ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΜΟΝΤΕΛΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΓΙΑ ΧΕΡΣΑΙΕΣ ΜΕΤΑΦΟΡΕΣ ΚΑΙ ΕΦΟΔΙΑΣΤΙΚΕΣ ΑΛΥΣΙΔΕΣ ΜΕ ΤΟ ΛΟΓΙΣΜΙΚΟ ΕΦΑΡΜΟΓΩΝ GUROBI PYTHON INTERFACE ΣΕ JUPYTER NOTEBOOKS

υπό
Αθανασίου Παντόπουλου

Επιβλέπων καθηγητής : Δρ. Αθανάσιος Ζηλιασκόπουλος

Διπλωματική Εργασία

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για
την απόκτηση του Διπλώματος Μηχανολόγου Μηχανικού

Βόλος, 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΜΟΝΤΕΛΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΓΙΑ ΧΕΡΣΑΙΕΣ ΜΕΤΑΦΟΡΕΣ ΚΑΙ ΕΦΟΔΙΑΣΤΙΚΕΣ ΑΛΥΣΙΔΕΣ ΜΕ ΤΟ ΛΟΓΙΣΜΙΚΟ ΕΦΑΡΜΟΓΩΝ GUROBI PYTHON INTERFACE ΣΕ JUPYTER NOTEBOOKS

υπό
Αθανασίου Παντόπουλου

Επιβλέπων καθηγητής : Δρ. Αθανάσιος Ζηλιασκόπουλος

Διπλωματική Εργασία

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για
την απόκτηση του Διπλώματος Μηχανολόγου Μηχανικού

Βόλος, 2021

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Αθανάσιος Ζηλιασκόπουλος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Ευαγγελία Χρυσοχόου Καθηγήτρια, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Γιώργος Σαχαρίδης Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής εργασίας μου, Καθηγητή κ. Αθανάσιο Ζηλιασκόπουλο, καθώς και τα υπόλοιπα μέλη της εξεταστικής επιτροπής, Καθηγητές κα. Ευαγγελία Χρυσοχόου και κ. Γιώργο Σαχαρίδη, για την προσεκτική ανάγνωση της εργασίας μου και για τις πολύτιμες υποδείξεις τους. Ιδιαίτερα θα ήθελα να ευχαριστήσω την κα. Ευαγγελία Χρυσοχόου για σημαντική βοήθεια της κατά τη διάρκεια της υλοποίησης της εργασίας μου. Επίσης, είμαι ευγνώμων σε όλους τους καθηγητές μου για τις πολύτιμες γνώσεις και αξίες που μου προσέφεραν κατά την διάρκεια της φοίτησης μου.

Ακόμα ευχαριστώ τους φίλους μου Βαγγέλη, Θανάση και Παναγιώτη και για την υποστήριξή τους και τις όμορφες εμπειρίες που είχαμε καθ' όλη την διάρκεια της φοίτησής μας στον Βόλο. Ένα μεγάλο ευχαριστώ οφείλω και στην κοπέλα μου Μαρία για την στήριξη και κατανόησή της, ιδιαίτερα κατά τη διάρκεια των τελευταίων μηνών της προσπάθειάς μου.

Τέλος, είμαι ευγνώμων στους γονείς μου, Κώστα και Ελένη για την αγάπη, υπομονή και υποστήριξή τους όλα αυτά τα χρόνια, και στους οποίους και αφιερώνω αυτή την εργασία.

ΜΟΝΤΕΛΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΓΙΑ ΧΕΡΣΑΙΕΣ ΜΕΤΑΦΟΡΕΣ ΚΑΙ ΕΦΟΔΙΑΣΤΙΚΕΣ ΑΛΥΣΙΔΕΣ ΜΕ ΤΟ ΛΟΓΙΣΜΙΚΟ ΕΦΑΡΜΟΓΩΝ GUROBI PYTHON INTERFACE ΣΕ JUPYTER NOTEBOOKS

Αθανάσιος Παντόπουλος

Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας, 2021

Επιβλέπων Καθηγητής: Δρ. Αθανάσιος Ζηλιασκόπουλος,

Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Περίληψη

Η έρευνα πάνω στην επίλυση προβλημάτων και στη βελτιστοποίηση της εφοδιαστικής αλυσίδας είναι εκτενής. Την τελευταία δεκαετία η κυριαρχία της παγκόσμιας εφοδιαστικής αλυσίδας, η άνοδος του ηλεκτρονικού εμπορίου και η ανάπτυξη της τεχνολογίας συντέλεσε στη δημιουργία νέων προκλήσεων αλλά και νέων εργαλείων σχεδιασμού και οργάνωσης της μοντέρνας εφοδιαστικής αλυσίδας. Ιδιαίτερο ενδιαφέρον παρουσιάζει η ανάδειξη των λογισμικών μαθηματικής βελτιστοποίησης (solvers), των οποίων η χρήση τους σε πρακτικές εφαρμογές συνεχώς αυξάνεται.

Στόχος της παρούσας διπλωματικής είναι ο εντοπισμός και η μοντελοποίηση των βασικότερων προβλημάτων της εφοδιαστικής αλυσίδας καθώς και η εφαρμογή τους σε προσομοιωμένα προβλήματα με τη χρήση της διεπαφής Python του λογισμικού βελτιστοποίησης της GUROBI σε περιβάλλον Jupyter Notebooks. Επιπλέον στόχος είναι η εισαγωγή του αναγνώστη στη χρήση του λογισμικού καθώς και εξαγωγή συμπερασμάτων από την συμπεριφορά των μοντέλων για διάφορα μεγέθη και τύπους προβλημάτων.

