBS scenario. The next highest pair of Table3.3 is $[4 - 5]$. Node 5 can be added next to node 4 as the constructed route doesn't violate the truck's capacity or the daily work-shift.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 2	1	2	0 – 3 – 9 – 4 – 5 – 0	5	5	5

The next permitted pair of nodes is the $[5 - 6]$ and the path of day 2 is converted to:

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 2	1	2	0 – 3 – 9 – 4 –5 – 6 – 0	6	6	6

After that, the frequency of nodes [1, 2, 3, 4, 5, 6, 7, 8, 9] is updated and becomes equal to [3, 3, 3, 2, 2, 2, 0, 0, 0] and consequently node 9 (as nodes 7 and 8) will not be visited any longer.

Once the daily work-shift of truck 1 at day 2 is reached we move on day 3. The process of days 1 and 2 is repeated. At the top of Table3.3 [7 – 8], but this pair cannot be selected because of zero frequency. The next pair with the largest saving is [1 – 2]. This time the [1 – 2] can feasibly be added at day 3, as the WBS scenario is not violated.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 3	1	3	0 – 1 – 2 – 0	4	4	4

The frequency of nodes 1 and 2 is updated and becomes equal to [2, 2]. All the next available pairs in the savings Table cannot be combined with route 3, because the WBS scenario is violated at any case. Since the saving's Table has been scanned entirely without any insertion the process is completed and we move on the next day.

Table 3.3 starts to be being scanned from the beginning for the next available day of the period which is day 4. At the top of the Table3.3 is the [7 – 8], but this pair cannot be selected because of zero frequency. The next pair with the largest saving is [1 – 2]. According to WBS, this pair cannot be inserted in day 4. The pairs [8 – 9] and [7 – 9] have at least one node of zero frequency so they can't be added as initial routes at day 4. The next feasible saving pair from Table3.3 is [4 – 5] which is set as the initial route of day 4.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 4	1	4	0 – 4 – 5 – 0	2	2	2

After the [4 – 5], pair [5 – 6] follows. Node 6 will be connected next to node 5 in route 4 as the truck's capacity and truck's work-shift are not exceeded.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 4	1	4	0 – 4 – 5 – 6 – 0	3	3	3

Pair [6 – 7] is skipped as node 7 has zero frequency. The next highest saving pair is [6 – 4], which is skipped, as both points 6 and 4 have already been added in the current route. Pair [2 – 4] is skipped since node 2 doesn't meet the WBS criterion. Continuing in Table3.3 the next available pair which is met consists of nodes 3 and 4. Node 3 is then added before node 4.

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 4	1	4	0 – 3 – 4 – 5 – 6 – 0	5	5	5

All the following pairs from Table3.3 don't meet the criteria of the problem and therefore route 4 is completed. By the same logic, we continue until the last day of the period. Consequently, two more routes will be created called route 5 and route 6 which are inserted at days 5 and 6 respectively. The results of the processes are shown in the following table.

Route No	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 1 – 2 – 7 – 8 – 0	6	6	6
Route 2	1	2	0 – 3 – 9 – 4– 5 – 6 – 0	6	6	6
Route 3	1	3	0 – 1 – 2 – 0	4	4	4
Route 4	1	4	0 – 3 – 4 – 5 – 6 – 0	5	5	5
Route 5	1	5	0 – 1 – 2 – 3 – 0	6	6	6
Route 6	1	6	0 – 1 – 2 – 3 – 0	6	6	6

As can been seen after the last day of the period (day 6), nodes 4, 5, 6 appear twice in the schedule, although the required frequency of these nodes is three. Therefore, the next available vehicle from the fleet will be chosen and we start again at day 1. Nodes 1, 2, 3, 7, 8, 9 have frequency equal to zero so the corresponding pairs in Table3.3 will be skipped. The first highest saving pair between nodes 4, 5 and 6 in Table3.3 is [4 – 5] with capacity equal to 2 bins/vehicle and work-shift equal to 2 bins/vehicle. The next available pair is [5 – 6] which will be combined with [0 – 4 – 5 – 0] to gives [0 – 4 – 5 – 6 – 0]. Therefore, the second vehicle in day 1 will execute the following route:

	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 7	2	1	0– 4 – 5 – 6 – 0	3	3	3

After the creation of Route 7 the frequency of all nodes becomes zero and the algorithm HA1 comes to an end. The final plan is shown in the following table.

Table 3. 4 The final plan of HA1 in application 1 (period = 6 day)

Route No	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 1 – 2 – 7 – 8 – 0	6	6	6
Route 7	2	1	0– 4 – 5 – 6 – 0	3	3	3
Route 2	1	2	0 – 3 – 9 – 4– 5 – 6 – 0	6	6	6
Route 3	1	3	0 – 1 – 2 – 0	4	4	4
Route 4	1	4	0 – 3 – 4 – 5 – 6 – 0	5	5	5
Route 5	1	5	0 – 1 – 2 – 3 – 0	6	6	6
Route 6	1	6	0 – 1 – 2 – 3 – 0	6	6	6

The total distance for all routes is 431 km. The first vehicle made 6 work-shifts during the period while the second vehicle made only one work-shift at day 1. So, the total number of routes is 7 through the whole period. None of the two vehicles made a double work-shift in a single day.

HA2 in Application 1

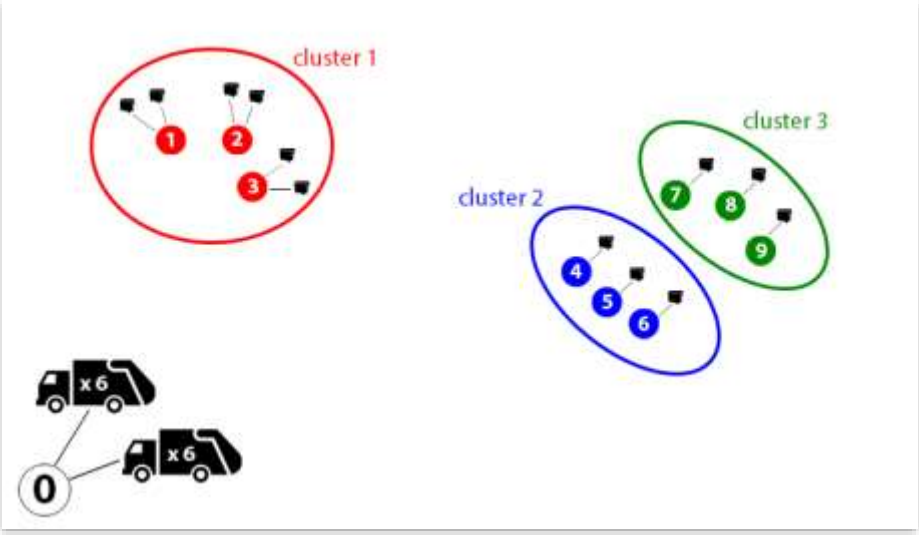
HA2 is implemented in two steps: the first part involves the assigning of nodes in the days of the period according to the problem's restrictions and the second part involves the implementation of the Clarke and Wright in each single day as a classic VRP. In contrast with HA1, HA2 gives priority to the variable of frequency of visits. The aim of this algorithm is to create as less routes as possible by creating appropriate clusters.

First Clustering of HA2

The first step is to create a cluster which contains nodes of the same frequency value. Each color on the graph (Figure 3.2) corresponds to a certain frequency: red corresponds to nodes which

will be visited four times in the week, blue to those that will be visited three times and green to those visited once per week. This classification can be seen in Figure 3.2 and Table 3.5:

Figure 3. 2 Graph illustration of classification of the nodes according to their frequency value.



Red Nodes: frequency = 4, Blue Nodes: frequency = 3, Green Nodes: frequency = 4, Black Nodes: bins in each point

Table 3. 5 Classification of the nodes according to their frequency value

Clusters	Nodes	Node's color	Node's frequency
Cluster 1	1, 2, 3	red	4
Cluster 2	4, 5, 6	blue	3
Cluster 3	7, 8, 9	green	1

Cluster 1 = [1 2 3] (red nodes), cluster 2 = [4 5 6] (blue nodes) and cluster 3 = [7 8 9] (green nodes).

Second Clustering of HA2

In the first cluster, nodes are placed in ascending order with respect to their distance from the depot; so cluster1 will be C1 = [3, 1, 2] (red nodes). The nodes of cluster 2 are also placed in ascending order, but this time in accordance with the distance from the last node of cluster 1. Thus, C2 = [4, 5, 6] (blue nodes). Finally, based on the last point of cluster 2, Cluster 3 (green nodes) becomes C3 = [9, 7, 8].

Final clustering of HA2

The next step is to merge all clusters in one cluster, cluster 4, by placing them side by side $[cluster\ 1, cluster\ 2, cluster\ 3]$. The result will take the following format: cluster 4 = [3 1 2 4 5 6 9 7 8]

Assigning nodes to days

The next procedure regards the selection of nodes one by one from cluster 4, in order to assign them to days, respecting vehicle's capacity (assum1), daily work-shift (assum2) and WBS scenario. This is done by the following steps:

Select the first available vehicle of the fleet (capacity = 6 bins/vehicle) and go to day 1. The first assignment concerns point 3 which is the first one from cluster 4.

	Vehicle	**Day**	**Assigned Nodes**	**Current capacity (number of bins) (assum1)**	**Work-shift (number of bins) (assum2)**	**Total Daily Work- Shift (assum2)**
Route 1	1	1	3	2	2	2

The next available node of cluster 4 is node 1. If we assign node 1 to day 1, the capacity and the work-shift will not be violated for that day.

	Vehicle	**Day**	**Assigned Nodes**	**Current capacity (number of bins) (assum1)**	**Work-shift (number of bins) (assum2)**	**Total Daily Work- Shift (assum2)**
Route 1	1	1	3, 1	4	4	4

The next available node of cluster 4 is node 2. The combination of nodes 3,1,2 at day 1 is feasible, as it just reaches the work-shift and capacity's limits but doesn't exceed them.

	Vehicle	**Day**	**Assigned Nodes**	**Current capacity (number of bins) (assum1)**	**Work-shift (number of bins) (assum2)**	**Total Daily Work- Shift (assum2)**
Route 1	1	1	3, 1, 2	6	6	6

The current frequencies of nodes 3, 1 and 2 are updated at 2 as they all have been used on the 1st day. Once the work-shift is reached, we go to the next available day of the period which is day 2. The first 3 points of cluster 4 are 1,2 and 3 which have already been used on day 1 and

therefore according to WBS scenario cannot be used in two days in a row. Consequently, point 4 of cluster 4 is assigned to the 2nd day.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 2	1	2	4	1	1	1

The next two nodes of cluster 4, are 5 and 6 which can feasibly be added on day 2 as the total work-shift and capacity are not exceeded.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 2	1	2	4, 5, 6	3	3	3

If we combine the remaining nodes of cluster 4 which are 9,7 and 8 with nodes 4,5 and 6 on day 2, none of the constraints will be violated.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 2	1	2	4,5,6,7,8,9	6	6	6

By the insertion of nodes 7,8 and 9, the frequency of these nodes is reduced to zero and consequently they will not be visited any longer. The constraints of work-shift and capacity are both reached and therefore we move to day 3 of the period. On day 3, nodes 1,2,3 from cluster 4 are available to be visited, while nodes 4,5,6 are not, according to WBS scenario.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 3	1	3	3, 1, 2	6	6	6

Similarly, to day 1, the insertion of nodes 1,2 and 3 from cluster 4 reaches the constraint of work-shift on day 4. According to WBS scenario, this leads us to the end of the processes for day 4.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 4	1	4	4,5,6	3	3	3

On days 5 and 6, WBS dictates that all the remaining nodes of cluster 4 are available to be visited, except nodes 7,8 and 9 which have been exhausted. So, the first 3 nodes from cluster 4 which are 1,2 and 3 will be selected to be visited on day 5.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 5	1	5	3, 1, 2	6	6	6

Furthermore, the daily work-shift of the current day is reached and therefore we move on the next available day of the period. By the last insertion, the corresponding frequencies of cluster 4 are the following: $cluster\ 4 = [3, 1, 2, 4, 5, 6, 9, 7, 8] \rightarrow frequencies\ of\ cluster\ 4 = [1, 1, 1, 1, 1, 1, 0, 0, 0]$

On day 6 and accordingly to WBS scenario all points are available to be used. This means that the first 3 nodes of cluster 4 will be used again as in days 1,3 and 5.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 6	1	6	3, 1, 2	6	6	6

In that way, the plan unlit now has been designed as follows:

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	3, 1, 2	6	6	6
Route 2	1	2	4,5,6,7,8,9	6	6	6
Route 3	1	3	3, 1, 2	6	6	6
Route 4	1	4	4,5,6	3	3	3
Route 5	1	5	3, 1, 2	6	6	6
Route 6	1	6	3, 1, 2	6	6	6

As shown, nodes 1,2,3,7,8 and 9 have been visited as many times as their frequency defines, although nodes 4,5 and 6 have not. On that occasion the next available truck will be chosen, and we select the first day of the period. Now according to the WBS scenario, all nodes are forced to be used respecting only the constraints of the work-shift and capacity. Therefore, on day 1 nodes 4,5 and 6 will be added.

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	3, 1, 2	6	6	9
Route 7	2		4,5,6	3	3	
Route 2	1	2	4,5,6,7,8,9	6	6	6
Route 3	1	3	3, 1, 2	6	6	6
Route 4	1	4	4,5,6	3	3	3
Route 5	1	5	3, 1, 2	6	6	6
Route 6	1	6	3, 1, 2	6	6	6

The next step for the algorithm is to execute the VRP problem for each single day of the week and for the corresponding assigned points. The computation of the VRP is solved by the use of C&W method and the required data, are the distance matrix (table 3.1) and the corresponding saving's Table(table 3.3).

After using the C&W method for each day of the period we take the following initial solutions (Table 3.6).

Table 3. 6 The final plan of HA2 in application 1 (period = 6 day)

	Vehicle	Day	Assigned Nodes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 1 – 2 – 3 – 0	6	6	9
Route 7	2		0 – 4 – 5 – 6 – 0	3	3	
Route 2	1	2	0 – 7 – 8 – 9 – 4 – 5 – 6 – 0	6	6	6
Route 3	1	3	0 – 1 – 2 – 3 – 0	6	6	6
Route 4	1	4	0– 4 – 5 – 6 – 0	3	3	3
Route 5	1	5	0 – 1 – 2 – 3 – 0	6	6	6
Route 6	1	6	0 – 1 – 2 – 3 – 0	6	6	6

The total distance cost for all routes becomes 418 km. An important remark is that we have significantly more routes and visited nodes during the first two days compared to the other days. Also, both available vehicles were needed to complete the plan; the first vehicle completed one daily work shift every day of the week, while the second one assisted only in the first day.

The results of HA1 and HA2 in application 1

In application 1 algorithm HA2 works better than the HA1 (418 km < 431 km). The reason for this outcome is that the main idea of HA2 is to combine the nodes with same visiting frequency first. Consequently, when the neighbors of the same frequency are close to each other, HA2 works efficiently. For further understanding of this explanation, refer to Figure 3.2.

As we can see the nodes of same frequency are placed close to each other and therefore algorithm HA2 works effectively. In the following chapter (application 2) an opposite case is considered (neighbors of the same frequency are arbitrarily placed), in order to see if the behavior of the two algorithms changes.

3.2 Application 2

The inputs of Application 2 are similar to application 1, except one parameter: the frequency of visits in each node. Consequently, the Table of distances and savings between all the points remain the same.

Table 3.1 Euclidean distance matrix between all (x,y) points

	0	1	2	3	4	5	6	7	8	9
0	0.00	30.07	29.43	27.46	16.40	16.28	16.28	30.02	29.68	28.60
1	30.07	0.00	3.16	4.24	18.79	20.12	21.47	13.60	14.87	16.55
2	29.43	3.16	0.00	2.00	16.76	18.03	19.31	10.44	11.70	13.42
3	27.46	4.24	2.00	0.00	14.87	16.16	17.46	10.05	11.18	12.65
4	16.40	18.79	16.76	14.87	0.00	1.41	2.83	13.93	13.42	12.21
5	16.28	20.12	18.03	16.16	1.41	0.00	1.41	14.56	13.93	12.53
6	16.28	21.47	19.31	17.46	2.83	1.41	0.00	15.30	14.56	13.00
7	30.02	13.60	10.44	10.05	13.93	14.56	15.30	0.00	1.41	3.61
8	29.68	14.87	11.70	11.18	13.42	13.93	14.56	1.41	0.00	2.24
9	28.60	16.55	13.42	12.65	12.21	12.53	13.00	3.61	2.24	0.00

Table 3.3 Clarke & Wright saving matrix between all (x,y) points of Table3.1

	0	1	2	3	4	5	6	7	8	9
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	56.33	53.28	27.68	26.22	24.87	46.48	44.88	42.11
2	0.00	56.33	0.00	54.89	29.07	27.68	26.39	49.00	47.40	44.61
3	0.00	53.28	54.89	0.00	28.99	27.58	26.27	47.43	45.96	43.41
4	0.00	27.68	29.07	28.99	0.00	31.27	29.85	32.49	32.67	32.80
5	0.00	26.22	27.68	27.58	31.27	0.00	31.14	31.74	32.03	32.35
6	0.00	24.87	26.39	26.27	29.85	31.14	0.00	31.00	31.40	31.88
7	0.00	46.48	49.00	47.43	32.49	31.74	31.00	0.00	58.28	55.01
8	0.00	44.88	47.40	45.96	32.67	32.03	31.40	58.28	0.00	56.05
9	0.00	42.11	44.61	43.41	32.80	32.35	31.88	55.01	56.05	0.00

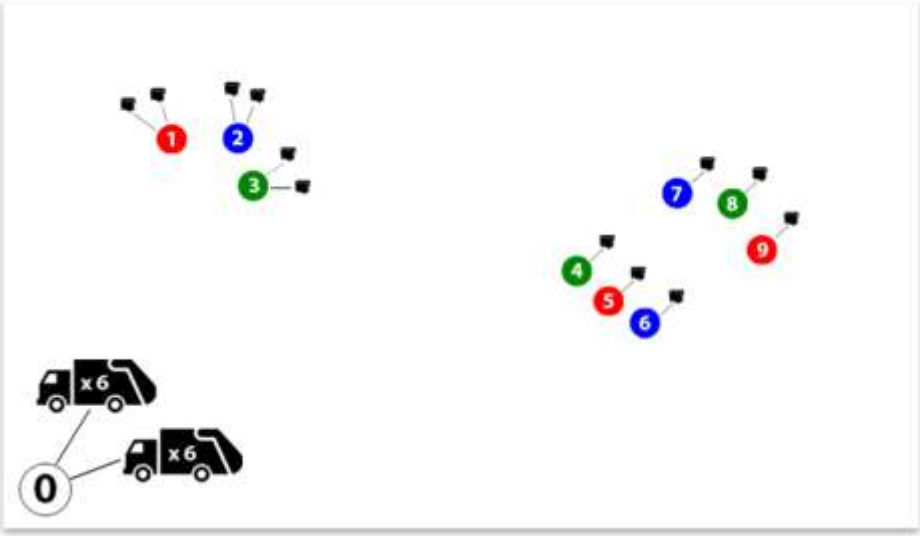
Moreover, the only difference between the two applications is the frequency of each node which is represented in the following table:

Table 3. 7 The number of bins in each node are shown by column demand and the number of visits in each node of application 2 are shown by column frequency

Node	**Number of bins in each node**	**Frequency**
1	2	4
2	2	3
3	2	1
4	1	1
5	1	4
6	1	3
7	1	3
8	1	1
9	1	4

Figure 3.3 illustrated the nodes of Table3.1

Figure 3. 3 Graph illustration of nine nodes consisting of 12 bins, one depot = 0 and two waste trucks of six units cabin



Red Nodes: frequency = 4, Blue Nodes: frequency = 3, Green Nodes: frequency = 4,

Black Nodes: bins in each point

HA1 in Application 2

The steps of HA1 in application 2 are the same with those of HA1 in application 1. The execution of algorithm HA1 in application 2 produces the following plan:

Table 3. 8 The final plan of HA1 in application 2 (period = 6 day)

Route No	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 1 – 2 – 7 – 8 – 0	6	6	7
Route 7	2	1	0– 5 –0	1	1	
Route 2	1	2	0 – 3 – 9 – 4– 5 – 6 – 0	6	6	6
Route 3	1	3	0 – 1 – 2 – 7 – 0	5	5	5
Route 4	1	4	0 – 5 – 9 – 6 – 0	3	3	3
Route 5	1	5	0 – 1 – 2 – 7 – 9 – 0	6	6	6
Route 6	1	6	0 – 1 – 9 – 5 – 6 – 0	5	5	5

The total cost for all routes is 463 km. The first vehicle made 6 work-shifts during the period while the second vehicle made just one work-shift on day 1. So, the total number of routes are 7 through the whole period.