Τα προβλήματα της εφοδιαστικής αλυσίδας που εξετάζονται εντός της εργασίας είναι το πρόβλημα δρομολόγησης στόλου οχημάτων (VRP), το πρόβλημα συλλογής παράδοσης (PDP) και το πρόβλημα δρομολόγησης αποθέματος (IRP). Για τα VRP και PDP εξετάζονται δύο εναλλακτικές μοντελοποιήσεις, η μία εκ των οποίων αξιοποιεί την δυνατότητα δυναμικής προσθήκης περιορισμών του λογισμικού της Gurobi.

Τα αποτελέσματα που προκύπτουν τόσο από την κατασκευή των μοντέλων βελτιστοποίησης όσο και από την εφαρμογή τους σε πλήθος προβλημάτων, φανερώνουν την

ευκολία χρήσης και ισχύ του λογισμικού της GUROBI, αναδεικνύοντας την ευελιξία και χρησιμότητα του.

Λέξεις-κλειδιά: Εφοδιαστική αλυσίδα, Βελτιστοποίηση, Πρόβλημα δρομολόγησης οχημάτων, Πρόβλημα συλλογής παράδοσης, Πρόβλημα δρομολόγησης αποθέματος, Gurobi, Python, Jupyter Notebooks

ANALYTIC AND NUMERICAL METHODS DEVELOPMENT FOR ASSESSING THE PERFORMANCE OF SYSTEMS XXX

Athanasios Pantopoulos

Department of Mechanical Engineering, University of Thessaly, 2021

Supervisor: Dr Athanasios Ziliaskopoulos

Professor, Department of Mechanical Engineering, University of Thessaly

Abstract

Research related to problem solving and supply chain optimization is extensive. The last decade the domination of global supply chain, the rise of e-commerce and the technological developments resulted in new challenges, yet also new tools for planning and organizing the modern supply chain. Of particular interest is the prominence of mathematical optimization software, whose use in real world applications is constantly increasing.

The objective of this Thesis is to identify and model the basic problems encountered in the supply chain and their application in simulated problems using the Python interface of the mathematical optimization software GUROBI in Jupyter Notebooks environment. Additional goals are the introduction of the reader to the use of the software and to draw some conclusions form how the models behave in problems of various sizes and types.

The supply chain problems that are contained in this Thesis are the vehicle routing problem (VRP), the pick-up and delivery problem (PDP) and the inventory routing problem (IRP). In both VRP and PDP, two mathematical models are examined, one of which utilizes the dynamic addition of constraints, which the GUROBI software allows.

Both the results of constructing optimization models and applying them in a variety of problems, show the simplicity and the power of GUROBI software, pointing out its usefulness and flexibility.

Key words: Supply chain, Optimization, Vehicle routing problem, Pick-Up and delivery problem, Inventory Routing Problem, Gurobi, Python, Jupyter Notebooks

Πίνακας Περιεχομένων

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Κεφάλαιο 1. Εισαγωγή

1.1 Κίνητρο και στόχοι

Η μαθηματική βελτιστοποίηση, γνωστή και ως μαθηματικός προγραμματισμός, αποτελεί ισχυρό αναλυτικό εργαλείο για τις επιχειρήσεις, επιτρέποντας την επίλυση περίπλοκων προβλημάτων και αποτελεσματικότερη αξιοποίηση πόρων και δεδομένων. Όλο και περισσότερες επιχειρήσεις αξιοποιούν τεχνολογίες μαθηματικής βελτιστοποίησης για την ελαχιστοποίηση των δαπανών και αύξηση της κερδοφορίας τους, σε διάφορους τομείς όπως η διανομή ενέργειας, το χρηματιστήριο, η παραγωγή προϊόντων, η οργάνωση αθλητικών αγώνων και η εφοδιαστική αλυσίδα, στην οποία επικεντρώνεται η τρέχουσα διπλωματική.

Η σύγχρονη εφοδιαστική αλυσίδα χαρακτηρίζεται από τον υπερανταγωνισμό και περιπλοκότητα της, με τις σχετιζόμενες επιχειρήσεις να αναζητούν συνεχώς την καλύτερη τεχνολογία, η οποία θα τους εξασφαλίσει το ανταγωνιστικό πλεονέκτημα βελτιώνοντας την ταχύτητα, απόδοση και ποιότητα των προϊόντων και υπηρεσιών που προσφέρουν στους πελάτες τους μέσω της εφοδιαστικής τους αλυσίδας.

Πέρα από την μαθηματική βελτιστοποίηση, οι εταιρείες για την διαχείριση της εφοδιαστικής τους αλυσίδας κάνουν χρήση και άλλων τεχνολογιών, κυρίως ευρετικών αλγόριθμων (metaheuristics, local search, machine learning algorithms ,evolutionary algorithm). Αν και η πλειοψηφία των επιχειρήσεων σήμερα βασίζεται κυρίως σε ευρετικούς αλγορίθμους λόγω της ταχύτητας και απλότητας εφαρμογής τους, η εξέλιξη της ταχύτητας, αξιοπιστίας και ευελιξίας των λογισμικών βελτιστοποίησης (solvers) καθιστά πλέον αρκετά ελκυστική την μαθηματική βελτιστοποίηση της εφοδιαστικής αλυσίδας.

Αντικείμενο της παρούσας εργασίας είναι η μοντελοποίηση και επίλυση βασικών προβλημάτων της εφοδιαστικής αλυσίδας με την Python διεπαφή του λογισμικού μαθηματικής βελτιστοποίησης GUROBI. Πέρα από την παρουσίαση των προβλημάτων και

των αποτελεσμάτων της επίλυσης τους, επιπλέον στόχος είναι η εισαγωγή και εξοικείωση του αναγνώστη με το λογισμικό της GUROBI, το οποίο λόγω της πρόσφατης σχετικά ανάπτυξης του είναι ακόμα άγνωστο, τουλάχιστον εκτός του ακαδημαϊκού χώρου, στην Ευρώπη και στην Ελλάδα.

1.2 Οργάνωση διπλωματικής εργασίας

Στο κεφάλαιο 2 παρουσιάζονται κάποια βασικά προβλήματα της εφοδιαστικής αλυσίδας, τα οποία είναι το πρόβλημα δρομολόγησης οχημάτων (VRP), το πρόβλημα συλλογής παράδοσης (PDP) και το πρόβλημα δρομολόγησης αποθέματος (IRP). Το κάθε πρόβλημα περιγράφεται αναλυτικά και στην συνέχεια εξετάζονται οι πιο σημαντικές παραλλαγές του, ενώ τελικώς πραγματοποιείται η μοντελοποίηση του σε ένα ή περισσότερα εναλλακτικά μοντέλα γραμμικού ακέραιου προγραμματισμού.

Στο κεφάλαιο 3 γίνεται αρχικά μία σύντομη αναφορά στην ιστορία των λογισμικών μαθηματικής βελτιστοποίησης (solvers), ενώ στη συνέχεια ακολουθεί παρουσίαση των πλεονεκτημάτων και δυνατοτήτων του solver της GUROBI και λίγα λόγια για τη γλώσσα προγραμματισμού Python και το περιβάλλον Jupyter Notebooks. Κατά την έκταση του υπολοίπου κεφαλαίου γίνεται εισαγωγή στη μοντελοποίηση και επίλυση προβλημάτων βελτιστοποίησης με χρήση της διεπαφής Python της GUROBI.

Στο κεφάλαιο 4 παρουσιάζονται τα τμήματα κώδικα που σχετίζονται με την κατασκευή των μοντέλων που παρουσιάστηκαν στο κεφάλαιο 2. Για κάθε πρόβλημα περιγράφεται η διαδικασία δημιουργίας των τυχαίων προβλημάτων πάνω στα οποία θα τρέξουν οι κώδικες, η μέθοδος υπολογισμού αρχικών εφικτών λύσεων και κάποια αποτελέσματα επίδοσης των αλγορίθμων για διάφορα μεγέθη και παραλλαγές των προβλημάτων. Τελικώς ακολουθούν τα συμπεράσματα που προκύπτουν από τα αποτελέσματα των επιδόσεων κάθε μοντέλου.

Στο κεφάλαιο 5 γίνεται μία σύνοψη των αποτελεσμάτων και συμπερασμάτων της διπλωματικής καθώς και προτάσεις για μελλοντική έρευνα.

Στο παράρτημα Α της διπλωματικής παρεντίθενται οι ολοκληρωμένοι κώδικες κάθε μοντέλου, οι οποίοι πέρα από την μοντελοποίηση και επίλυση τον προβλημάτων περιέχουν την δημιουργία τυχαίου προβλήματος, τον υπολογισμού αρχικής εφικτής λύσης καθώς και μία προτεινόμενη οπτικοποίηση της λύσης. Στο παράρτημα Β περιέχονται τα δεδομένα όλων

των προβλημάτων καθώς και οι βέλτιστες ή καλύτερες (στην περίπτωση που δεν έχει βρεθεί η βέλτιστη) λύσεις τους.

Κεφάλαιο 2. Προβλήματα βελτιστοποίησης στην εφοδιαστική αλυσίδα

2.1 Εφοδιαστική Αλυσίδα

Η χρήση του όρου εφοδιαστική αλυσίδα στην σύγχρονη ιστορική και επιστημονική αναφορά εντοπίζεται αρχικά κατά τον Β' Παγκόσμιο Πόλεμο από τις ΗΠΑ και τους Συμμάχους για τον εφοδιασμό των νηοπομπών των συμμαχικών δυνάμεων. Το ενδιαφέρον σε επιστημονικό και επιχειρηματικό επίπεδο γύρω από την εφοδιαστική αλυσίδα αρχίζει να στρέφεται συστηματικά από τη δεκαετία του 1960, με σκοπό την ενιαία διαχείριση των επιμέρους λειτουργιών του τομέα της διανομής σε επίπεδο επιχείρησης [1].

Ο ακριβής ορισμός του όρου εφοδιαστική αλυσίδα είναι δύσκολος, τόσο λόγω του μεγάλου εύρους δραστηριοτήτων που καλύπτει όσο και του συνεχή επαναπροσδιορισμού του, λόγω της εξελισσόμενης τεχνολογίας και της αυξανόμενης παγκοσμιοποίησης. Συνοπτικά, η εφοδιαστική αλυσίδα αποτελεί τις εγκαταστάσεις, λειτουργίες και δραστηριότητες που περιέχονται στην παραγωγή και την παράδοση ενός προϊόντος ή μίας υπηρεσίας από τους προμηθευτές (και τους προμηθευτές τους) στους πελάτες (και τους πελάτες τους).