HA2 in Application 2

The steps of HA2 in application 2 are the same with those of HA2 in application 1. The execution of algorithm HA2 in application 2 produces the following plan:

Table 3. 9 The final plan of HA2 in application 2 (period = 6 day)

Route No	Vehicle	Day	Routes	Current capacity (number of bins) (assum1)	Work-shift (number of bins) (assum2)	Total Daily Work- Shift (assum2)
Route 1	1	1	0 – 1 – 2 – 9 – 5 – 0	6	6	6
Route 2	1	2	0 – 3 – 7 – 8 – 4 – 6 – 0	6	6	6
Route 3	1	3	0 – 1 – 2 – 9 – 5 – 0	6	6	6
Route 4	1	4	0 – 7 – 9 – 5 – 6 – 0	6	6	6
Route 5	1	5	0 – 1 – 2 – 9 – 5 – 0	6	6	6
Route 6	1	6	0 – 1 – 2 – 9 – 5 – 0	6	6	6

The total cost of all routes is 595 km. In this execution only the one waste truck is used out of the two available. The vehicle makes just 6 routes, each for every single day of the period.

The results of HA1 and HA2 in application 1 and application 2

The total cost for the routes of algorithm HA1 and HA2 in application 2 are 463 km and 595 km respectively. In contrast to the last example, in application 1 algorithm HA2 has lower distance cost than HA1. These results are due to the different frequency value of the nodes. Moreover, in the first example, the nodes of same frequency were close to each other, and in this case algorithm HA2 fits better. On the other hand, in the second case the coordinates of all points do not change but the corresponding frequencies are arbitrarily chosen and HA1 produces routes with lower distance cost.

3.3 Application in large-scale problems

The collection and transport of bins from the streets of a populated area is a difficult and demanding task for any municipality. In many cases, these areas are large and extend to the area of entire cities. To examine the performance of the proposed heuristics to such large cases, in this section we present eight applications with a range of 50 until 1400 pick-up points. For each case we generated 50 instances.

The problem generator has been provided with the following inputs:

- Number of vehicles
- Number of nodes
- Number of days of the period
- Number of total bins
- Vehicle capacities
- Node demand (number of bins)
- Node frequency
- Coordinates of nodes
- Coordinates of depot
- Distance matrix
- Available nodes to visit in a day

Additionally, note that the following probability distributions have been used for generating the data:

- Vehicle capacity is generated from a Uniform distribution $U(100,100)$.
- Node demand is generated from a Uniform distribution $U(1,3)$.
- Node frequency is generated from a Uniform distribution $U(2,5)$.
- The coordinates of the nodes and depot are generated from a Uniform distribution $U(50,10000)$.
- Available nodes to visit each vehicle in a day is generated from Uniform distribution $U(100,100)$.

The data have been generated using Matlab, algorithms HA1 and HA2 have been applied in each problem and the corresponding results are presented and further analyzed in the following paragraphs of this section.

For each number of nodes, 50 problems have been generated and solved, and the results are provided by the averages presented in Table 3.10 and in Figures 4.3, 4.4. The first column of Table 3.10 shows the number of points, the second column is the average number of bins. The third column presents the average number of visits in each execution. Columns 4 and 5 present the results of average costs in km for HA1 and HA2 respectively. The last column computes the percentage difference between the 4^{th} and the 5^{th} column.

Table 3. 10 Average total distances and percentage differences for heuristic algorithm HA1 and heuristic algorithm HA2 for a fleet of 15 waste trucks with capacity = 100

Number of Nodes	Average total number of bins	Average total number of visits per period	Average distance cost in km for HA1	Average distance cost in km for HA2	Percentage difference of distance cost (HA1 – HA2)
50	97.66	170.04	187.1	186.68	0.22%
150	297.96	520.4	438.2	453.89	-3.45%
350	698.4	1223.8	924.81	957.92	-3.45%
500	1001.08	1742.92	1258.35	1301.66	-3.32%
700	1396.6	2451.98	1723.68	1769.69	-2.59%
1000	1995.12	3492.38	2379.07	2443.50	-2.63%
1200	2403.86	4204.6	2835.25	2905.65	-2.42%
1400	2801.92	4901.54	3264.27	3333.19	-2.06%

Figure 3.4 presents the average distance vs. the number of nodes. Naturally, as the number of nodes increases the total distance increases too. As can been seen, heuristic algorithm HA2 is better only in the first case where the number of nodes is 50.

Figure 3. 4 Average total distance of heuristic algorithm HA1 and HA2 for a fleet of 15 vehicles with capacity = 100

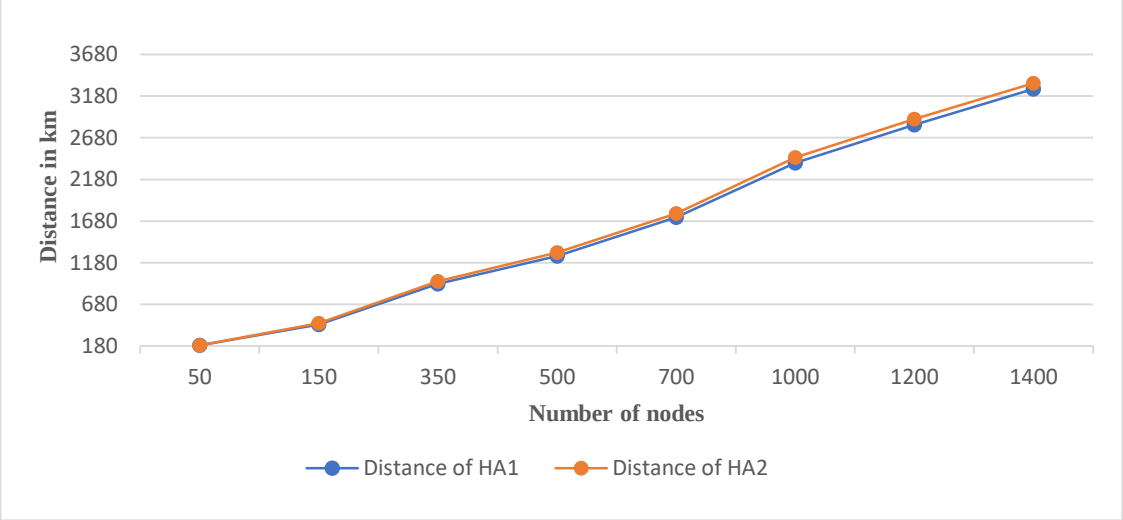
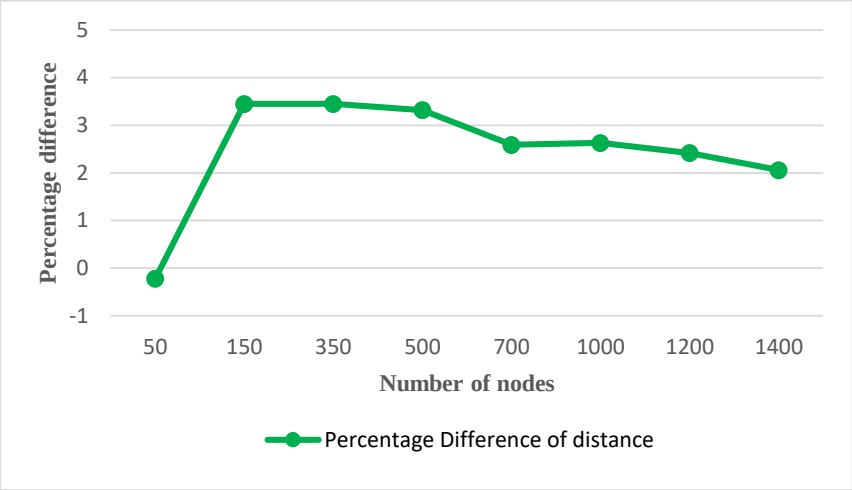


Figure 3.5 shows the percentage difference of the average distance between algorithm HA1 and HA2 (column 6 of Table3.10). The difference varies between 0% - 4.50% . Only one result is below zero, and algorithm HA1 produces better results in all other cases.

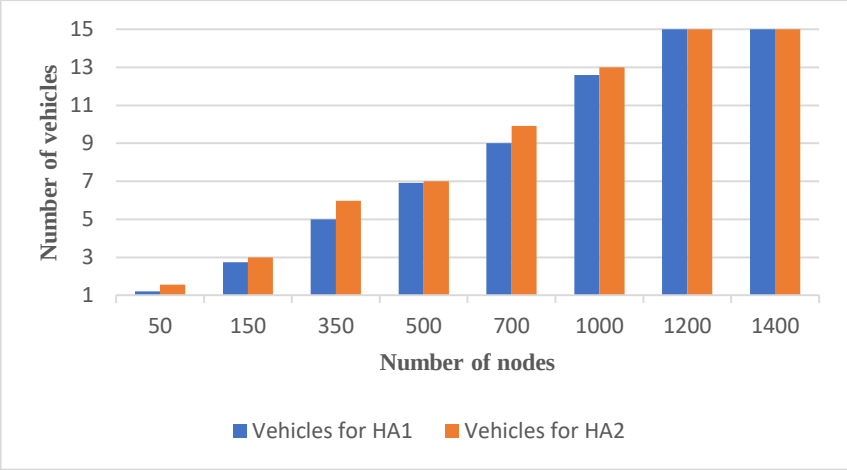
Figure 3. 5 Percent difference of distance between heuristic algorithm HA1 and HA2 for a fleet of 15 vehicles with capacity = 100



Use of fleet

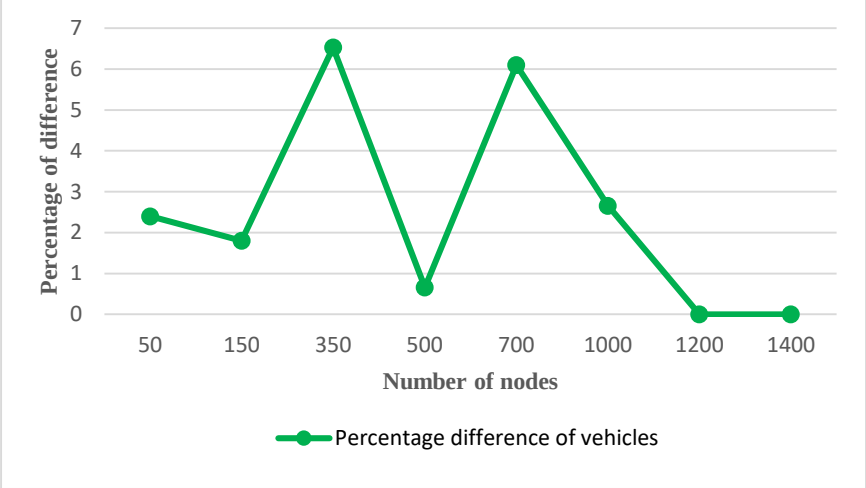
Figure 3.6 presents how many vehicles were needed in order to empty all the bins of the corresponding examples. Naturally, the number of waste trucks increases while the number of nodes increases. In most cases algorithm HA2 uses more vehicles than HA1. Moreover, for the cases of 1200 nodes and more, both algorithms use100% of the fleet.

Figure 3. 6 Average total number of vehicles that needed for heuristic algorithm HA1 and HA2 for a fleet of 15 vehicles with capacity = 100



In Figure 3.7 the percentage difference of use of the fleet between algorithms HA1 and HA2 is shown. The points of the graph which are below zero show that algorithm HA2 uses less vehicles than HA1. In the current graph most points are above zero which actually means that HA1 needs less waste trucks than HA2. For the cased 1200 nodes and above both heuristics use 100% of the fleet.

Figure 3. 7 Percent difference of vehicles between of heuristic algorithm HA1 and HA2 for a fleet of 15 vehicles with capacity = 100



Total number of routes

Table 3.11 presents some extra information about the 8 implementations. The first, the second and the third column contain the total number of nodes, the average total number of bins and the corresponding average total number of visits for each implementation, as in Table3.10. The fourth and the fifth columns present the total average number of routes that HA1 and HA2 need in order to pick up all waste. Heuristic HA2 has been built to minimize the total number of routes and as expected, it produces less routes than HA1.

Table 3. 11 Average total number of routes for heuristic algorithm HA1 and heuristic algorithm HA2 for a fleet of 15 waste trucks with capacity = 100

Number of Nodes	Average total number of bins	Average total number of visits per period	Total number of unloads for HA1	Total number of unloads for HA2	Percentage difference of total routes (HA1 – HA2)
50	97.66	170.04	13.7	13.38	2.33%
150	297.96	520.4	38.74	39.76	-2.56%
350	698.4	1223.8	86.26	85.78	0.55%
500	1001.08	1742.92	119.96	118.4	1.3%
700	1396.6	2451.98	167.82	166.04	1.06%
1000	1995.12	3492.38	237.82	237.64	0.07%
1200	2403.86	4204.6	281.32	280.28	0.36%
1400	2801.92	4901.54	330.54	328.08	0.74%

Figure 3.8 presents how many routes are produced for each example by HA1 (blue line) and HA2 (orange line). As expected, HA2 constructs fewer routes than those of HA1, in most cases.

Figure 3. 8 Average total number of routes of algorithm HA1 and HA2 for a fleet of 15 vehicles with capacity = 100

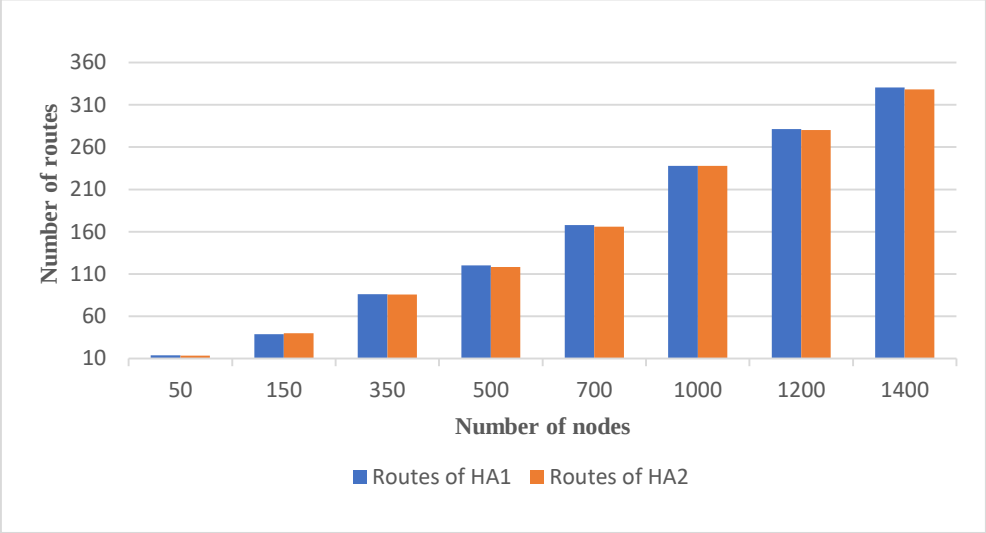
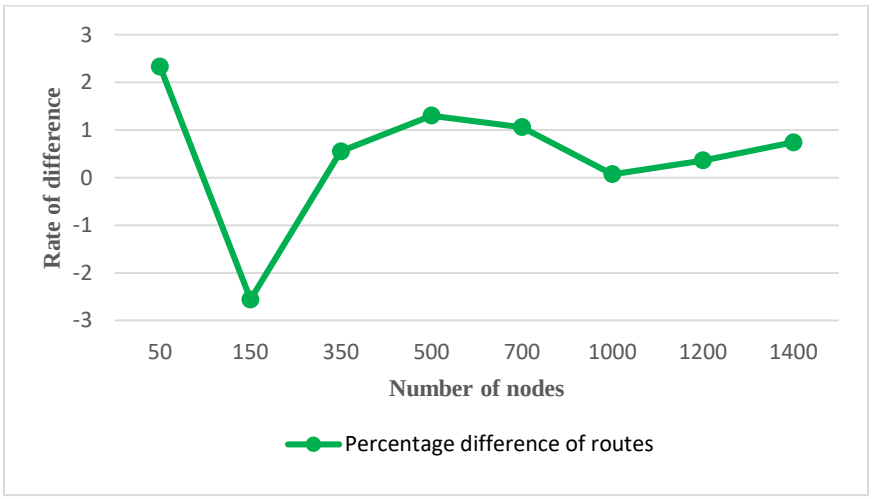


Figure 3.9 indicated that in the nine out of ten cases, algorithm HA2 creates less routes than HA1. Of course, less number of routes doesn't automatically mean less distance cost.

Figure 3. 9 Percent difference of routes between of heuristic algorithm HA1 and HA2 for a fleet of 15 vehicles with capacity = 100



4. Local Search Methods: Improvements Heuristics for Single Route and Multi Route problems

Improvement Heuristics Description

This category of algorithms can be used if an initial solution of a routing problem has already been found. The categories of local search algorithms that analyzed in this section will be executed in the initial solutions of the case study in chapter 5. The aim of these heuristic methods is to further refine a route. In each iteration of the algorithm, the cost either remains the same or is improved. Improvements are accepted.

As an example, assume the following route: [1 – 2 – 3 – 4 – 1]. We compute the corresponding distance cost of the route and save it in C_1. Thereafter, we consider which improvement heuristic option is more suitable for the existing route. Let us suppose that the outcome in the very first iteration whereby nodes 2 and 3 have been swapped is the following:

ROUTE = [1 – 3 – 2 – 4 – 1] and the corresponding distance cost is C_2, where C_2 < C_1.