Οι βασικές λειτουργίες ή διαδικασίες που λαμβάνουν χώρα καθημερινά εντός της εφοδιαστικής αλυσίδας και συνθέτουν το κύκλωμα εφοδιασμού σε μία επιχείρηση είναι οι ακόλουθες :

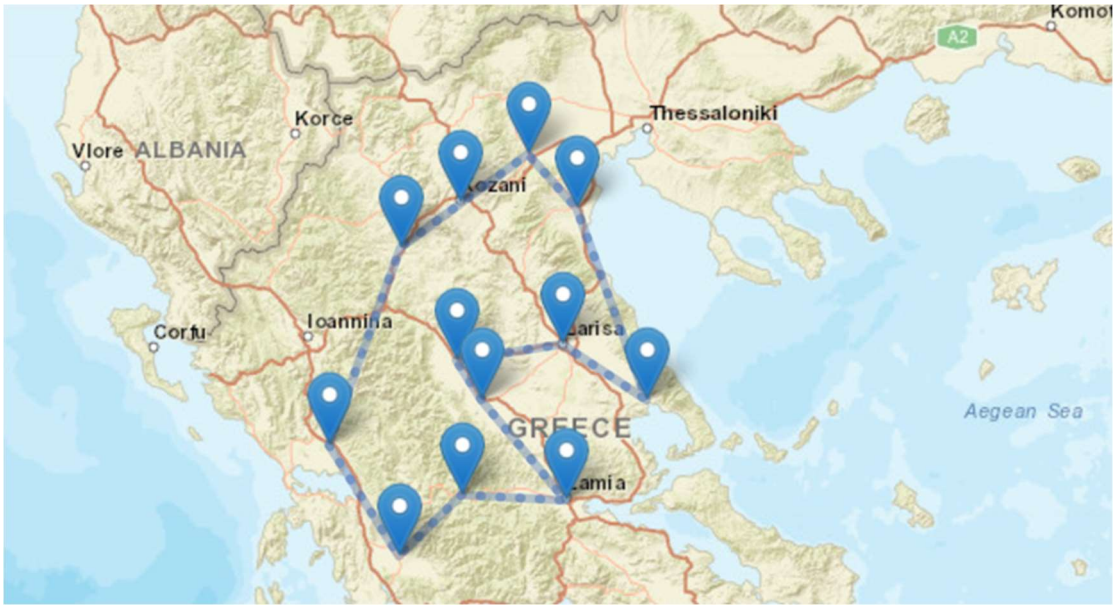
- Αγορές-Προμήθειες : Η εργασία απόκτησης προϊόντων ή υπηρεσιών από τρίτους, από προμηθευτές και από πηγές εκτός της επιχείρησης.
- Διαχείριση αποθέματος : Η εργασία υπολογισμού του άριστου επιπέδου των προϊόντων που θα πρέπει να διατηρεί μια επιχείρηση για να διεκπεραιώνει με επιτυχία τις εργασίες της.
- Διακίνηση (μεταφορές / διανομές) : Η εργασία εξεύρεσης του άριστου τρόπου της φυσικής μετακίνησης των προϊόντων που η επιχείρηση πρέπει να παραλάβει από τους προμηθευτές της ή να παραδώσει στους πελάτες της.
- Αποθήκευση : Ο σχεδιασμός, οργάνωση και λειτουργία της αποθήκης.

Για την αποτελεσματική διαχείριση της εφοδιαστικής αλυσίδας, σε κάθε μία από τις βασικές λειτουργίες της, απαιτείται εκτενής μελέτη και στρατηγικός σχεδιασμός, ώστε να εξασφαλιστεί η αποδοτική λειτουργία της καθώς και η ανταγωνιστικότητα της επιχείρησης. Στα πλαίσια της τρέχουσας διπλωματικής ο εντοπισμός των προβλημάτων βελτιστοποίησης θα αφορά σχεδόν εξ ’ολοκλήρου τις λειτουργίες διακίνησης, δηλαδή μεταφοράς και διανομής προϊόντων. Αξίζει να σημειωθεί πως το κόστος διακίνησης αποτελεί ένα από τα πιο αξιόλογα στοιχεία του συνολικού κόστους.

2.2 Το πρόβλημα του πλανόδιου πωλητή (TSP)

2.2.1 Περιγραφή και ιστορία του TSP

Το παλιότερο, ίσως, πρόβλημα διακίνησης στην εφοδιαστική αλυσίδα είναι το πρόβλημα του πλανόδιου πωλητή ή TSP (Travelling Salesman Problem), που αφορά την εύρεση της μικρότερης σε μήκος κλειστής διαδρομής σε ένα δίκτυο n σημείων, με κάθε σημείο να είναι επισκέψιμο ακριβώς μία φορά πριν την επιστροφή στο αρχικό σημείο.



Σχήμα 2.1: Βέλτιστη λύση του προβλήματος TSP για πόλεις της κεντρικής και βόρειας Ελλάδας. Στην συγκεκριμένη περίπτωση ως αρχικό σημείο έχει θεωρηθεί η Λάρισα.