The next step is to save that change and to move on for further improvement. The number of iterations depends on how large the problem is and the type of algorithm used. It is noted that all possible changes cannot be examined, if the total number of nodes is very large. It is therefore necessary to find an appropriate terminal condition in a reasonable computation time, so that the distance will be reduced as much as possible.

The improvement heuristic methods are categorized into single route problems (Intra Route) and multi route problems (Inter Route). The former category consists of:

- λ – opt
- Or – opt

and the latter category consists of:

- Exchange-operator
- Relocate-operator
- Cross-exchange

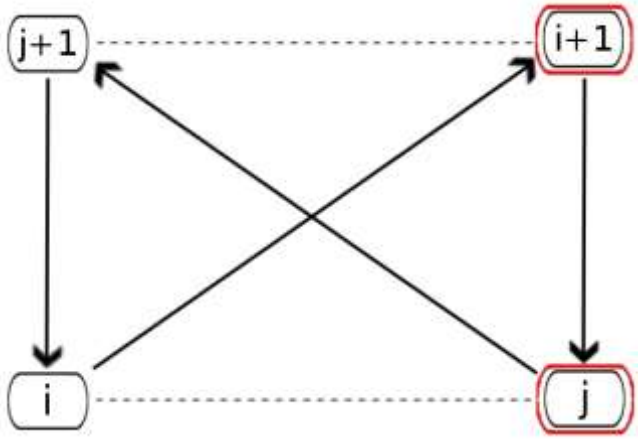
- Cyclic-transfer
- Geni-exchange

So, the variations of the first category are adequate for the Travelling Salesman Problem. In a multi-route problem, a method from the Inter Route category is recommended. This chapter is dealing with λ – opt and Exchange-operator, which will be used in the case study of the thesis. Also, the Or-opt, Relocate-operator and cross-exchange are analyzed in the appendices.

4.1 λ – opt [Intra Route]

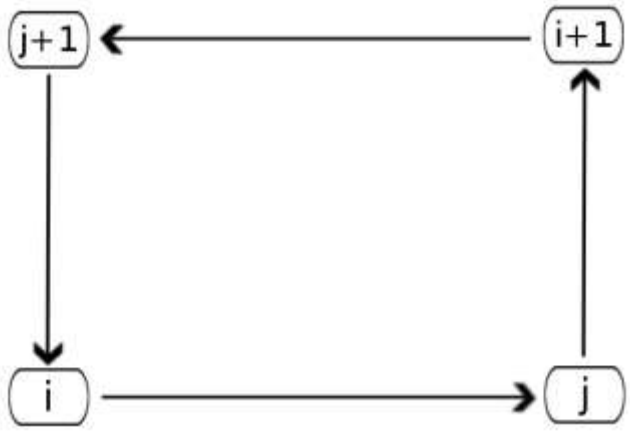
The λ – opt algorithm belongs to Intra Route category and is one of the most popular local search methods. In particular, this method is implemented by swapping two nodes in each iteration in a way that it can completely alter and disentangle the edges of the graph, reducing the total distance. Moreover, if the exchange includes more than two nodes in each iteration, the model takes the corresponding domain name. For instance, if we swap three nodes at a time the model is known as 3-opt, otherwise if the swap includes 4 nodes at a time it is known as 4-opt etc. Consider the simple example below: $\mathrm{ROUTE} = [\,i \rightarrow i+1 \rightarrow j \rightarrow j+1 \rightarrow I\,]$.

Figure 4. 1 Schematic illustration of initial feasible solution for a single route problem with 4 nodes, 1 vehicle and 1 depot



Since 2-opt application has been selected, we must choose two different nodes from the route to exchange each other. To save time, only the first repetition of the problem is executed. If we exchange node (i+1) with node (j), the result is going to be this: $\mathrm{ROUTE} = [\,i \rightarrow j \rightarrow i+1 \rightarrow j+1 \rightarrow i\,]$.

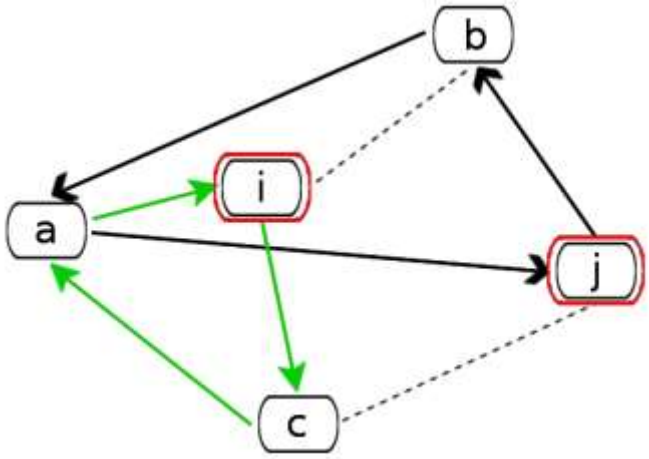
Figure 4. 2 Representation of route (Figure 4.1) following the application of λ-opt algorithm



4.2 Exchange – Operator [Inter Route]

The execution of exchange-operator is similar to λ-opt, as in this case an exchanging method between single nodes is used. The difference here is that any exchange must be made between routes and not in a single route. In more detail, two nodes from separate routes have to be swapped in cases in which the distance decreases. In literature this method is also known as 2-opt*. If the exchange is related with 3 or more nodes (n), then the name becomes n-opt*. In the next example there are two routes, which consist of two customers: ROUTE 1 = [a → j → b → a] and ROUTE 2 = [a → i → c → a]

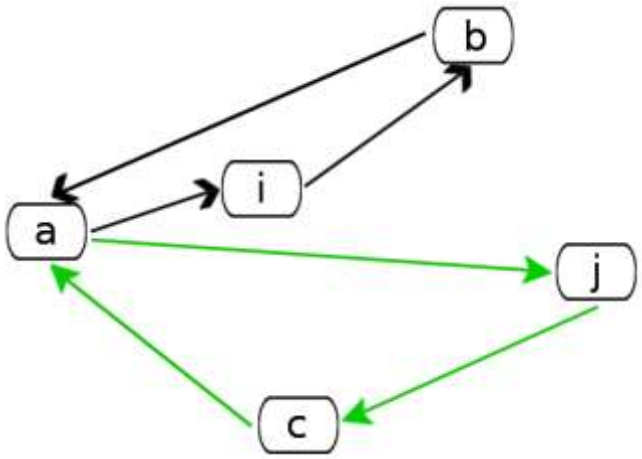
Figure 4. 3 Schematic illustration of initial feasible solution for a multi route problem with 5 nodes, 1 vehicle and 1 depot



To implement the exchange-operator, the same number of customers in each route must be selected. In ROUTE 1 we select node (i) and in ROUTE 2 customer (j). The result after exchanging

these customers between the two routes is the following: $\text{ROUTE } 1 = [\,a \rightarrow i \rightarrow b \rightarrow a\,]$ and $\text{ROUTE } 2 = [\,a \rightarrow j \rightarrow c \rightarrow a\,]$

Figure 4. 4 Representation of route (Figure 4.3) following the application of Exchange-operator algorithm



5. Case Study

This thesis aims to propose a solution to the collection and transportation of recyclable material in the municipality of Chios. A collaboration with the corresponding municipal cleaning service of Chios was arranged in order to collect the needed data. During the first meeting, the author realized that the process of waste collection and transportation was planned based on experience. A questionnaire was constructed and sent to the appropriate authority to acquire all needed data. The content of the questionnaire gathered information about the fleet of collection trucks, truck timetables, the location of the pick-up points etc.

In the case study we construct the plan to route the municipality's vehicles in order to serve all bins at the prescribed frequencies using algorithm HA1.

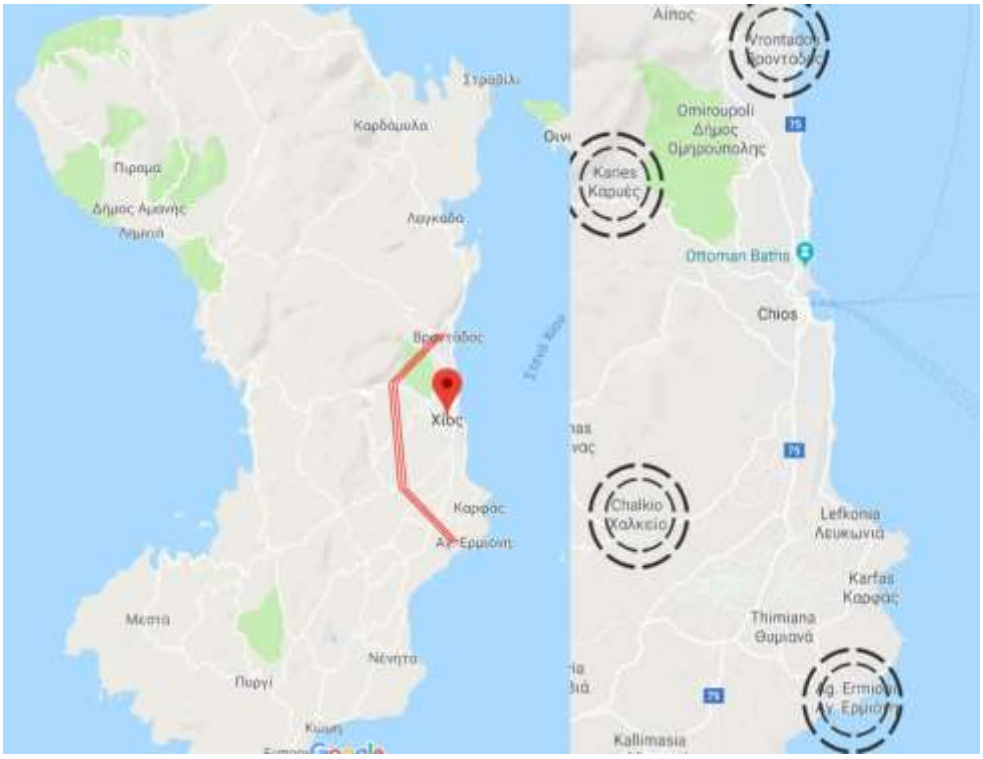
Below is the data needed to solve the problem:

- the nodes with the corresponding distance matrix
- the number of bins at each node,
- the frequency of visits,
- the number of available vehicles

These points are covered in more detail in appendix (Appendix L, Appendix M).

The area under consideration concerns the capital ("Chios") and some villages in the north and south of the city. The depot and the disposal site are located in the village of Chalkeios. The upper boundary of a daily route is on the north part of the city in village "Vrontados" and north west of Chios in Karyes. The most southern area of a daily route is at the village "Agia Ermioni" and south west in village Chalkeios where the depot is also located. The visualization of these are presented in the following map (figure 5.1) which was taken from google maps.

Figure 5. 1 Geographical bounds in which the collection of plastic materials executed daily by municipality of Chios.



As mentioned before, the depot and the appropriate storage site are in the same location (village of "Chalkeios"). The material collection fleet consists of three vehicles with capacity 8000, 12000 and 16000 liters, respectively. The vehicle with 12000 lt capacity is used for the collection of plastic. The quantity of waste inside each bin is not considered. Instead the vehicle may collect up to 135 bins per shift independently. Furthermore, the frequency of visits required per location is known from historical data of the municipality.

Table 5.1 provides information about the number of bins and the total visits of the nodes. The second column provides the data about the total number of bins, which must be selected by the appropriate vehicle. Column number three presents the total number of visits which is computed as the total number of nodes multiplied by the frequency of visits. Finally, in the last column we find the ID of the waste truck which corresponds to the vehicle's registration number.

Table 5. 1 Information for the collection of plastic garbage bins

Number of nodes	Total number of bins	Total number of visits	ID of vehicle
116	171	360	KHH2895
Total demand in pickup-points = 171		**Total visits in a week = 360**	

5.1 Creating the distance matrix

The first step to address the case study of this thesis is to create a map. This procedure is based on Google Maps and Google Earth tools and involves the transition of geographical points to google maps according to the following steps:

Step 1. Open a browser
Step 2. Connect your browser with your google account
Step 3. Type https://www.google.com/maps/about/mymaps/ into the URL bar
Step 4. Press the appropriate button to get started
Step 5. Press the button "Create a new map"
Step 6. Type the needed address into the search bar
Step 7. Add that address by the appropriate button
Step 8. If you haven't added all the geographical points go to **step 6.**

The following procedure saves the coordinates of all geographical points of our map into an excel sheet. To do that follow the next steps:

Step 1. Open your saved map in your browser
Step 2. Go the menu bar and export the map as KML/KMZ
Step 3. Download the file
Step 4. Open the google earth application
Step 5. Go to the menu bar of google earth and select "Open file"
Step 6. Choose the file which has been downloaded in **step 3.**
Step 7. Go to the menu bar of google earth again and press "save as" and save that file as .KML
Step 8. Open a new excel sheet
Step 9. Drag and drop the saved file from **step 7.** on the open excel sheet
Step 10. Select the column with all coordinates and press copy "Ctrl+C"
Step 11. Open another excel sheet
Step 12. Paste "Ctrl+V" the column from **step 10.** and mark it with your cursor
Step 13. Go to "Data" from excel menu bar and press "Text to columns"
Step 14. Now in the open window of the previous step separate the columns by comma
Step 15. Swap the two columns in order to bring all the latitude points in the first column and all the longitude points in the second one
Step 16. Finally save the excel sheet

As the coordinates have been saved, the computation of the distance matrix can be easily found. The first step to create the matrix is to use a tool such us google maps to compute the distances between all the points. In some occasions however, as in the case study of this thesis, the matrix computation cannot be done by hand in reasonable time due to the large number of nodes. For that reason, a google api tool called “Distance Matrix Api” is utilized. To use the api follow the next steps:

Step 1. Open a browser
Step 2. Type https://cloud.google.com/maps-platform/?apis=routes in the URL bar
Step 3. Select the “Distance Matrix Api” in the open window
Step 4. Select or create a project as and press next
Step 5. Create a billing account and proceed
Step 6. Save the Api key number

Once the api key has been enabled, the following Matlab code is run to compute the distance matrix:

Matlab code

```
clear, clc
clear variables
% Read the excel file with the coordinates
xlsFile = ‘name.xlsx';
% Put that file into a variable
array = xlsread(xlsFile);
% Read the number of lines from the file
[M, ~] = size(array);
% Distance matrix initialization: create a matrix with M X M dimensions
D = zeros(M,M);
%  Save in the variable url the following URL address
url = 'https://maps.googleapis.com/maps/api/distancematrix/json';
% Save in variable apiKey the api key you have taken from the previous procedure
apiKey = ‘insert_THE_api_KEY_here';
```

```
% Variable destCoords will show the destination and must be saved as a string
destCoords = ' ';
% Read all the coordinates which are going to be the destinations
for i = 1 : M
   % Save the chosen coordinates in a row and separate each other by "|"
   destCoords = [ destCoords, '|', sprintf('%0.4f, %0.4f', array(i,1), array(i, 2))];
end
for i = 1 : M
   % Save the first point (latitude, longitude) in the string variable originCoords
   originsCoords = sprintf('%0.4f,%0.4f', array(i, 1), array(i, 2));
   % Compute the distance by the function "webread"
   res = webread(url, 'origins', originsCoords, 'destinations', destCoords, 'duration', dur, 'key', …
apiKey);
  % Save the result
   distances = [res.rows.elements.distance];
end
disp(D)
```

5.2 The solution of HA1 for collection of plastic materials

The solution of the case study was performed in Matlab - R2016a version. The size of the distance matrix was 116X116 and the algorithm used to run the data was heuristic HA1. The problem was solved in three seconds in a Linux Mint environment. The technical characteristics of the laptop were Intel Core i5-5200U, with SSD hard drive of 120 Gigabyte capacity.

Table 5.2 provides information for the routes which were executed by the Municipality of Chios (MoC). The first column presents the route number and in the second one the day that the route is executed. The third column presents the number of bins collected in the corresponding day. The total distance is shown in the last column. The "Maximum collected bins" informs how many bins the vehicle can collect in a single day (constraint of work-shift).

Table 5. 2 Information for the routes of the collection of plastic garbage bins by MoC

Number of routes	Scheduled day	Total number of bins	Distance cost in km
1st	Monday	135	65.7
2nd	Tuesday	110	45.77
3rd	Thursday	135	65.7
4th	Wednesday	110	45.77
Total distance cost in km for all days = 222.93		**Maximum collected bins = 135**	

Table 5.3 provides information for the routes planned by the heuristic algorithm HA1. The first column presents the number of the route and in the second one the day that the route is executed. The third column presents the number of bins collected in the corresponding day. The total distance is presented in the last column. The "Maximum collected bins" informs how many refuse-bins the vehicle can collect in a single day (constraint of work-shift).

Table 5. 3 Information for the routes of the collection of plastic garbage by HA1

Number of routes	Scheduled day	Total number of bins	Distance cost in km
1st	Monday	135	77.5
2nd	Tuesday	26	9.25
3rd	Wednesday	135	77.5
4th	Thursday	26	9.25
5th	Friday	74	21.18
6th	Saturday	74	21.18
Total distance cost in km for all days = 215.87		**Maximum collected bins = 135**	

5.3 Comparison results between MoC and HA1

Table 5.4 presents the results of MoC and HA1, as well as the differences between the two heuristics. The first column presents the weekly distance for MoC. In the next column, the weekly distance of HA1 is given. The fourth and fifth columns show the total number of routes for MoC and HA1 respectively. The last column computes the percentage difference between the distance cost of the first and the second column.

Table 5. 4 Comparison between the routes of municipality of Chios (MoC) and algorithm HA1, for a graph with 117 nodes, one depot, one vehicle of 12000 lt and 360 visits in one week

Distance cost in km for MoC	Distance cost in km for HA1	Total number of routes following MoC	Total number of routes following HA1	Percentage difference of distance (HA1 – MoC)
222.93	215.87	4	6	-3.2%

Figure 5.2 shows the daily distance for both MoC and HA1. The blue line corresponds to the MoC distance and assumes zero values in two days out of the six available. On the other hand, algorithm HA1 produces solutions for every single day of the period and in that way two more routes are created compared with MoC. The HA1 routes result in a shorter overall distance.