Αν και η ακριβής προέλευση του TSP παραμένει άγνωστη, η πρώτη μαθηματική μορφοποίηση του εντοπίζεται στις αρχές του 19ου αιώνα από τους μαθηματικούς W.R. Hamilton και Thomas Kirkman. Η πρώτη αναφορά του προβλήματος με την σημερινή

ονομασία του γίνεται το 1949 σε μία αναφορά της RAND Corporation από την μαθηματικό Julian Robinson. Η μοντελοποίηση του TSP, ως πρόβλημα γραμμικού ακέραιου προγραμματισμού μπορεί να αποδοθεί στους μαθηματικούς George Dantzig, Delbert Ray Fulkerson και Selmer M. Johnson, κατά την εργασία τους στην RAND Corporation. Το 1972 η ο θεωρητικός υπολογιστών Richard M. Karp προσδιορίζει το πρόβλημα ως NP δυσχερές, δηλαδή πως δεν επιδέχεται λύση σε πολυωνυμικό χρόνο. Το 1976 οι Christofides και Serdyukov αναπτύσσουν ανεξάρτητα έναν ευρετικό αλγόριθμο, ο οποίος μέχρι και σήμερα επιλύει το TSP με την μικρότερη πιθανή απόκλιση από την βέλτιστη λύση (περίπου 1.5 φορές).

2.2.2 Μαθηματική σημειογραφία

Η συγκεκριμένη σημειογραφία έχει παρθεί από το βιβλίο των Toth και Vigo [2]. Έστω ένα υποθετικό σύνολο V={0,1,2,…,n} και ένα σύνολο A που προκύπτει από το V και ορίζεται ως $A = \{(i,j) \in V^2 : i \neq j\}$. Εάν S ένα υποσύνολο του A τότε ορίζω :

- $\delta^-(S) = \{\{(i,j) \in A : i \notin S, j \in S\}\}$
- $\delta^+(S) = \{\{(i,j) \in A : i \in S, j \notin S\}\}$
- $\delta^-(\{0\}) = \delta^-(0) \ \kappa\alpha\iota \ \delta^+(\{0\}) = \delta^+(0)$

Εάν το *S* είναι της μορφής $\{\dots, (i_k, j_m), (i_m, j_n), (i_n, j_b), \dots\}$, τότε ουσιαστικά εκφράζει μία αλληλουχία ή διαδρομή από ζεύγη $(i,j) \in A$. Σε αυτή την περίπτωση το $\delta^-(S)$ εκφράζει το σύνολο των ζευγών που εισέρχονται στην διαδρομή *S*, ενώ αντίστοιχα το $\delta^+(S)$ εκφράζει το σύνολο των ζευγών που εξέρχονται από την διαδρομή *S*.

2.2.3 Μοντελοποίηση του TSP

Στην υποενότητα 2.2.1 έγινε μία σύντομη περιγραφή του TSP. Ο αναλυτικός ορισμός του προβλήματος σύμφωνα με τους Miller, Tucker και Zemlin [3] είναι ο εξής: Ένας πωλητής πρέπει να επισκεφθεί κάθε μία πόλη από ένα σύνολο n πόλεων. Ξεκινώντας από μία πόλη αφετηρία, ο πωλητής επιτρέπεται να περάσει από κάθε πόλη μόνο μία φορά, ενώ πρέπει να επιστρέψει στην πόλη αφετηρία t φορές, πραγματοποιώντας έτσι t διαδρομές. Επιπλέον κατά μήκος μίας διαδρομής επιτρέπεται να επισκεφθεί έναν μέγιστο αριθμό πόλεων p, τέτοιο

ώστε tp≥n. Η πιο συνηθισμένη και γνωστή μορφή του προβλήματος είναι αυτή όπου t=1 και p=n. Σκοπός του προβλήματος είναι η εύρεση τής βέλτιστης αλληλουχίας επίσκεψης των πόλεων ώστε ο πωλητή να διανύσει την μικρότερη απόσταση.

Αρχικά παρουσιάζεται η μοντελοποίηση TSP των Miller, Tucker και Zemlin ως πρόβλημα γραμμικού ακέραιου προγραμματισμού [3]. Η συγκεκριμένη μοντελοποίηση αφορά το ασύμμετρο TSP, στο οποίο το κόστος μετάβασης μεταξύ δύο σημείων δεν είναι απαραίτητα το ίδιο και στις δύο κατευθύνσεις. Τα δεδομένα του προβλήματος είναι :

n → ο αριθμός των πόλεων προς επίσκεψη

N → το σύνολο των πόλεων, όπου $\mathrm{N} = \{1,2,\ldots,\mathrm{n}\}$

O → η πόλη αφετηρία, όπου $\mathrm{O} = \{0\}$

V → το σύνολο των κόμβων, όπου $\mathrm{V} = \mathrm{O} \cup \mathrm{N}$

A →σύνολο τόξων μεταξύ κόμβων, όπου $\mathrm{A} = \{(\mathrm{i},\mathrm{j}) \in \mathrm{V}^2 : \mathrm{i} \neq \mathrm{j}\}$

$\mathbf{d_{ij}}$→το μήκος του τόξου $(\mathrm{i},\mathrm{j}) \in \mathrm{A}$

t →αριθμός αναγκαστικών επιστροφών στην πόλη αφετηρίας

p →μέγιστός αριθμός επισκεπτόμενων πόλεων ανά διαδρομή

Οι μεταβλητές απόφασης και οι βοηθητικές μεταβλητές του μοντέλου ορίζονται ως :

$\boldsymbol{x_{ij}}$→δυαδική μεταβλητή απόφασης που παίρνει την τιμή 1 όταν το τόξο (i,j) χρησιμοποιείται, $(i,j) \in A$

$\boldsymbol{u_i}$→μη αρνητική ακέραια βοηθητική μεταβλητή, $i = (1,2,\ldots,n)$

Το μοντέλο βελτιστοποίησης του προβλήματος θα είναι :

$$minimize \sum_{(i,j \in A)} d_{ij} x_{ij} \quad (1)$$

Υπό τους περιορισμούς:

$$\sum_{j \in \delta^+(i)} x_{ij} = 1 \ , \forall\, i \in N \quad (2)$$

$$\sum_{i \in \delta^-(j)} x_{ij} = 1 \ , \forall \ j \in N \quad (3)$$

$$\sum_{i \in \delta^-(0)} x_{i0} = t \quad (4)$$

$$u_i - u_j + p x_{ij} \leq p - 1 \ , \forall \ (i,j) \in A(N) \quad (5)$$

$$x_{i,j} \ binary \ \ και \ u_i > 0 \ integer \quad (6)$$

Η αντικειμενική συνάρτηση (1) ορίζεται ως η εύρεση της ελάχιστης απόστασης. Οι περιορισμοί (2) και (3) εξασφαλίζουν πως σε κάθε πόλη αντιστοιχεί μόνο ένα τόξο εξόδου και εισόδου αντίστοιχα. Ο περιορισμός (4) φροντίζει πως ο πωλητής θα επιστρέψει στην πόλη «αφετηρία» t φορές. Στην περίπτωση όπου t=1 ο περιορισμός (4) μπορεί να παραληφθεί. Ο περιορισμός (5) αποτρέπει την δημιουργία κλειστών διαδρομών που δεν περιλαμβάνουν το {0} και εξασφαλίζει πως σε κάθε διαδρομή θα γίνει επίσκεψη το πολύ p πόλεων. Στην περίπτωση όπου t=1, στον περιορισμό 5, το p μπορεί να αντικατασταθεί με n.

Για την απόδειξη της λειτουργίας του περιορισμού (5) έστω μία κλειστή διαδρομή S_m η οποία αποτελείται από m σημεία και δεν διέρχεται από το 0. Ξεκινώντας από ένα τυχαίο σημείο της διαδρομής, έστω m_1, δημιουργώντας τις ανισότητες του περιορισμού (5) για κάθε σημείο και αθροίζοντας τες κατά μέλη προκύπτει:

$$p \sum_{(i,j) \in S_m} x_{ij} \leq m(p-1) \rightarrow pm \leq m(p-1) \rightarrow p \leq (p-1) \quad \acute{\alpha}τοπο$$

Ακόμα έστω μία κλειστή διαδρομή της μορφής $(0,1,2,\dots,m,0)$ με $p < m$. Εφαρμόζοντας τον περιορισμό (5), αρχίζοντας από το σημείο 1, σε κάθε σημείο της διαδρομής δημιουργούνται m-1 ανισότητες (τα τόξα που περιέχουν το 0 δεν παράγουν περιορισμούς) και προσθέτοντας τες κατά μέλη προκύπτει:

$$u_1 - u_m + p(m-1) \leq (p-1)(m-1) \rightarrow u_1 - u_m \leq 1 - m < 0 \rightarrow u_1 < u_m$$

Παρομοίως εάν ξεκινώντας από το σημείο m και καταλήγοντας στο 1 τότε προκύπτει αντίφαση:

$$u_m - u_1 + p(m-1) \leq (p-1)(m-1) \rightarrow u_m < u_1$$

Όπως είναι φανερό με την αύξηση των πόλεων ο αριθμός των περιορισμών (5) αυξάνεται τετραγωνικά, μειώνοντας δραματικά την απόδοση του μοντέλου για μεγάλα n. Μία λύση σε αυτό το πρόβλημα, εάν t=1, είναι το μοντέλο του Dantzig [4] το οποίο έχει την ίδια αντικειμενική (1) και περιορισμούς (2) και (3). Η διαφορά είναι πως αντί του περιορισμού (5) γίνεται η χρήση του περιορισμού:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \ , S \subseteq N \quad (6)$$