Figure 5. 2 Distance of MoC and heuristic algorithm HA1 for each single day for a one vehicle with capacity = 135

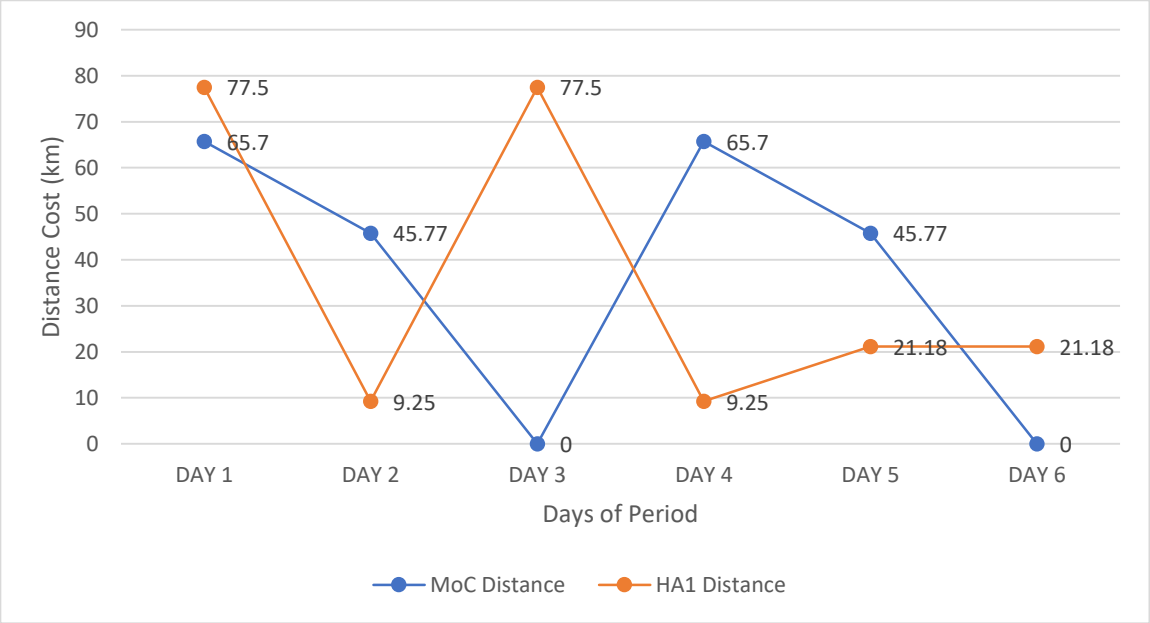
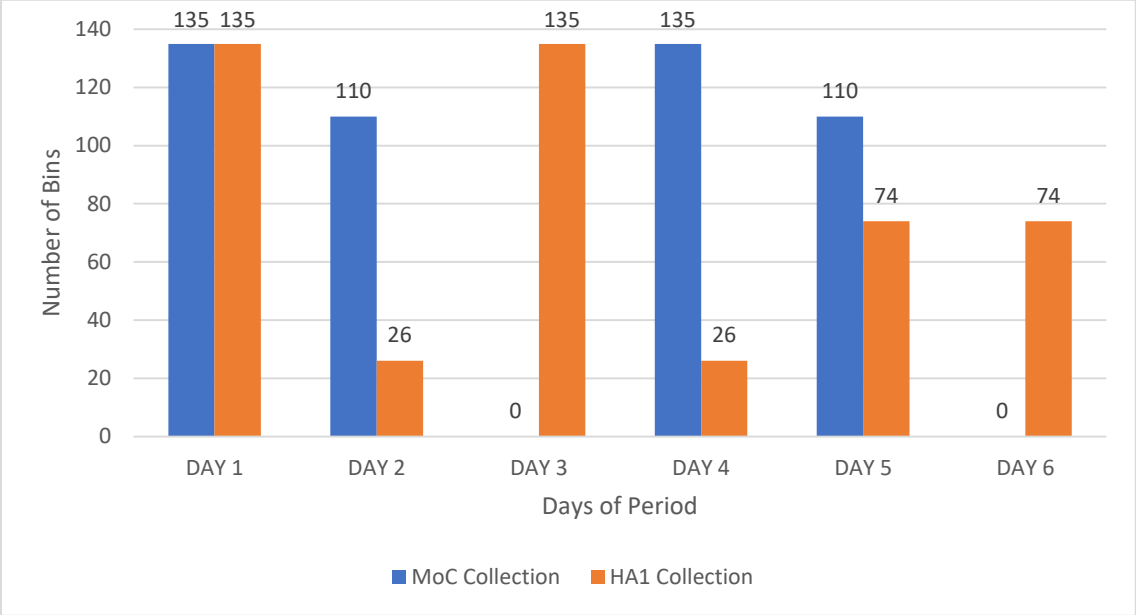


Figure 5.3 shows how many bins are collected daily by both plans. As expected, on days 3 and 6 there is no bar for MoC, while there are pick-ups by the plan of algorithm HA1 in each day.

Figure 5. 3 The number of collected bins of MoC and heuristic algorithm HA1 for each single day for one vehicle with capacity = 135 bins



5.4 Implementation of 2- opt and Exchange – Operator in HA1

In this paragraph the local search methods of 2-opt and exchange-operator (Chapter 4) are implemented in the initial solutions of HA1, for further improving the overall distance. The first column of Table 5.5 shows the distance cost of HA1 in the case study of this thesis. The second column gives the new distance cost, after using the exchange operator. Finally, the third column presents the distance after executing 2-opt in the result of the second column. The fourth and the fifth columns show the percentage reductions of the second and the third columns respectively. The total reduction following the implementations of 2-opt and exchange-operator, is shown in the last column.

It is noted that the cost reduction between the MoC solution and the refined HA1 solution is 10%, with more than half of it been attributed to the refinement of the initial HA1 solution.

Table 5. 5 Implementation of exchange-operator and 2-opt method in the initial solution of HA1

Distance cost in km of initial solutions of HA1	Distance cost in km following exchange-operator	Distance cost in km following 2-opt method	Reduction rate of distance after exchange-operator	Reduction rate of distance after 2-opt method	Total reduction rate of distance
215.87	213.08	202.93	-1%	-5%	-6%

6. Conclusions and opportunities for future research

The collection of solid waste by municipalities is a problem of high importance and concerns all societies around the world. Every year, almost 2 billion tons of waste are generated globally, and this continues to grow annually [22]. The largest part of this total amount relates to solid waste transported by trucks and delivered to appropriate stations.

In this thesis the solid waste collection problem from municipalities is solved by adopting a method inspired by the Periodic Vehicle Routing (PVRP). We deal with the disposal of solid waste in an urban area in order to minimize the total distance of the routes that are executed by a fleet of collection trucks. The model embraces the constraints of capacity and work-shift.

To solve this problem two heuristics, namely HA1 and HA2, have been developed. Both create initial solutions based on the Clarke & Wright savings algorithm. HA1 creates feasible routes based on C&W while at the same time distributing the constructed routes over the days of the period. On the other hand, HA2 involves clustering according to frequency first, and then the nodes are assigned to the days of the period before a classic VRP is executed for each day.

Algorithm HA2 creates a limited number of routes. However, HA1 results in lower overall cost, especially in larger problems. Comparing both algorithms, it can be seen that HA2 is better only when the number of nodes is less than 100. Using algorithm HA1 is to solve the collection of plastic waste in the Municipality of Chios (MoC), the solution had lower cost that the plan currently executed by the Chios municipality. Refining the HA-1 solution by vrp-exchange and 2-opt resulted in further reduction of the overall distance. In fact, the cost reduction between the MoC solution and the refined HA1 solution is 10%, with more than half of it been attributed to the refinement of the initial HA1 solution.

The case study illustrated that HA1 produces results which contain a multiple node in some routes and far fewer in others. Consequently, it appears that the WBS constraint doesn't allow distributing the visited nodes through the plan uniformly, creating paths of uneven length. Furthermore, the case examined in this thesis does not deal with multiple disposal sites and depots.

Further research could focus on a more efficient assignment of nodes to the days of the set period, minimizing the total number of routes. Future research could also consider the implementation of additional local search algorithms to refine initial solutions, such as those presented in the appendix.

References

[1]. eea.europe.eu, (1990). European Environment Agency of the European Union. Available at:https://www.eea.europa.eu/data-and-maps/indicators/waste-recycling-1/assessment

[2]. Buenrostro Delgado Otoniel, & Bocco Gerardo, & Silke Cram, (2000). Classification of sources of municipal solid wastes in developing countries. *Resources, Conservation and Recycling* 32(1), 29–41.

[3]. Gaurav K. Singh, & Kunal Gupta, & Shashank Chaudhary (2014). Solid Waste Management: Its Sources, Collection, Transportation and Recycling. *International Journal of Environmental Science and Development*, 5(4), 347-351.

[4]. KatjaBuhrkal, & AllanLarsen, & StefanRopke, (2012). The Waste Collection Vehicle Routing Problem with Time Windows in a City Logistics Context. *Procedia - Social and Behavioral Sciences*, 39, 241-254.

[5]. Aljoscha Gruler, & Christian Fikar, & Carlos Contreras Bolton, (2015). A Simheuristic for the Waste Collection Problem with Stochastic Demands in Smart Cities. *Simulation in Production and Logistics*, pp.49-58.

[6]. Iliya Markov, & Sacha Varone, & Michel Bierlaire, (2015). The waste collection VRP with intermediate facilities, a heterogeneous fixed fleet and a flexible assignment of origin and destination depot. *Transport and Mobility Laboratory. Ecole Polytechnique Fédérale de Lausanne*, 85, pp 256-273

[7]. Nickolas J Themelis, & Karstein Millrath, (2004). The case for waste to energy as a renewable source of energy. *Earth Engineering Center, Columbia University and Waste-to-Energy, Research and Technology Council, EEC.*

[8]. worldbank.org, (1944). Component of the World Bank Group. Available at: http://siteresources.worldbank.org/INTURBANDEVELOPMENT/Resources/336387-1334852610766/Chap3.pdf

[9]. Sofie Coene, & Arent Arnout, & Frits C.R. Spieksma (2008). The Periodic Vehicle Routing Problem: A Case Study. *Department of Decision Sciences and information management (KBI)*, 1-16.

[10]. web.mit.edu, (1861). Massachusetts Institute of Technology/ Available at: http://web.mit.edu/urban_or_book/www/book/chapter6/6.4.12.html

[11]. Vera C.Hemmelmayr, & Karl F.Doerner, & Richard F.Hartl, (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3), 791-802.

[12]. Jens Lysgaard (1997). Clarke & Wright's Savings Algorithm. *Department of Management Science and Logistics. The Aarhus School of Business.*

[13]. Olli Braysy, & Michel Gendreau (2005). Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Peer-reviewed Academic Journal Transportation Science,* 39(1), 1-29.

[14]. Mohd. Junedul Haque, & Khalid. W. Magld (2012). Improving the Solution of Traveling Salesman Problem Using Genetic, Memetic Algorithm and Edge assembly Crossover. *International Journal of Advanced Computer Science and Applications*, 3(7), 108-111.

[15]. Christian Prins (2008). The route-first cluster-second principle in vehicle routing. *Institute Charles Delaunay, University of Technology of Troyes (UTT) France.*

[16]. Vivian W.Y.Tam (2008). Economic comparison of concrete recycling: A case study approach. *Resources, Conservation and Recycling,* 52(5), 821–828

[17]. Dr.Salah M., & El-Haggar PE (2007). Municipal solid waste (MSW) is a pool of various solid wastes by towns and cities from different types of household activities. *Environmental Materials and Waste, 2016*

[18]. epa.gov, (1970). United States Environmental Protection Agency. Available at: https://www.epa.gov/sites/production/files/2016-03/documents/r02002.pdf

[19]. Teresa Bianchi-Aguiar, Maria Antonia Carravilla, Jose F. Oliveira (2011) Vehicle routing for mixed solid waste collection – comparing alternative hierarchical formulations. *Instituto de Engenharia de Sistemas e Computadores do Porto, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias*

[20]. nrcrecycles.org,(1978). National Recycling Coalition at: https://nrcrecycles.org/glossary-3/

[21]. afdc.energy.gov, (2018). United States Department of energy https://afdc.energy.gov/data/10309

[22]. theworldcounts.com . Live counter of earth's resources at: http://www.theworldcounts.com/counters/shocking_environmental_facts_and_statistics/world_waste_facts

Appendix A: Algorithm for the initialization of the inputs of HA1

A.1 Algorithm in Matlab: mainHA1.m

```
% C contains the distance matrix
C = xlsread('distance_matrix_in_Excel.xlsx');
% The distance matrix saved in variable dist too.
dist=C;
% Define the truck's capacity. The vector can consist of one non negative number in case of a single vehicle
% otherwise can be a lot of vehicles.
fleet = [capacity];
% Define the demand for each node. The demand can be zero if the load of a node is a negligible amount or another non negative
% number. The first number must be always zero as the first node refers to the depot.
d = [0, demand];
% Define the frequency for each node and initialize the first index as zero because refers to the depot.
f = [0, frequency];
% Define the M days of the period and the available driving hours per day for each vehicle.
period = M; dailyLIMIT = max_NODES_per_day;
% The following function takes as input the distance matrix and produces the list of savings as Clarke & Write method defines.
saving_C = savingARRAY( dist );
% The second step of Clarke & Write classifies the list of savings in descend order.
```

```
sorted_C = sortARRAY( saving_C );
% The following function implements the Clarke & Wright in Periodic VRP. The function gives priority to build the routes following the
% Clarke & Write steps, while those routes adjusted to the days of the period randomly. The results are the number of vehicles that
% needed (truckNUM) to execute the routes, the path of visits (r), the total number of routes (rtrnsOFvhcl) and the total cost in
% terms of distance (totCOST). The inputs of the problem have already been defined in the previous lines.
[ r,truckNUM, rtrnsOFvhcl, totCOST ] = CWpvrp( sorted_C,dist,fleet, d, f, period, dailyLIMIT );
% Variable r saved in another variable as r1. Each row is a different day and each cell is a different route.
r1=r;
% In variable initTC the cost of CWpvrp is saved in order to be used in function vrpEXCHANGE as an input. That input represents the
% distance cost of the initial solution.
initTC = totCOST;
% In the following function the VRP - Exchange is implemented. VRP-exchange is a method which belongs to inter route algorithms
% and is used to reduce the cost of distance of an initial route. The following function takes as input the travel cost (initTC)of the
% route (rteVRPex) which have already been produced in the previous functions. The outcome of that function is the new path
%(rteVRPex),  while the (tcVRPex) and the (rrVRPex) save the improved cost and the reduced rate respectively.
[ rteVRPex, tcVRPex,rrVRPex ] = vrpEXCHANGE( rteVRPex,d,fleet,dist,initTC );
% In the following function the 2-opt is implemented
rte2opt=rteVRPex; initTC = tcVRPex;
[ rr2opt ] = tsp2OPT( rte2opt,dist,initTC );
```

Appendix B: Algorithm for the initialization of the inputs of HA2

B.1 Algorithm in Matlab: mainHA2.m

```
% C contains the distance matrix
C = xlsread('distance_matrix_in_Excel.xlsx');
% The distance matrix saved in variable dist too.
dist=C;
% Define the truck's capacity. The vector can consist of one non negative number in case of a single vehicle otherwise can be a lot of % vehicles
fleet = [capacity];
% Define the demand for each node. The demand can be zero if the load of a node is a negligible amount or another non negative
% number. The first number must be always zero as the first node refers to the depot.
d = [0, demand];
% Define the frequency for each node and initialize the first index as zero because refers to the depot.
f = [0, frequency];
% Define the M days of the period and the available driving hours per day for each vehicle.
period = M; dailyLIMIT = max_NODES_per_day;
% The following function takes as input the distance matrix and produces the list of savings as Clarke & Write method defines.
saving_C = savingARRAY( dist );

% The second step of Clarke & Write classifies the list of savings in descend order.
```

```
sorted_C = sortARRAY( saving_C );
% Function freqCLUSTER classifies all the nodes in accordance with their frequency value. Then, sorts the nodes of highest frequency in
% accordance with the distance from the depot. Moreover, the rest groups sorted in relation to the last node of the precedent group of frequency.
% To produce that classes the vector of the frequencies and the distance matrix are needed to be set as inputs. The output printed in the vector as
% cluster1.
[ cluster1 ] = freqCLUSTER(f,dist);
% Function freqCLUSTER2 chooses individually nodes from cluster1, in order to place them inside the days of the period, creating a comprehensive
% schedule. The procedure of that insertion must be done with respect to the SWCPVRP constrains. Vector cluster1, variable f, constrain dailyLIMIT,
% demand d and the distance matrix C are needed as inputs to execute the commands of the current function.
function [ cluster2,truckNUM ] = freqCLUSTER2( period,f,cluster1,dailyLIMIT,d,C )
% Initialize the matrix in which the initial solutions will be saved.
r2 = [];
% TCHA2 will sum up the travel cost for each route.
TCHA2 = 0;
% This variable will count the vehicles that used in each iteration.
truckTOT = 0;
% Variable day run for every single day of the period in order to produce routes in them.
for day=1:size(cluster2,1)
```

```
% Check if in the current day there are nodes.
if ~isempty([cluster2{day,1}])
% From cluster2 select the nodes of the current day.
nodesID=[cluster2{day,:}];
% According to HA2 to solve the SWCPVRP, the method of "cluster-first route-second" is followed. Once the  nodes clustered and shared
% properly to days creating a schedule, the algorithm of Clarke & Wright  implemented in every single day to produce initial solutions.
%Therefore, this function for the nodes of each single day isolates the needed informations from the distance matrix to crate smaller
% distance matrices. In that way, it will be known not only the id of the nodes in each day but also the corresponding distance matrix. To run
% that function, the id of the nodes and the whole distance matrix of the graph are needed as inputs.
matrix=matCREATE( nodesID,dist );
% This function implements the first Clarke and Wright step which is the computation of savings. The saving's formula is s(i,j) = dist(i,0) % +
% dist(j,0) - dist(i,j).
saving_C=savingARRAY( matrix );
% This function takes as inputs the matrix of savings and produces all the pairs of nodes in a descent order list named sorted_C.
sorted_C = sortARRAY( saving_C );
% CWvrp function executes the Clarke and Wright method. Takes as inputs the matrix of distances, the list of savings, the demand of each node,
% the limit of visits in a day and variable day.
[ r,truck,totCOST,v ] = CWvrp( matrix,sorted_C,d,fleet,dailyLIMIT,day );
% Sum up the cost that produced from CWvrp.
TCHA2 = totCOST + TCHA2;
```

```
    % Sum up the vehicles.
    truckTOT = truck + truckTOT;
    % Variable cnt counts the routes of the current day one by one.
    cnt = 0;
      % For all field of schedule r..
      for rCELL = 1:size(r,2),
        % Increase cnt by one.
         cnt = cnt+1;
         % The selected route-field add it to r2 properly.
         r2{day,cnt} = [r{1,cnt}];
         % Also update the matrix v2 which contains the vehicles.
         v2{day,cnt} = [v{1,cnt}];
      end % End for rCELL.
    end % End if ~isempty.
end % End for day.
```

Appendix C: Algorithm for the matrix of savings (Clarke & Wright)

C.1 Algorithm in Matlab: savingARRAY.m

```
% This function implements the first clark and wright step which is the computation of savings. The saving's formula is s(i,j) = dist(i,0)
% + dist(j,0) - dist(i,j).
function [ s ] = savingARRAY( dist )
% rowsdist is the number of rows of matrix dist.
% colsdist is the number of columns of matrix dist.
[rowsdist, colsdist] = size(dist);
% we initialize the saving list
s = zeros(rowsdist, colsdist);
% For each row i of distance matrix.
for i = 1:rowsdi
   % For each column j of distance matrix.
   for j = 1: colsdist
      % Compute the value of saving between i, j and the depot=1.
      s(i,j) = (dist(i,1) + dist(1,j) - dist(i,j));

   end  % End for j
end % End for i
```

```
% Turn the diagonal of the s into zero, in case that the distance matrix is not symmetric.
[s(logical(eye(size(s))))] = 0;
% Delete the first row which refers to the distance between the depot and the other nodes.
s(1,:) = 0;
% Delete the first column which refers to the distance between the depot and the other nodes.
s(:,1) = 0;
```

Appendix D: Algorithm for the descend sort of saving matrix (Clarke & Wright)