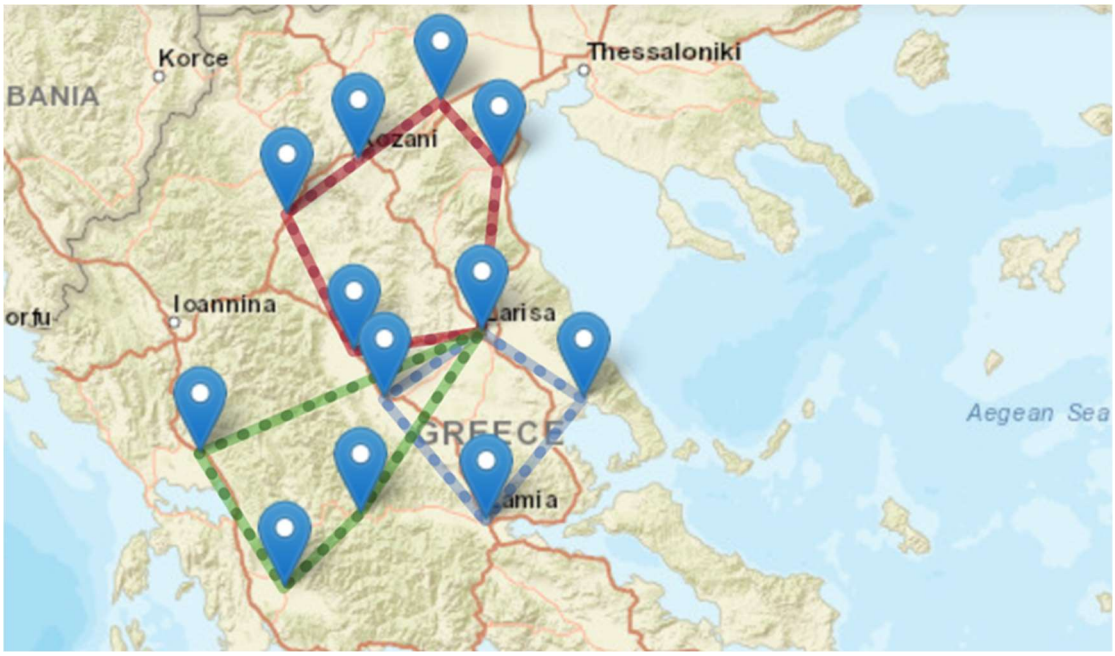
Πλέον ο περιορισμός (6) είναι υπεύθυνος για την εξάλειψη των κλειστών διαδρομών που δεν περιλαμβάνουν το {0}. Εάν μία τέτοια διαδρομή υποθετικά υπάρχει θα περιέχει $|S|$ τόξα (i, j), με αποτέλεσμα το δεξί μέλος του περιορισμού να ισούται με $|S|$ καταπατώντας έτσι τον περιορισμό.

Η επιλογή των S γίνεται αλγοριθμικά. Αρχικά γίνεται επίλυση του TSP μόνο με τους περιορισμούς (2) και (3). Οι διαδρομές που προκύπτουν από την λύση του προβλήματος ελέγχονται εάν τηρούν τον περιορισμό (6) και για κάθε διαδρομή που τον παραβιάζει γίνεται προσθήκη του αντίστοιχου περιορισμού στο αρχικό μοντέλο. Η διαδικασία της επίλυσης και πρόσθεσης νέων περιορισμών επαναλαμβάνεται και τερματίζει όταν στην λύση του μοντέλου καμία διαδρομή δεν παραβιάζει τον περιορισμό 6. Η λύση αυτή είναι η βέλτιστη λύση του TSP. Περιορισμοί τέτοιου τύπου, όπως θα γίνει εμφανές, θα αξιοποιηθούν σε κάθε πρόβλημα της τρέχουσας διπλωματικής και θα αποκαλούνται *περιορισμοί callback*.

2.3 Το πρόβλημα δρομολόγησης στόλου οχημάτων (VRP)

2.3.1 Περιγραφή και ιστορία του VRP

Το πρόβλημα δρομολόγησης στόλου οχημάτων ή VRP (Vehicle Routing Problem) είναι το κεντρικό και πιο χαρακτηριστικό πρόβλημα διακίνησης της εφοδιαστικής αλυσίδας. Το VRP αφορά την εύρεση των βέλτιστων δρομολογίων παράδοσης ή συλλογής από ένα ή περισσότερα κέντρα διανομής σε ένα πλήθος πόλεων ή πελατών υπό διάφορους περιορισμούς. Μελετώντας πλήθος εφαρμογών οι Toth και Vigo [5] κατέληξαν στο συμπέρασμα πως τα πλάνα διανομής παραγόμενα από την επίλυση του VRP μπορούν να μειώσουν το κόστους διανομής από 5% έως 20%.



Σχήμα 2.2: Βέλτιστη λύση του προβλήματος VRP (CVRP) για πόλεις της κεντρικής και βόρειας Ελλάδας. Στην συγκεκριμένη περίπτωση το κέντρο διανομής είναι η Λάρισα.

Το πρόβλημα παρουσιάζεται για πρώτη φορά, ως μία γενίκευση του TSP, από τους Dantzig και Ramser το 1959 [6], οι οποίοι πραγματοποιούν την μοντελοποίηση του και πρότειναν μία αλγοριθμική προσέγγιση επίλυσης του. Τα επόμενα χρόνια ακολουθεί πλήθος μελετών με στόχο με τον προσδιορισμό και μοντελοποίηση κατηγοριών του VRP. Ακριβείς μέθοδοι επίλυσης του συμμετρικού και ασύμμετρούς VRP παρουσιάζονται από τους Laporte και Nobert (1987)[7]. Ακόμα πολλές μελέτες γίνονται γύρω από ευρετικές μεθόδους επίλυσης του VRP από πλήθος ερευνητών. Με την αύξηση της υπολογιστικής απόδοσης και την εξέλιξη

τεχνολογιών γεωεντοπισμού και συλλογής - ανάλυσης δεδομένων, ευνοήθηκε η ανάπτυξη επαγγελματικών λογισμικών δρομολόγησης με έμφυτες αλγοριθμικές προσεγγίσεις επίλυσης διάφορων τύπων VRP.