D.1 Algorithm in Matlab: sortARRAY.m

```
% This function takes as inputs the matrix of savings and produces all the pairs of nodes in a descent order list named sorted_C.
function sorted_C = sortARRAY( saving_C )
% Sort the indexes of the saving matrix ascendingly and save the corresponding index of each value.
[~, i] = sort(saving_C(:));
% For each index of the matrix find its corresponding row and column in the matrix of savings.
[x,y] = ind2sub(size(saving_C),i);
% Save the new list of the immediately precedent command into variable z.
z=[x,y];
% Delete all pairs consists of same indexes such as (5,5) (6,6) etc.
z(z(:,1) == z(:,2), :) = [];
% Reverse the columns of matrix z, because the outcome must contain the biggest savings first.
sorted_C = flipud(z);
end % End function.
```

Appendix E: Algorithm for the main function of HA1

E.1 Algorithm in Matlab: CWpvrp.m

```
% Function CWpvrp produces initial solutions for the periodic-VRP using the Clarke & Write method.
function [ r,truckNUM, rtrnsOFvhcl, totCOST ] = CWpvrp( sorted_C,dist,fleet, d, f, period, dailyLIMIT )
% Initialize the name of the truck that is going to be used first.
TruckID = 1;
% Initilize the capacity of the truck as the first truck.
truck_cap=fleet(truckID);
% The total distance cost initialized as: totCOST.
TotCOST = 0;
% Count the number of periods we need to run in order to accomplish the frequencies of nodes.
CountDAYS=0;
% truckNUM counts how many times a truck is needed. Initially the first truck of the fleet is needed. Moreover, if that truck is not
% enough to execute the whole schedule within the period, then one more truck will be called.
TruckNUM=1;
% rtrnsOFvhcl is a matrix which count how many returns to the depot the vehicles have been made in order to unload the waste.
rtrnsOFvhcl=zeros(period,1);
% i reads all rows of the first column of sorted_C j reads all rows of the second column of sorted_C
i=sorted_C(:,1); j = sorted_C(:,2);

% n shows the length of dist
n=length(dist);
```

```
% initialize the variable day
day=0; r={};
% conseqCUST is a matrix, which shows if a node can be connected on a day or not in accordance with the scenario of consecutive
% visitations.
conseqCUST=zeros(period,n);
% assignCUST shows if a node is taking part in a route of a specific day
assignCUST=zeros(period,n);
% Initialize variable empty_days
emptyDAYS = zeros(1,n);
% emptyDAYS is the subtraction of the frequency of a node from the period. More specific defines how many days should be passed
% to visit a node. For instance if a node must be visited 6 times per period and the period was set at 6 six days, then (period-
% f(node)=0) which means that node must be visited every day. The first day of variable emptyDAYS is the depot and defined as zero.
for k = 1:n, emptyDAYS(1,k)=period-f(k); end; emptyDAYS(1,1)=0;
% While the frequencies are not exhausted continue.
while sum(f) ~= 1
    % in_vtx shows if a node is interior
    % b_vtx shows if a node is a beginning node of a route
    % e_vtx shows if a node is an ending node of a route
    [in_vtx,b_vtx,e_vtx] = deal(zeros(1,n));
    % eliminate all nodes with zero frequency
    for x = 2:length(f), if f(x) == 0, in_vtx(x) = 1; end; end
    % Check if the period increased If not continue counting days
    countDAYS=countDAYS+1;
```

```
    % If variable countDAYS exceed th limits of the period the following computations must be done
    if countDAYS==period+1
        % Set the countDAYS as 1, call a new available truck, set which vehicle is going to be used and update the capacity.
        countDAYS=1; truckNUM = truckNUM+1; truckID = truckID +1;
        % If the fleet is exhausted use the first vehicle and update
        if truckID>length(fleet), truckID = 1; truck_cap=fleet(truckID); else truck_cap=fleet(truckID);
        end % The check of the fleet
    end % The check of the period
    % A new day-route starts
    day = day + 1;
    % Set the real day
    realDAY=day-(period*(truckNUM-1));
    % Set the previous day of the current day
    if realDAY==1, prevDAY=1; else prevDAY=realDAY-1;end
    % rep shows the iterations of main loop
    rep = 1;
    % rte_num shows the number of the route
    rte_num = 0;
fprintf(' PVRP HA1: Day = %d, Vehicle = %d, ',realDAY, truckID);
fprintf(' Capacity = %d, ',fleet(truckID));
while rep <= length(i)
   % Both vtx i & j must not be  interior or assigned at the running day. Also both vtx i & j should not be asigned at previous day if its
   % possible.
```

```
if ~in_vtx(i(rep)) && ~in_vtx(j(rep)) && ~assignCUST(realDAY,i(rep)) && ~assignCUST(realDAY,j(rep))...
   && ~conseqCUST(prevDAY,i(rep)) && ~conseqCUST(prevDAY,j(rep))
  % i_status shows the position ofnode i
  i_status = b_vtx(i(rep)) + e_vtx(i(rep));
  % j_status shows the position of node j
  j_status = b_vtx(j(rep)) + e_vtx(j(rep));
  % Make new loc using vtx i & j
  if ~i_status && ~j_status && d(i(rep)) + d(j(rep)) <= truck_cap
    % Confirm if the route of day t has already been started, If yes compute the total number of bins.
    if size(r,1)==day && sum([c{day,:}]) + d(i(rep)) + d(j(rep)) <= dailyLIMIT
      % creation of new route
      rte_num = rte_num + 1;
      % insertion of nodes
      r{day,rte_num} = [i(rep) j(rep)];
      % update the capacity of the route
      c{day,rte_num} = d(i(rep)) + d(j(rep));
      % label the i node as beginning node
      b_vtx(i(rep)) = rte_num;
      % label the j node as ending node
      e_vtx(j(rep)) = rte_num;
      % frequency of node i reduced by one
      f(i(rep)) = f(i(rep)) – 1;
      % frequency of node j reduced by one
```

```
        f(j(rep)) = f(j(rep)) - 1;

    % If the first route of the day starts include the pair of nodes
    elseif size(r,1)<day
        % creation of new route
        rte_num = rte_num + 1;
        % insertion of nodes
        r{day,rte_num} = [i(rep) j(rep)];
        % update the capacity of the route
        c{day,rte_num} = d(i(rep)) + d(j(rep));
        % label the i node as begining node
        b_vtx(i(rep)) = rte_num;
        % label the j node as ending node
        e_vtx(j(rep)) = rte_num;
        % frequency of node i reduced by one
        f(i(rep)) = f(i(rep)) – 1;
        % frequency of node j reduced by one
        f(j(rep)) = f(j(rep)) - 1;
    end
% Add vtx j to loc i
elseif i_status && ~j_status && e_vtx(i(rep)) && c{day,i_status} + d(j(rep)) <= truck_cap && sum([c{day,:}]) + d(j(rep)) <= dailyLIMIT
    % place node j in the corresponding route
    r{day,i_status} = [r{day,i_status} j(rep)];
```

```
    % update the capacity of the route
    c{day,i_status} = c{day,i_status} + d(j(rep));

    % node j became the ending node
    e_vtx(j(rep)) = i_status;
    % node i became interior
    in_vtx(i(rep)) = 1;
    % frequency of node j reduced by one
    f(j(rep)) = f(j(rep)) - 1;
% Add  vtx i to loc j
elseif ~i_status && j_status && b_vtx(j(rep)) && c{day,j_status} + d(i(rep)) <= truck_cap...
        && sum([c{day,:}]) + d(i(rep)) <= dailyLIMIT
    % place node i in the corresponding route
    r{day,j_status} = [i(rep) r{day,j_status}];
    % update the capacity of the route
    c{day,j_status} = c{day,j_status} + d(i(rep));
    % node i became the beginning node
    b_vtx(i(rep)) = j_status;
    % node j became interior
    in_vtx(j(rep)) = 1;
    % frequency of node i reduced by one
    f(i(rep)) = f(i(rep)) - 1;
    % Combine loc i & j
```

```
  elseif i_status && j_status && i_status ~= j_status && e_vtx(i(rep)) && b_vtx(j(rep))...
        && c{day,i_status} + c{day,j_status} <= truck_cap

    % the last node of route j_status changes route
    e_vtx(r{day,j_status}(end)) = i_status;
    % we connect 2 routes into one
    r{day,i_status} = [r{day,i_status} r{day,j_status}];
    % update the capacity of the route
    c{day,i_status} = c{day,i_status} + c{day,j_status};
    % the route j_status has been merged in another route
    r{day,j_status} = [];
    % the deleted route has no weight
    c{day,j_status} = [];
    % both nodes i and j became interior
    in_vtx([i(rep) j(rep)]) = 1;
    % Can't make new loc seq
  else
    replot = 0;
  end
end
% rep increasing by one
rep = rep + 1;
% Re-set in case i(rep) = 1 or j(rep) = 1 was added to loc
```

```
  b_vtx(1) = 0; e_vtx(1) = 0;
  % reset the frequency of depot
  f(1) = 1;
end   % while
% The vector with visits for the current day
if size(r,1) == day
   rMAT = cell2mat(r(day,:));
   % compute the total cost of the new route of day x
   rCOST = sum(diag(dist(rMAT(1:end-1),rMAT(2:end) )));
   % Sum the costs in order to find the total cost
   totCOST = totCOST + rCOST;
else rCOST = 0;
end
fprintf(' Cost = %0.f \n',rCOST);
% In the following loop the returns of the vhcl to the depot will be computed.
if size(r,1) == day
   TESTEDr=r(day,:); idx=sum(~cellfun(@isempty,TESTEDr),2);
   rtrnsOFvhcl(realDAY,1) = rtrnsOFvhcl(realDAY,1) + idx;
else rtrnsOFvhcl(realDAY,1) = 0;
end
if size(r,1)==day
   % If a node did not insert at the current day, update the corresponding
   % column of matrix "emptyDAYS".
```

```
   for m=2:n, index=find([r{day,:}]==m,1); if isempty(index) &&...
      emptyDAYS(1,m)>0, emptyDAYS(1,m)=emptyDAYS(1,m)-1; end; end
   % If a node inserted at the current day, update the corresponding
   % column of matrix "conseqCUST".
   v=cell2mat(r(day,:)); rCOL=size(v,2); for m=1:rCOL, conseqCUST(realDAY,v(m))=1; assignCUST(realDAY,v(m))=1;end;


    conseqCUST(realDAY,1)=0; assignCUST(realDAY,1)=0;
elseif size(r,1)<day
   for m=2:n, emptyDAYS(1,m)=emptyDAYS(1,m)-1; end;
end
% conseqCUST defined as the matrix of the working days (rows) X nodes, while the indices of that matrix are binary. 1 shows if a
% specific day can be visited by a specific node and 0 otherwise.
for m=2:n, if ~emptyDAYS(m), conseqCUST(realDAY,m)=0; end; end
end
% In the following loops the fields with informations (non-empty) will be kept and the fields do not contain informations will be eliminated.
% r2 is defined as a trial matrix of matrix r
r2={};
for i=1:size(r,1), counter=0; for j=1:size(r,2), if ~isempty(r{i,j})
   counter=counter+1; r2{i,counter}=r{i,j}; end; end; end
r=r2;
% In the following loops the fields with information (non-empty) will be kept and the fields that not contains infos will be eliminated
% numOFselectedNDSha1 is defined as a trial matrix of matrix c
numOFselectedNDSha1={};
```

```
for i=1:size(c,1), counter=0; for j=1:size(c,2), if ~isempty(c{i,j})
   counter=counter+1; numOFselectedNDSha1{i,counter}=c{i,j}; end; end; end
c=numOFselectedNDSha1;
% Sum the retrurnes of the vehicles
rtrnsOFvhcl = sum(rtrnsOFvhcl);

fprintf(' \n PVRP HA1: Total Cost = %0.f \n',totCOST);
end
```

Appendix F: Algorithm for the first cluster of nodes in HA2

F.1 Algorithm in Matlab: freqCLUSTER.m

```
% Function freqCLUSTER classifies all the nodes in accordance with their frequency value. Then, sorts the nodes of highest frequency % in accordance with the distance from the depot. Moreover, the rest groups sorted in relation to the last node of the precedent group % of frequency. To produce that classes the vector of the frequencies and the distance matrix are needed to be set as inputs. The
% output printed in the vector as cluster1.
function [ cluster1 ] = freqCLUSTER(f,dist)
 % The first node refers to the depot and deleted because has no frequency value.
f(1)=[];
% The first important information is how many different types of frequencies do the nodes have.
freqNUM=fliplr(unique(f));
% Once the information in the previous command taken, a reestablishment of vector f is made.
f=[1 f];
% finNODE refers to the last node of cluster1, under which the next nodes will be sorted ascendingly. Initially, vector cluster1 will be % empty, so the nodes of highest frequency will be sorted in accordance with the distance from the depot. The depot is node 1 so set % finNODE as 1.
finNODE=1;
% cluster1 will save the id of the nodes in the right order with respect to their frequency.
cluster1=[];
% The next for loop will start running for all different frequencies.
for i=1:length(freqNUM)
   % selectROW selects that row from the distance matrix, which defined in variable finNODE.
   selectROW=dist(finNODE,:);
```

```
    % In the following line, function sort places the distances of vector selectROW ascendingly in relation to the depot, while variable
    % col saves the changes that produced to the columns of the vector.
    [~,col]=sort(selectROW(:));
    % Convert col into one row.
    col=col';
    % The next for loop will run for all indexes of col.
    for j=2:length(col)
       % If the selected node j has same frequency with freqNUM and is not the depot, add it to the cluster.
       if freqNUM(i)==f(col(j)) && col(j)~=1
          % cluster1 will extended by the selected node j.
           cluster1=cat(2, cluster1, col(j));
       end End if  freqNUM.
    end End for j.
    % Now set the last node of cluster1 to proceed.
    finNODE=cluster1(end);
end % End for I.
 end % End function.
```

Appendix G: Algorithm for the second cluster of nodes in HA2

G.1 Algorithm in Matlab: freqCLUSTER2.m

```
% Function freqCLUSTER2 chooses individually nodes from cluster1, in order to place them inside the days of  the period, creating a
% comprehensive schedule. The procedure of that insertion must be done with respect to % the SWCPVRP constrains. Vector cluster1, variable f,
% constrain dailyLIMIT, demand d and the distance matrix C are needed as inputs to execute the commands of the current function.
function [ cluster2,truckNUM ] = freqCLUSTER2( period,f,cluster1,dailyLIMIT,d,C )
% truckNUM counts how many times a vehicle is needed.
truckNUM=1;
% counter computes how many days have been passed before the current day.
counter=0;
% day shows the current day.
day=0;
% The number of rows of variable c is equal with the M days of the period. Each row is corresponded to each
% day of the period and computed by the number of the shifts multiplied with the load of the truck.
c=zeros(period,1);
% cluster2 is the variable which contains the schedule of visits, consisting of rows equal to the M days of the
% period and one column. The column in each row contains the id pf the nodes which have to be visited on that % specific day.
cluster2=cell(period,1);
% n represents the total number of nodes that included up to the whole graph.
n=length(C);
% assignCUST indicates if a specific point has been visited in a specific day.
assignCUST=zeros(period,n);
```

```
% conseqCUST is a matrix, which shows if a node can be connected on a day or not in accordance with the scenario of consecutive
% visitations.
conseqCUST=zeros(period,n);
% Initialize variable empty_days
emptyDAYS = zeros(1,n);counter2=0;
% emptyDAYS is the subtraction of the frequency of a node from the period. More specific defines how many days should be passed
% to visit a node. For instance, if a node must be visited 6 times per period and the period was set at 6 six days, then (period-
% f(node)=0) which means that node must be visited every day. The first day of variable emptyDAYS is the depot and defined as zero.
for k = 1:n, emptyDAYS(1,k)=period-f(k); end; emptyDAYS(1,1)=0;
 % While f is not exhausted the schedule is not completed.
while sum(f)~=1
   % count how many days passed including the current day from the begging of the current period.
   counter=counter+1;
   % if variable counter has exceeded the length of the set period.
   if counter==period+1,
         % reset counter to 1.
         counter=1;
         % Increase  truckNUM by one.
         truckNUM=truckNUM+1;
         Reestablish variable day to 1.
         day=1;

   % Other wise, just go to the next unmet day.
```

```
else day=day+1;
end; % end if
% Now start scanning one by one the nodes from vector cluster1.
for i=1:length(cluster1)
  % Now check if node i can feasibly be added in the current day respecting the restrictions of repeated nodes on same day, avoid consecutive
  % visitations, capacity and frequency.
  if ~assignCUST(day,cluster1(i)) && f(cluster1(i))>0 && c(day,1)+d(cluster1(i))<=truckNUM*dailyLIMIT…
       && ~conseqCUST(day,cluster1(i)),
      % add the node in the appropriate field
      cluster2{day,1}=[cluster2{day,1} cluster1(i)];
      % update the demand
      c(day,1)=c(day,1)+d(cluster1(i));
      % update the frequency
      f(1,cluster1(i))=f(1,cluster1(i))-1;
      % assign the node to the corresponding day
      assignCUST(day,cluster1(i))=1;
     % update variable emptyDAYS
     emptyDAYS(1,cluster1(i))=emptyDAYS(1,cluster1(i))-1;
     %  If the node we met has still empty days until its next visit update the conseqCUST properly.
     if day<period && emptyDAYS(1,cluster1(i))>0,
        % Update of  conseqCUST as 1, which means the next day will be avoided the insertion of the node of the current repetition.
        conseqCUST(day+1,cluster1(i))=1;
      else conseqCUST(day,cluster1(i))=0;
```

```
            end % End if day
        end % End if assign …
    end % End for I
end % End while sum
end % End Function
```

Appendix H: Algorithm for the main function of HA2

H.1 Algorithm in Matlab: freqCLUSTER2.m

```
% CWvrp function executes the Clarke and Wright method. Takes as inputs the matrix of distances, the list of savings, the demand of each node,
% the limit of visits in a day and variable day.
function [ r,truck,totCOST,v ] = CWvrp( matrix,sorted_C,d,fleet,dailyLIMIT,day )
 % Initilize the total cost.
totCOST = 0;
% When the shift of a truck completed, and the initial solutions have not been produced comprehensively a new or even the same truck of the fleet
% is needed. truckID counts how many times that need appears through the period.
truckID = 0;
 % i reads all rows of the first column of sorted_C.
% j reads all rows of the second column of sorted_C.
i=sorted_C(:,1); j = sorted_C(:,2);
% n shows the length of dist.
n=length(matrix);
% Initilize the variable truck as 0. This variable increased when the current vehicle completes its shift.
truck=0;
% in_vtx shows if a node is interior
% b_vtx shows if a node is a beginning node of a route
% e_vtx shows if a node is an ending node of a route
[in_vtx,b_vtx,e_vtx] = deal(zeros(1,n));
% The vector f.
```

```
f=ones(1,n);
% Initialize capacity c as an empty cell.
c = {[]};
% Initialize vehicle id matrix c as an empty cell.
v = {[]};

% While f is not exhausted.
while sum(f)~=1
    % rep shows the iterations of main loop
    % rte_num shows the number of the route
    rep = 1; rte_num = 0;
    % If f did not exhaust select a truck.
    truck = truck + 1;
    % If the entire fleet used and f is not exhausted use again the first vehicle and update.
    if truckID >= length(fleet),
        % Use the first vehicle.
        truckID = 1;
        % Select the first vehicle from the fleet.
        truck_cap = fleet(truckID);
    else
        % If the fleet has still vehicles to give go to the next one.
        truckID = truckID + 1;
        % Select the next vehicle from the fleet.
```

```
        truck_cap=fleet(truckID);
    end
    % Eliminate all nodes with zero frequency.
    for x = 2:length(f), if f(x) == 0, in_vtx(x) = 1; end; end
fprintf(' PVRP HA2: Day = %d, Vehicle = %d, ',day, truckID);
fprintf(' Capacity = %d, ',fleet(truckID));
    % While all the pair of savings have not scanned continue
    while rep <= length(i)
        % If the pair do not contain  any  interior node at all continue.
        if ~in_vtx(i(rep)) && ~in_vtx(j(rep))
            % i_status shows the position of node i.
            i_status = b_vtx(i(rep)) + e_vtx(i(rep));
            % j_status shows the position of node j.
            j_status = b_vtx(j(rep)) + e_vtx(j(rep));
           % Make new loc using vtx i & j.
            if ~i_status && ~j_status && d(i(rep))+d(j(rep))<=truck_cap
                % Firstly, test the capacity of the route.
                c2=c; rte_num2=rte_num; rte_num2=rte_num2+1;
                c2{truck,rte_num2} = d(i(rep)) + d(j(rep));

                if sum([c2{truck,:}]) <= dailyLIMIT
                    % creation of new route.
                    rte_num = rte_num + 1;
```

```
        % insertion of nodes.
        r{truck,rte_num} = [i(rep) j(rep)];
        % update the capacity of the route.
        c{truck,rte_num} = d(i(rep)) + d(j(rep));
        % Note the vehicle.
        v{rte_num} = truckID;
        % Make the i node beginning node.
        b_vtx(i(rep)) = rte_num;
        % Make the j node ending node.
        e_vtx(j(rep)) = rte_num;
        % Frequency of node i reduced by one.
        f(i(rep)) = f(i(rep)) - 1;
        % Frequency of node j reduced by one.
        f(j(rep)) = f(j(rep)) - 1;
    end
% Add vtx j to loc i
elseif i_status && ~j_status && e_vtx(i(rep)) && c{truck,i_status} + d(j(rep)) <= truck_cap && sum([c{truck,:}]) + d(j(rep)) …
            <= dailyLIMIT

    % Place node j in the corresponding route
    r{truck,i_status} = [r{truck,i_status} j(rep)];
    % Update the capacity of the route.
    c{truck,i_status} = c{truck,i_status} + d(j(rep));
```

```
    % Node j became the ending node.
    % Sign the vehicle.
    v{i_status} = truckID;
    e_vtx(j(rep)) = i_status;
    % Node i became interior.
    in_vtx(i(rep)) = 1;
    % Frequency of node j reduced by one.
    f(j(rep)) = f(j(rep)) - 1;


% Add  vtx i to loc j
elseif ~i_status && j_status && b_vtx(j(rep)) && c{truck,j_status} + d(i(rep)) <= truck_cap && sum([c{truck,:}]) + d(i(rep)) …
            <= dailyLIMIT

    % Place node i in the corresponding route.
    r{truck,j_status} = [i(rep) r{truck,j_status}];
    % Update the capacity of the route.
    c{truck,j_status} = c{truck,j_status} + d(i(rep));
    % Note the vehicle.
    v{j_status} = truckID;
    % Node i became the beginning node.
    b_vtx(i(rep)) = j_status;
    % Node j became interior.
```

```
    in_vtx(j(rep)) = 1;
    % Frequency of node i reduced by one.
    f(i(rep)) = f(i(rep)) - 1;

% Combine loc i & j
elseif i_status && j_status && i_status ~= j_status && e_vtx(i(rep)) && b_vtx(j(rep)) && c{truck,i_status} + c{truck,j_status} …
        <= truck_cap

    % The last node of route j_status changes route.
    e_vtx(r{truck,j_status}(end)) = i_status;
    % Connect the 2 routes into one.
    r{truck,i_status} = [r{truck,i_status} r{truck,j_status}];
    % Update the capacity of the route.
    c{truck,i_status} = c{truck,i_status} + c{truck,j_status};
    % The route j_status has been merged in another route.
    r{truck,j_status} = [];
    % The deleted route has no weight
    c{truck,j_status} = [];
    % Deleted vehicle from the deleted route.
    v{j_status} = [];
    % Both nodes i and j became interior.
    in_vtx([i(rep) j(rep)]) = 1;
```

```
            end % End if ~i_status...
        end % End if ~in_vtx...
        % rep increasing by one.
        rep = rep + 1;
        % Re-set in case i(rep) = 1 or j(rep) = 1 was added to loc.
        b_vtx(1) = 0; e_vtx(1) = 0;
        % reset the frequency of depot
        f(1) = 1;
    end % End while rep.
    % Save the vector of visits.
    mat = cell2mat(r(truck,:));
    % Save the cost of the mat.
    rCOST = sum(diag(matrix(mat(1:end-1),mat(2:end))));
    % Sum the distance cost.
    totCOST = totCOST + rCOST;
fprintf(' Cost = %0.f \n',rCOST);
end % End while f.
    % In the following loops the non-empty fields will be kept, and the empty fields will be not signed in the schedule with the routes.
    r2 = {}; v2 = {}; counter=0;
    for i=1:size(r,1)
        for j=1:size(r,2),
            if ~isempty(r{i,j})
                counter=counter+1;
```

```
            r2{1,counter} = r{i,j};
            if i <= length(fleet), v2{1,counter} = i; else
              v2{1,counter} = i - length(fleet); end
          end
        end
    end
r = r2;
 v = v2;
end
```

Appendix I: Algorithm for the vrp-exchange

I.1 Algorithm in Matlab: vrpEXCHANGE.m

```
% Function vrpEXCHANGE executes the vrp-exchange method of local search algorithms. The following function takes as input the
% travel cost (initTC)of the  route (rteVRPex)  which have already been produced in the previous functions. The outcome of that
% function includes the new path (rteVRPex) and also the (tcVRPex) and the (rrVRPex) which save the improved cost and the reduced
% rate respectively.
function [ rteVRPex,tcVRPex,rrVRPex ] = vrpEXCHANGE( rteVRPex,d,fleet,dist,initTC )
% Initilize the total cost as 0
tcVRPex = 0;
% done is a parameter which stops the loops
done = 0;
% Done is a matrix of binary indexes which shows the connection between the routes, if two routes tested by vrp-exchange the
% corresponding index becomes 1. For example, if all possible change between route 1 & 2 have been made then the index (1,2) and
% (2,1) of the matrix is true, otherwise is false.
Done = false(numel(rteVRPex));
done initially is zero.
while ~done
    % done is initializing as 1.
    done=1;
    % In the following loop choose a specific route.
    for i = 1:numel(rteVRPex)-1
        % The row of the route.
```

```
  [rteROWi,~] = ind2sub(size(rteVRPex),i);

% In the following loop choose another route
for j = i+1:numel(rteVRPex)
   % In which row the route I places can be found in the next line.
   [rteROWj,~] = ind2sub(size(rteVRPex),j);
   % Continue if and only if the routes are not empty
   if ~isempty(rteVRPex{i}) && ~isempty(rteVRPex{j})
      % Choose a node from route I.
      for m = 2:length(rteVRPex{i})-1

         % Choose a node from route j.
         for n = 2:length(rteVRPex{j})-1
            % Implement the exchange as trial.
            rteTRIALim = [rteVRPex{i}(1:m-1) rteVRPex{j}(n) rteVRPex{i}(m+1:end)];
            % Implement the exchange as trial.
            rteTRIALjn = [rteVRPex{j}(1:n-1) rteVRPex{i}(m)  rteVRPex{j}(n+1:end)];
            % Check if the new route im exceeds the capacity.
            totDMNDim = sum(d(rteTRIALim(1:end)));
         % Check if the new route jn exceeds the capacity.
            totDMNDjn = sum(d(rteTRIALjn(1:end)));
         % Update the variables capIDX if the criteria of capacity are met.
            if totDMNDim <= fleet(1) && totDMNDjn <= fleet(1), capIDX=1; else capIDX=0; end
```

```
    % Check if a node i is repeated in a single day.
    vecNODESi = cell2mat(rteVRPex(rteROWi,:));
% Check if a node j is repeated in a single day.
    vecNODESj = cell2mat(rteVRPex(rteROWj,:));



    % First, delete the depots from route i which will be tested for repeated nodes.
    vecNODESi(vecNODESi==1)=[];
% First, delete the depots from route j which will be tested for repeated nodes.
vecNODESj(vecNODESj==1)=[];
% In the following line length of routes, I and j is checked.
 if length(vecNODESi) == length(unique(vecNODESi)) && length(vecNODESj) == length(unique(vecNODESj))
     repIDX = 1; else repIDX = 0;
 end
  % distNEim is the summation of the fake route I after having add node m.
   distNEWim = sum(diag(dist(rteTRIALim(1:end-1), rteTRIALim(2:end))));
   % distNEjn is the summation of the fake route j after having change node n.
    distNEWjn = sum(diag(dist(rteTRIALjn(1:end-1), rteTRIALjn(2:end))));
 % distNEWtot sums up the two fake routes.
    distNEWtot = distNEWim + distNEWjn;
    % distOLDim is the summation of the current route i.
    distOLDim = sum(diag(dist(rteVRPex{i}(1:end-1), rteVRPex{i}(2:end))));
    % distOLDjn is the summation of the current route j.
```

```
                distOLDjn = sum(diag(dist(rteVRPex{j}(1:end-1), rteVRPex{j}(2:end))));
            %  distOLDtot sums up the total travel cost for the current routes
                distOLDtot = distOLDim + distOLDjn;
            %  if the new is better than the old and all the constrains implement the change.
                if distNEWtot < distOLDtot && capIDX && repIDX
                 % route i replaced by the new one
                    rteVRPex{i} = rteTRIALim;
                 % route j replaced by the new one
                    rteVRPex{j} = rteTRIALjn;
                 % One change made, so may be others
                    done = 0;
                 % Once a change has been done updated as 0 to try more feasible changes
                    Done([i j],:) = 0; Done(:,[i j]) = 0;
                end % End if
            end % End for n
        end % End for m
    end % End if isempty
  end % End for j
 end % End for i
end % End while
for day = 1:size(rteVRPex,1)
 % Save in a mat vector the visits of day i
 matVRPex = cell2mat(rteVRPex(day,:));
```

```
    % Save the cost for day i
    costVRPex = sum(diag(dist(matVRPex(1:end-1),matVRPex(2:end))));
    % Sum the cost and save it
    tcVRPex = tcVRPex + costVRPex;
end
% Compute the rate of reduction
rrVRPex = ((initTC - tcVRPex) / initTC)*100;
fprintf('\n VRP Exchange: TC = %0.f, Reduced Rate = %f \n',tcVRPex,rrVRPex);
end
```

Appendix J: Algorithm for the 2-opt

J.1 Algorithm in Matlab: tsp2OPT.m

```
% tsp2opt belongs to intra-route category and is one of the most popular local search methods. The main procedure of the function is
% to select all the routes of the schedule individually in order to change the position of the nodes reducing the total travel cost. That function takes
as inputs the rte2opt which is the schedule with the initial solutions, variable dist in which includes the distance matrix and the initCOST which
has the travel cost of all routes.
function [ rr2opt ] = tsp2OPT( rte2opt,dist,initTC )
% Initialize the total cost before the implementation of 2-opt.
tc2opt = 0;
% The following for loop goes to each row of the schedule.
for day = 1:size(rte2opt,1)
   % The next for loop goes to each field – route.
   for k = 1:numel(rte2opt)
      % If the chosen field is not empty must be checked to proceed.
      if ~isempty(rte2opt{k})
         % To make changes in the route, the field must be transformed to a vector.
         route = rte2opt{k};
         % Variable enough terminates the whole procedure.
         enough=false;

         % Initially enough is equal to false so while – loop will start working.
         while ~enough
```

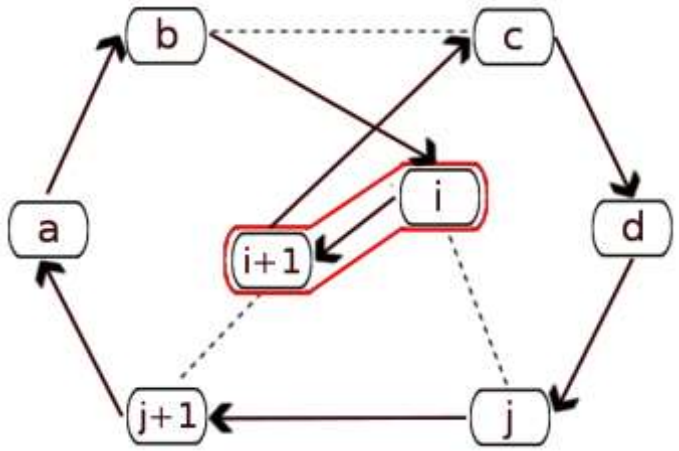
```
% Set enough as true.
enough = true;
% The next for loop selects every node from the vector.
for i = 1:length(route)-3
  % Variable j is equal to i+2, because in the previous loop we set i starting from the first node of the vector which is this
  % depot. We could also set j equal to i+1 if i has started from the second node and not from the first one to avoid
  % swapping the depot.
   for j = i+2:length(route)-1
     % Implement the exchange by swapping i+1 with j but do not save the result yet.
      rteTRIAL = [route(1:i) route(j) route(i+2:j-1) route(i+1) route(j+1:end)];



      % Sum up the distance of the new fake route.
      distNEW = sum(diag(dist(rteTRIAL(1:end-1), rteTRIAL(2:end))));
      % Sum up the distance of the current existing route.
      distOLD = sum(diag(dist(route(1:end-1), route(2:end))));
      % Compare the two distance costs.
      if distNEW < distOLD
         % Save the exchange in case of a good cost change.
         route = rteTRIAL;
          % Set enough as false
         enough = false;
      end % End if
```

```
                % If an exchange made break the j.
                 if ~enough, break, end
              end % End for j
               % If an exchange made break the i.
              if ~enough, break, end
            end % End for i
            % Save the change
            rte2opt{k} = route;
          end % End while
        end % End isempty
    end % End for k
     % Transpose the routes of the whole day in a vector.
    mat = cell2mat(rte2opt(day,:));
    % Compute the distance cost of the day.
    cost2opt = sum(diag(dist(mat(1:end-1),mat(2:end) )));
    % Sum the cost of the day with the converted one.
    tc2opt = cost2opt + tc2opt;
end % End for day
% Compute the rate of reduction.
rr2opt = ((initTC- tc2opt) / initTC)*100;
% Print the result.
fprintf('\n 2-opt: TC = %0.f, Reduced Rate = %f \n',tc2opt, rr2opt);
end % End function
```

Appendix K: Further improvement heuristics

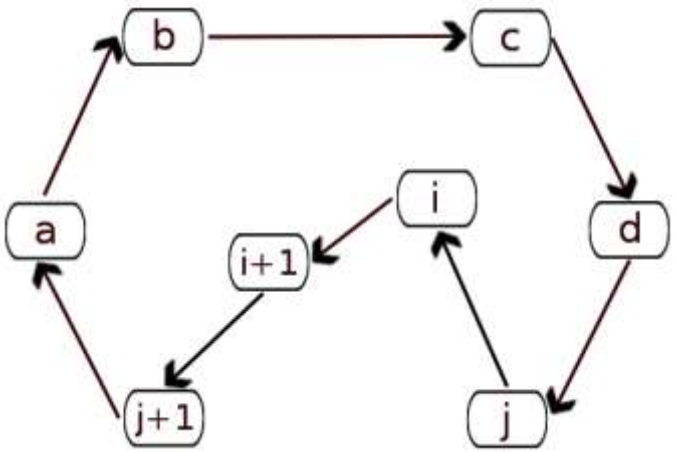
K.1 Or-opt [Intra Route]

The Or-opt, as in the λ-opt can be applied only in a problem of TSP form. This means that all possible changes can be performed only in a single route. The idea of this case is pretty much different in comparison with λ-opt because the selection of the customers is associated with consecutive arcs and not with single nodes. The execution of this method implies the deletion of consecutive nodes from their current position and presupposes a relocation into another position in the same route. For better understanding of this method, we consider the next example: ROUTE = [a → b → i → i + 1 → c → d → j → j + 1 → a].



Schematic illustration of initial solution for a single route problem with 8 nodes, 1 vehicle and 1 depot

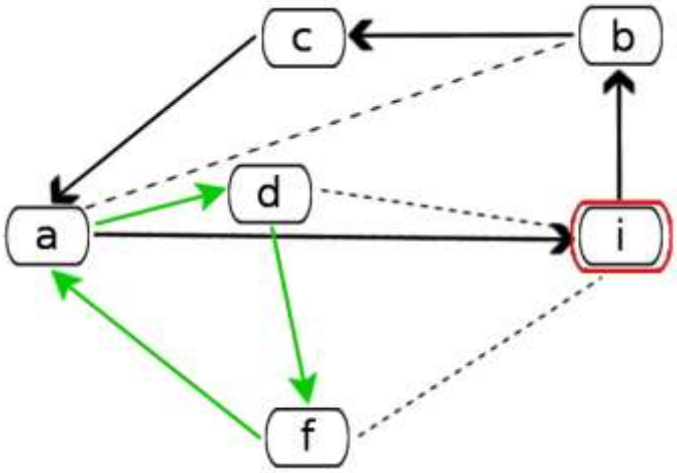
Initially, we can select a chain consisting of two or more continuing arcs. For that reason, we select the edge (i, i + 1) and relocate it into another position of the route. For instance, if we place it between node (j) and (j + 1), the result will be the following: ROUTE = [a → b → c → d → j → i → i + 1 → j + 1 → a].



Representation of route (Figure 6.3) following the application of Or-opt algorithm

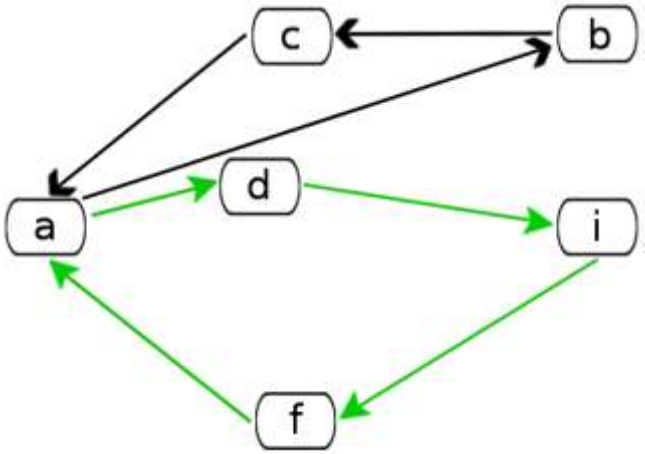
K.2 Relocate – Operator [Inter Route]

This method belongs to inter route category. In this algorithm two different routes are taking part in each repetition. On the first route one position is deleted and the length of the route is getting smaller and on the other route a new position is created. This application can be performed by removing a customer from a tour and replace it in the middle of the second tour. With this method it must be ensured that there will be no exceed of visits per day. The following example shows the first iteration of this method: ROUTE 1 = [a → i → b → c → a] and ROUTE 1 = [a → d → f → a].



Schematic illustration of initial feasible solution for a multi route problem with 6 nodes, 1 vehicle and 1 depot

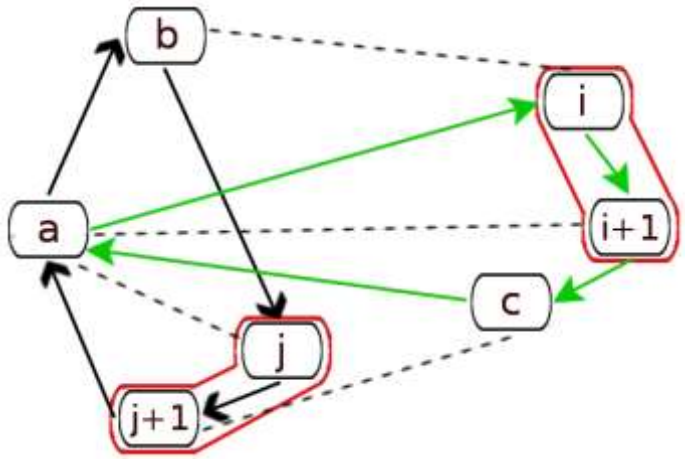
Customer i is selected from ROUTE 1 and deleted. Then the same customer goes between customer (d) and customer (f) from the second route. The result is shown in the next figure: ROUTE 1 = [a → b → c → a] and ROUTE 2 = [a → d → i → f → a].



Representation of route (Figure 6.7) following the application of Relocate-operator algorithm

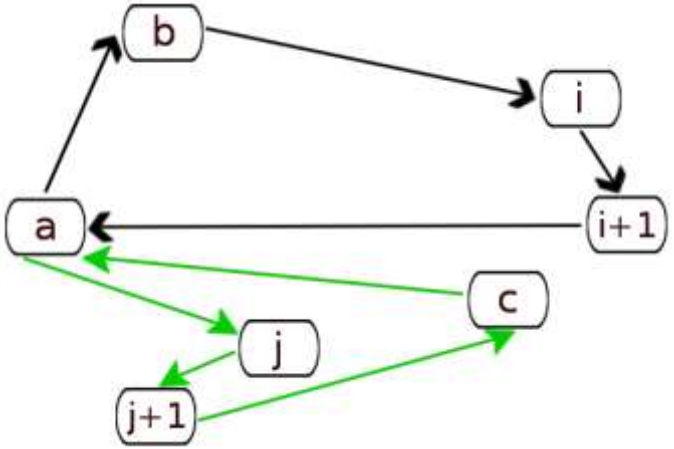
K.3 Cross – Exchange [Inter Route]

Cross-exchange model works as Or-opt from Intra-Route category. The purpose is to select two parts of consecutive nodes and exchange them with each other. It should be noted that each part of the nodes must include at least two nodes, otherwise the form of the model is the same with exchange operator. Furthermore, the selected parts can't have different length of arcs. After implementation, if the total travel cost decreases, the appropriate change must be saved, otherwise it has to be rejected. Let's see the following case for further understanding: ROUTE 1 $=$ [a $\rightarrow$ b $\rightarrow$ j $\rightarrow$ j + 1 $\rightarrow$ a] and ROUTE 2 $=$ [a $\rightarrow$ i $\rightarrow$ i + 1 $\rightarrow$ c $\rightarrow$ a].



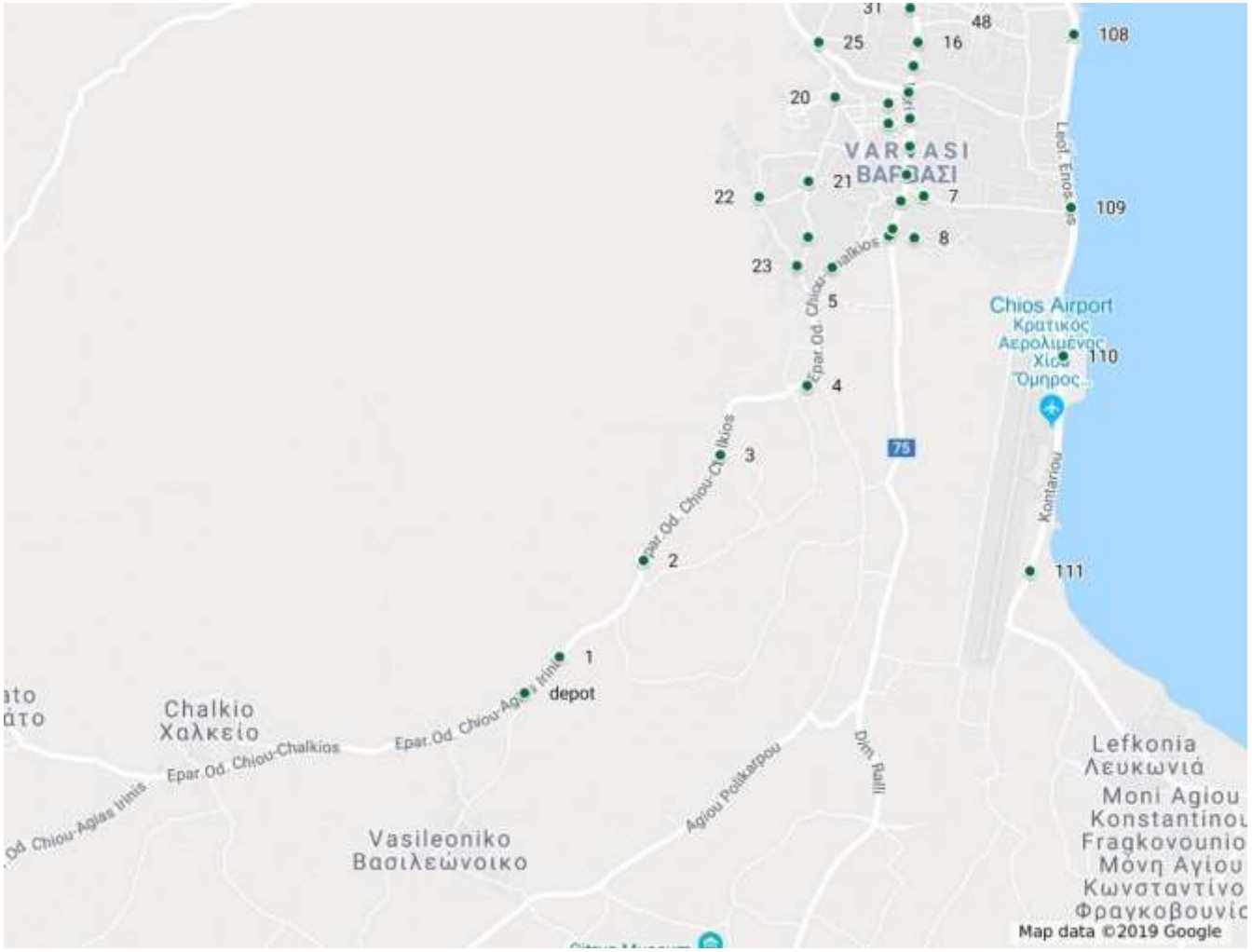
*Schematic illustration of **i**nitial feasible solution for a multi route problem with 7 nodes, 1 vehicle and 1 depot*

A double node chain is selected from each of the two available routes. The edge $(j, j+1)$and $(i, i+1)$ are selected from ROUTE 1 and ROUTE 2 respectively. The swapping of those edges converts the routes as: ROUTE 1 $=$ [a $\rightarrow$ b $\rightarrow$ i $\rightarrow$ i + 1 $\rightarrow$ a] and ROUTE 2 $=$ [a $\rightarrow$ j $\rightarrow$ j + 1 $\rightarrow$ c $\rightarrow$ a].



Representation of route (Figure 6.9) following the application of Cross-exchange algorithm

Appendix L: Maps with all nodes of plastic bins



In this figure is presented the depot which located in village Chalkeios.
The rest geographical points are located in the neighborhood of Varvasi and around the airport called "Omiros"

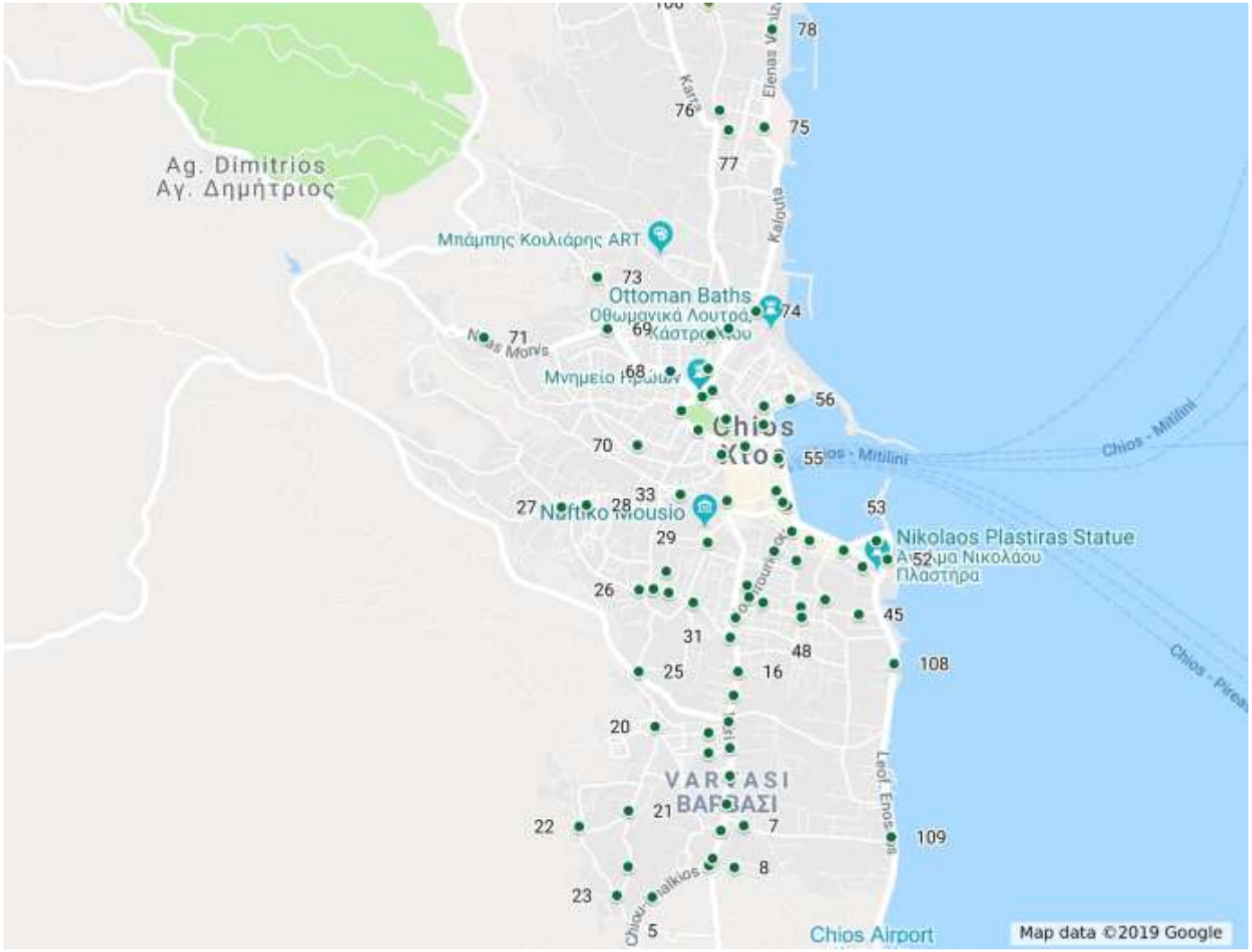


Figure L2. *This figure is the continuation of the previous map (Figure L1), while some geographical points are same in both maps. This area placed in the central city of Chios.*

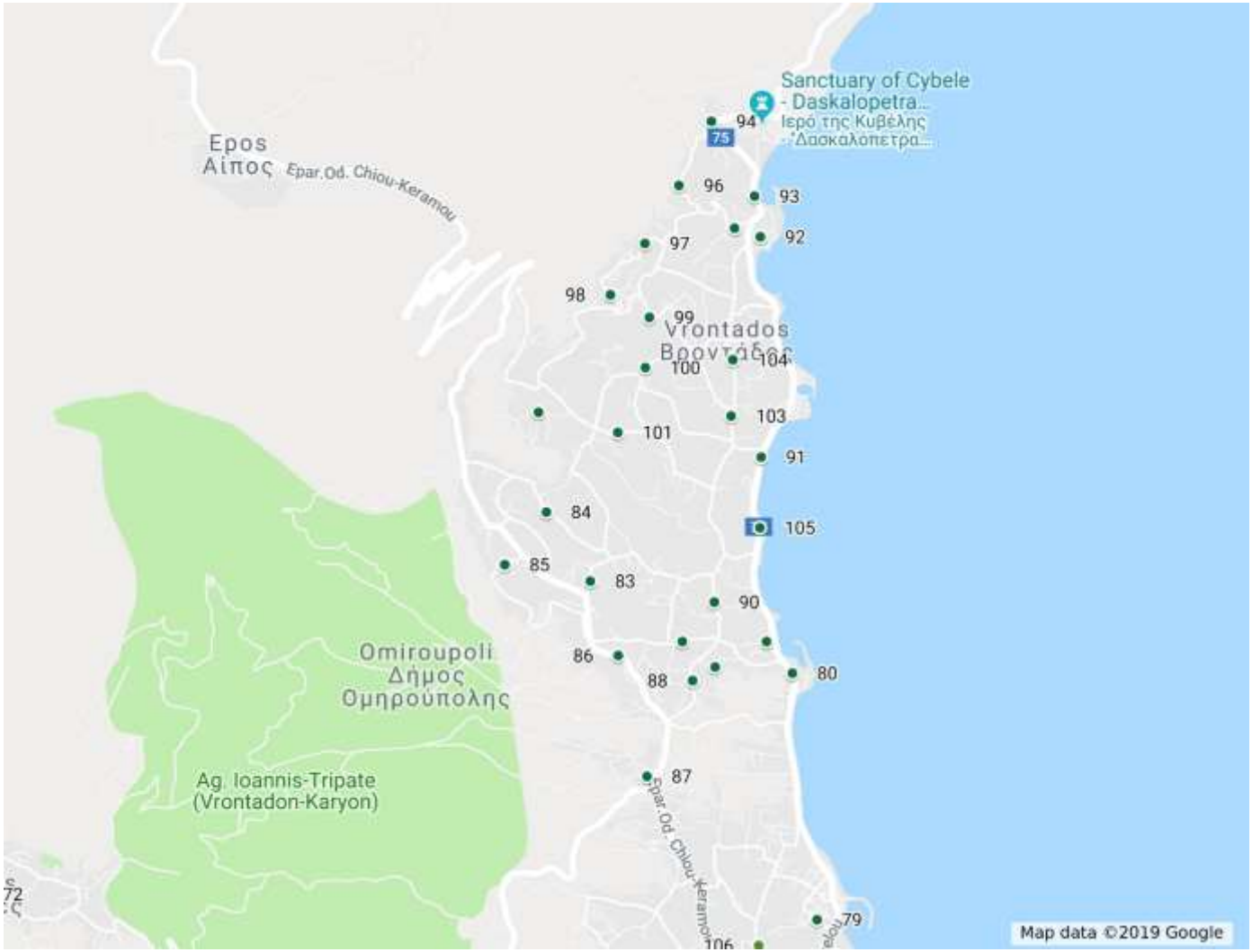


Figure L3. *This figure is the continuation of the previous map (Figure L2). Some geographical points are the same in both maps. This area placed in the north part of the central city in which village Vrontados is located.*

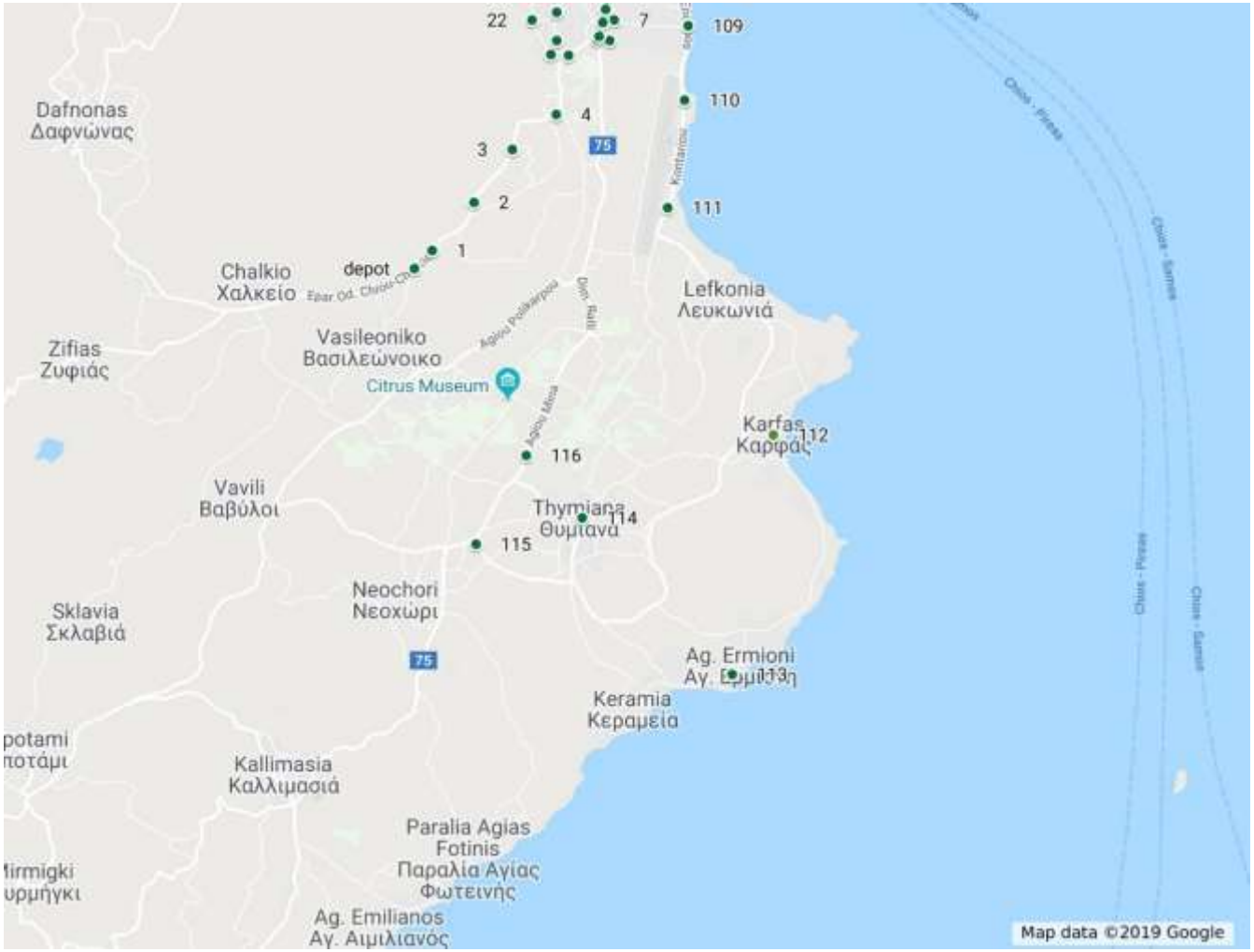


Figure L4. *This figure is the continuation of the previous map (Figure L1), while some geographical points are the same in both maps. This area placed in the south part of the central city in which villages Thimiana, Ag.Ermioni, Karfas are located..*

Appendix M: Input data for the case study

In this table is presented the number of bins and the frequency for each geographical point of the case study

Points	Number of bins in each point	Frequency of each point
depot	0	0
1	2	4
2	2	4
3	2	4
4	2	4
5	1	4
6	1	4
7	1	4
8	1	4
9	1	4
10	1	4
11	1	4
12	1	4
13	1	4
14	1	4
15	1	4
16	1	4
17	1	4
18	1	4
19	1	4
20	1	4
21	1	4
22	1	4

23	1	4
24	1	4
25	1	4
26	1	4
27	1	4
28	1	4
29	2	4
30	1	4
31	1	4
32	1	4
33	1	4
34	1	4
35	1	4
36	3	4
37	1	4
38	1	4
39	1	4
40	1	4
41	1	4
42	1	4
43	1	4
44	1	4
45	1	4
46	2	4
47	1	4
48	1	4
49	1	4

50	1	4
51	1	4
52	2	4
53	1	4
54	1	4
55	1	4
56	1	4
57	1	4
58	1	4
59	1	4
60	1	4
61	1	4
62	2	4
63	1	4
64	1	4
65	1	4
66	1	2
67	1	2
68	1	2
69	1	2
70	1	2
71	1	2
72	4	2
73	1	2
74	1	2
75	1	2
76	1	2

77	2	2
78	2	2
79	1	2
80	1	2
81	1	2
82	1	2
83	1	2
84	1	2
85	1	2
86	1	2
87	1	2
88	1	2
89	1	2
90	1	2
91	1	2
92	1	2
93	1	2
94	1	2
95	1	2
96	1	2
97	2	2
98	1	2
99	2	2
100	1	2
101	1	2
102	1	2
103	1	2

104	3	2
105	1	2
106	2	2
107	1	2
108	2	2
109	2	2
110	2	2
111	3	2
112	3	2
113	7	2
114	8	2
115	4	2
116	5	2

Appendix N: Algorithm and Pseudocode for HA1

The steps of HA1 to solve are the following:

Step 1. For each index of the distance matrix compute the savings by using the following formulation: $c_{0j} + c_{i0} - c_{ij}$, where 0 is the depot. Set the produced matrix as $(M\_savings)$. Sort the point pairs of $(M\_savings)$ in descending order and define it as $(C\_savings)$. Initialize variable $status\_scan = 1$. Each time we select a pair of nodes from $(C\_savings)$, variable $status\_scan$ will increased by one.

Step 2. Set the total number of days in the period as $period$ and initialize the current day as variable $day = 0$. Create a list of $period\ x\ n$ dimensions, where n is the total number of nodes and set it as $visit\_list$. When a node i inserted in a route of the current day the corresponded index in $visit\_list(day, i)$ will be equal to 1. Initially all indices of $visit\_list$ are 0.

Step 3. First, set the capacity of the current truck as $cap = 0$. The fleet consists of homogeneous capacity. Set cabin's capacity as $MAXcap$. Now define how many collection trucks the fleet has as $fleet = [MAXcap, \dots, MAXcap]$, where $MAXcap$ is the maximum capacity you have set. The shift of each collection truck defined by a limited number of nodes per day, so set that limited number as $DAYlimit$. Also let $TOTALcap = 0$ be the total number of bins that the current vehicle has selected in the current day. A truck can do more than one shift per day, for this reason initialize $veh_h = 1$, where veh shows vehicle's id and h the current shift. Now select the first vehicle of the fleet $fleet(veh_h)$.

Step 4. Create the matrix with the frequencies f. Variable matrix f consists of one row and n columns. The number of the column represents the id of the node, while the index of the column shows the frequency value. Each node can have more than one bins. Set the number of bins in a node as w_i, where w is the total number of bins in node i.

Step 5. Set the id of the route as rte_{day}^{veh}, where day shows the current day and veh shows the current vehicle. For instance, this path: $rte_{2}^{1} = [0 - 5 - 7 - 3 - 9 - 0]$, is implemented by the vehicle 1 during the day 2.

Step 6. If all days of the period have been passed and the problem has not been completed, go to the first day of the period.

If $day == 0$ **OR** $day > period$

Go to day 1:

$day = 1$

Select the next available vehicle:

$veh_h = veh_h + 1$

Step 7. $length(fleet)$ computes how many collection trucks the fleet has. If all the trucks have already been used and the problem has not been completed, select again the first vehicle of the fleet.

If $veh_h > length(fleet)$

Select the first truck of the fleet again:

$veh_h = 1$

Step 8. According to Clarke and Wright's steps the list of savings $(C\_savings)$ must be scanned only once. In case of multi day schedule of the PVRP, this procedure is repeated several times through the period. If $(C\_savings)$ has been totally scanned in the current day go to the next day of the period.

If $status\_scan > length(C\_savings)$

Go to the next day:

$day = day + 1$

Start scanning the $(C\_savings)$ from the beginning:

$status\_scan = 1$

Send the truck to the depot and empty the cabin:

$cap = 0$

Once the day has changed eliminate the shift:

$TOTALcap = 0$

Step 9. The basic concept of our model is to route multiple visits in the same nodes in a whole period. This idea explained by matrix f, which contains the information of how many times we must make those visits in each point. Each time a node is inserted in a route, variable f is updated appropriately by subtracting one visit in the appropriate column. Once the variable f has exhausted and all its values are equal to zero, the algorithm will be terminated.

If f is not exhausted

In matrix ($\boldsymbol{C\_savings}$) search for pair of nodes $\boldsymbol{C\_savings}(\boldsymbol{status\_scan})$ and save them $(\boldsymbol{i,j})$.

Else

End of HA1

Step 10. The method of the Clarke & Wright is implemented in each single day of the period to create feasible initial solutions. Step number 10 ensures the variable day will not exceed the length of the set period.

If $day \leq period$

The vehicle can select a limited number of bins per day, according to the definition of the shift.

If $TOTALcap \leq DAYlimit$

a. First, ensure that C&W steps are followed. **Make the appropriate insertions but do not update the changes permanently.**

$c1i = 0$ ***AND*** $c1j = 0$

If the path is empty insert both ($\boldsymbol{i, j}$) between the first and last point (depot)**.**

If $rte_{day}^{veh}(end-1) == 0\ (depot)$

Insert $i,j\ between\ the\ the\ first\ and\ last\ depot$ in rte_{day}^{veh}

$TOTALcap = \ TOTALcap + \ w_i$

$TOTALcap = \ TOTALcap + \ w_j$

$cap = \ cap + \ w_i$

$cap = \ cap + \ w_j$

$visit\_list(day, i) = 1$

$visit\_list(day, j) = 1$

$c1i = 1$

$c1j = 1$

Insert j in the end of the path before the depot.

Else if $rte_{day}^{veh}(end-1) == i$

Insert $j\ next\ to\ i\ in\ rte_{day}^{veh}$

$TOTALcap = \ TOTALcap + \ w_j$

$cap = \ cap + \ w_j$

$visit\_list(day, j) = 1$

$c1j = 1$

Insert i in the beginning of the path after the depot.

Else if $rte_{day}^{veh}(2) == j$

Insert $i\ next\ to\ j\ in\ rte_{day}^{veh}$

variable $TOTALcap = \ TOTALcap + \ w_i$

variable $cap = \ cap + \ w_i$

matrix $visit(day, 1) = 1$

$c1i = 1$

Else

$c1 = \ 0$

$status\_scan \ = status\_scan \ + 1$

Go to step 8

b. The inserted node must not be visited twice in the same day.

If i is going to be inserted in the current day check if that node has already been routed.

If $c1i \ == 1$ **AND** $visit\_list(day, i) == 1$

Check variable $c2 = \ 0$

$status\_scan \ = status\_scan \ + 1$

Go to step 8

If j is going to be inserted in the current day check if that node has already been routed.

Elseif $c1j == 1$ **AND** $visit\_list(day, j) == 1$

Check variable $c2 = 0$

$status\_scan = status\_scan + 1$

Go to step 8

Elseif $c1i == 1$ **AND** $c1j == 1$ **AND** $visit\_list(day, i) == 1 \ldots$

AND $visit\_list(day, j) == 1$

Check variable $c2 = 0$

$status\_scan = status\_scan + 1$

Go to step 8

Otherwise, note that the insertion of i or j can be feasibly done.

Else

$c2 = 1$

c. The scenario of consecutive days must be followed.

If $c1j == 1$ **AND** $visit\_list(day - 1, j) == 1$ **AND . . .** $period - f(j) > 0$

Check variable $c3 = 0$

$status\_scan = status\_scan + 1$

Go to step 8

If $c1i = = 1$ **AND** $visit\_list(day - 1, i) == 1$ **AND . . .** $period - f(i) > 0$

Check variable $c3 = 0$

$status\_scan = status\_scan + 1$

Go to step 8

If $c1i == 1$ **AND** $c1j == 1$ **AND** $visit\_list(day-1, i) == 1$ **. . . AND** $visit\_list(day-1, j) == 1$ **AND** $period - f(i) > 0$ **. . .** **AND** $period - f(j) > 0$

Check variable $c3 = 0$

$status\_scan = status\_scan + 1$

Go to step 8

Else

$c3 = 1$

d. We need to know if the content of the selected nodes can be loaded in the current vehicle.

If $cap \leq \text{MAX}cap$

$c4 = 1$

Else

$c4 = 0$

$status\_scan = status\_scan + 1$

Go to step 8

e. In case that all algorithm's parameters $(c1.\, c2.\, c3, c4)$ are complied in step **e**. the next appropriate computations can be proceeded.

If $c1 = 1$ && $c2 = 1$ && $c3 = 1$ $c4 = 1$

$status\_scan = status\_scan + 1$

Update the route rte_{day}^{veh}, the capacity of the vehicle cap, the list of the visits $visit\_list$, update variable f and finally update $TOTALcap$.

Go to step 8

Else

$status\_scan = status\_scan + 1$

Go to step 8

Else

If the shift completed go to the next day:

$day = day + 1$

Once the day has changed start scanning the ($\boldsymbol{C\_savings}$) from the beginning:

$status\_scan = 1$

Send the truck to the depot to unload the cabin:

$cap = 0$

Finally end the shift:

$TOTALcap = 0$

Go to step 6

Else

$cap = 0$

$TOTALcap = 0$

$day = 0$

$veh_h = 0$

Go to step 6

Appendix O: Algorithm and Pseudocode for HA2

The steps of HA1 to solve PVRP are the following:

Step 1. Create a matrix contains the frequencies and set it as f. Variable f consists of one row and n columns. Each column represents the id of the node. Each node can have more than one bins. Set the number of bins in a node as w_i, where w is the total number of bins in node i.

Step 2. In the first step of HA2 all nodes are clustered in different groups according to their frequency value. For example, if the problem contains some points that they need to be picked two times in the period and the rest of them three times, two group of points will be created. The first one includes those with frequency equal to two and the other one those with frequency equal to three. Let those groups be $cluster\_1_f$, where f is the frequency of the corresponding group.

Step 3. Step 3 sorts only the nodes from the first group in ascending order in accordance with their distance from the depot. Let this cluster be $cluster\_2_f$, where this f is the highest frequency value of the problem.

Step 4. In this step a procedure of similar logic will be made for the rest groups. In the current step the travel cost is not computed from the depot but from the latter point of the immediately preceding group. Consequently, the procedure of sorting group 2 bases on the last node of the 1st group, the 3rd group is based on the last point of 2nd group etc. Let these groups be $cluster\_3_f$.

Step 5. In step 4 a merge between all the groups is made. Each group one beneath the other, are connected in one giant list of $n\ X\ 1$ dimensions, where n are the nodes. Let this single list be $cluster\_list$, which will be scanned in the following steps by variable $status\_scan = 0$.

Step 6. Set the total number of days in the period as $period$ and initialize the current day as variable $day = 1$. Create a list of $period\ x\ n$ dimensions, where n is the total number of nodes and set it as $visit\_list$. When a node i inserted in the current day the corresponding index in $visit\_list(day, i)$ will be equal to 1. Initially all indices of $visit\_list$ are 0.

Step 7. First, set the capacity of the current truck as $cap = 0$. The fleet consists of homogeneous capacity. Set cabin's capacity as $MAXcap$. Now define how many collection trucks the fleet has as $fleet = [MAXcap, \dots, MAXcap]$, where $MAXcap$ is the maximum capacity you have set. The shift of each collection truck defined by a limited number of nodes per day, so set that limited number as $DAYlimit$. Also let $TOTALcap = 0$ be the total number of bins that the current vehicle has selected in the current day. A truck can do more than one shift per day, for this reason initialize $veh_h = 1$, where veh shows vehicle's id and h the current shift. Now select the first vehicle of the fleet $fleet(veh_h)$.

Step 8. In HA2 nodes are going to be inserted in days into specific classes creating a schedule of visits. After that the nodes of each class will be selected to produce a corresponding route. So let the classes be cls_{day}^{veh} and the corresponded route of that will be rte_{day}^{veh}, where day shows the current day and veh shows the current vehicle. For instance, let suppose a random class $cls_{day}^{veh} = [3, 5, 7, 9]$ and the path of that will be : $rte_{2}^{1} = [0 - 5 - 7 - 3 - 9 - 0]$, implemented by the vehicle 1 during the day 2.

Step 9. If all days of the period have been passed and the scheduling of visits has not been completed, go to the first day of the period.

If $day == 0$ **OR** $day > period$

Go to day 1:

$day = 1$

Select the next available vehicle:

$veh_h = veh_h + 1$

Step 10. $length(fleet)$ computes how many collection trucks the fleet has. If all the trucks have already been used and the problem has not been completed, select again the first vehicle of the fleet.

If $veh_h > length(fleet)$

Select the first truck of the fleet again:

$veh_h = 1$

Step 11. Variable $status\_scan$ is needed in order to select the nodes from the merged list one by one in each repetition.

If $status\_scan > length(cluster\_list)$

- Go to the next day:

  $day = day + 1$

- Start scanning the ($cluster\_list$) from the beginning:

  $status\_scan = 1$

- Reset the truck's cabin:

  $cap = 0$

- Once the day has changed eliminate the shift:

  $TOTALcap = 0$

Step 12. The basic concept of our model is to route multiple visits in the same nodes in a whole period. This idea explained by matrix f, which contains the information of how many times we must make those visits in each point. Each time a node is inserted in a route, variable f is updated appropriately by subtracting one visit in the appropriate column. Once the variable f has exhausted and all its values are equal to zero, the algorithm will be terminated.

If f is not exhausted

In matrix $\boldsymbol{cluster\_list}$ search for the node $\boldsymbol{cluster\_list}$ ($\boldsymbol{status\_scan}$) and save it as $\boldsymbol{i}$.

Else

Go to step 14

Step 13. If $day \leq period$

- The vehicle can select a limited number of bins per day, according to the definition of the shift.

  If $TOTALcap \leq DAYlimit$

  a. The inserted node must not be visited twice in the same day.

  If i is going to be inserted in the current day check if that node has already been routed.

  If $visit\_list(day, i) == 1$

  Check variable $c1 = 0$

  $status\_scan = status\_scan + 1$

  Go to step 11

  Else

  $c1 = 1$

  b. The scenario of consecutive days must be followed.

  If $visit\_list(day - 1, i) == 1$ **AND** $period - f(i) > 0$

  Check variable $c2 = 0$

  $status\_scan = status\_scan + 1$

Go to step 11

Else

$c2 = 1$

c. We need to know if the content of the selected nodes can be loaded in the current vehicle.

If $cap \leq \text{MAX}cap$

$c3 = 1$

Else

$c3 = 0$

$status\_scan = status\_scan + 1$

Go to step 11

d. In case that all algorithm's parameters $(c1.c2.c3)$ are complied the next appropriate computations can be proceeded.

If $c1 = 1$ **AND** $c2 = 1$ **AND** $c3 = 1$

$status\_scan = status\_scan + 1$

Update rte_{day}^{veh}, the capacity of the vehicle cap, the list of the visits $visit\_list$, update variable f and finally update $TOTALcap$.

Go to step 11

Else

- If the shift completed go to the next day:

$day = day + 1$

- Once the day has changed start scanning the $cluster\_list$ from the beginning:

  $status\_scan = 1$

- Reset vehicle's cabin:

  $cap = 0$

- Finally end the shift:

  $TOTALcap = 0$

  Go to step 9

Else

Set the day as 0.

$day = 0$

Go to step 9

Step 14. This step is the final one of the algorithm and what it does is to select individually all the unrouted classifications rte_{day}^{veh}, in order to make them into feasible paths. As happened in HA1 the initial solutions produced by Clarke & Wright's orders. For each index of the distance matrix compute the savings by using the following formulation: $c_{0j} + c_{i0} - c_{ij}$, where 0 is the depot. Set the produced matrix as $(M\_savings)$. Sort the point pairs of $(M\_savings)$ in descending order and define it as $(C\_savings)$.

a. **For** all days of the period and **for** all cls_{day}^{veh} create the corresponding routes rte_{day}^{veh}

b. In matrix $(\boldsymbol{C\_savings})$ search for next pair of nodes and save them as $(\boldsymbol{i,j})$

- **Make the following temporary changes. If those changes respect all the restrictions make them permanent**

- If the penultimate node of rte_{day}^{veh} is the depot, insert both $(\boldsymbol{i}, \boldsymbol{j})$ between the first and last point

  If $rte_{day}^{veh}(end-1) == 0\ (depot)$

  - Insert $i,j\ between\ the\ the\ first\ and\ last$ point in rte_{day}^{veh}

    $rte_{day}^{veh} = [0\ i\ j\ 0\ ]$

  - Update the capacity of the cabin, adding the number of nodes of i

    $cap = cap + w_i$

  - Update the capacity of the cabin, adding the number of nodes of j

    $cap = cap + w_j$

  - Confirm that node i has been inserted

    $c1i = 1$

  - Confirm that node j has been inserted

    $c1j = 1$

  Insert j in the end of the path before the depot.

  Else if $rte_{day}^{veh}(end-1) == i$

  - Insert $j\ next\ to\ i\ in\ rte_{day}^{veh}$

    $rte_{day}^{veh} = [0, i-k, \dots,\ i-1,\ \boldsymbol{j}\ 0\ ]$

  - Update the capacity of the cabin adding the number of nodes of j

    $cap = cap + w_j$

  - Confirm that node j has been inserted

    $c1j = 1$

Insert i in the beginning of the path after the depot.

Else if $rte_{day}^{veh}(2) == j$

Insert $i\ next\ to\ j\ in\ rte_{day}^{veh}$

$rte_{day}^{veh} = [0, \boldsymbol{i}, j, \dots, j+k, 0\ ]$

- Update the capacity of the cabin, adding the number of nodes of i

  $cap = cap + w_i$

- Confirm that node i has been inserted

  $c1i = 1$

If $(C\_savings)$ is totally scanned

go to a

Else

go to b