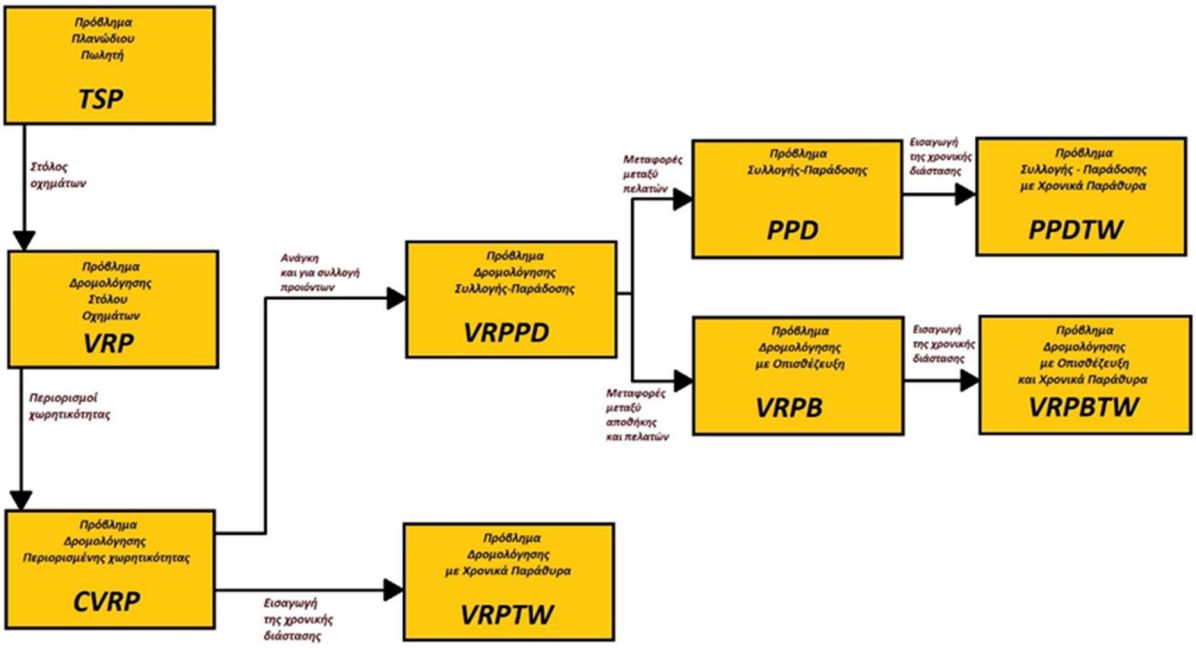
2.3.2 Παραλλαγές του VRP

Το μεγάλο ενδιαφέρον της διεθνής ερευνητικής κοινότητας στον προσδιορισμό, μοντελοποίηση και επίλυση πλήθους παραλλαγών του VRP δεν οφείλεται μόνο στη δυσκολία της συνδυαστικής βελτιστοποίησης αλλά επίσης, και στην πρακτικότητα του προβλήματος. Η συχνότητα εμφάνισης του VRP, σχεδόν σε κάθε εφαρμογή διακίνησης εντός της εφοδιαστικής αλυσίδας, έχει ως αποτέλεσμα την ανάγκη προσαρμογής του σε πλήθος διαφορετικών απαιτήσεων, οι οποίες επιδρούν τόσο στην μορφή του προβλήματος όσο και στον τρόπο επίλυσης του.

Το VRP εφαρμόζεται σε ένα δίκτυο σημείων, στους κόμβους του οποίου βρίσκονται οι πελάτες, με τα τόξα μεταξύ των κόμβων να αποτελούν τις διαδρομές μετάβασης μεταξύ τους. Επίσης η ζήτηση του κάθε πελάτη είναι γνωστή και η εξυπηρέτηση του πρέπει να γίνει μόνο από ένα όχημα. Στην ποιο βασική μορφή του, το ζητούμενο του VRP είναι η βέλτιστη δρομολόγηση, δηλαδή ελαχιστοποίηση της συνολικής απόστασης ή του συνολικού κόστους διανομής, ενός στόλου ομοιών οχημάτων από ένα κέντρο διανομής για την εξυπηρέτηση όλων των πελατών. Οι αποφάσεις συνεπώς που πρέπει να παρθούν στο βασικό VRP είναι ποιοι πελάτες θα περιλαμβάνονται σε κάθε δρομολόγιο και με ποια αλληλουχία θα γίνει η επίσκεψή τους. Η χρησιμότητα του βασικού VRP δεν εντοπίζεται τόσο στην πρακτικότητα του αλλά στη χρήση του ως βάση διάφορων παραλλαγών VRP, στις οποίες ανάλογα τις απαιτήσεις της εκάστοτε εφαρμογής γίνονται διάφορες προσθήκες που εξασφαλίζουν την ικανοποίηση τους.

Μία από τις συχνότερες απαιτήσεις είναι η ύπαρξη μέγιστης χωρητικότητας των οχημάτων σε μάζα, όγκο ή αριθμό προϊόντων. Τα VRP που ενσωματώνουν τις απαιτήσεις χωρητικότητας χαρακτηρίζονται ως VRP περιορισμένης χωρητικότητας ή CVRP (capacitated VRP) και αποτελούν την πιο μελετημένη παραλλαγή του VRP. Στο CVRP απαιτείται λήψη επιπλέον αποφάσεων σχετικά με τις ποσότητες φόρτωσης σε κάθε όχημα. Αξίζει να σημειωθεί πως περιορισμοί χωρητικότητας εντοπίζονται στην πλειοψηφία των εφαρμογών VRP και συνεπώς θεωρούνται αυτονόητοι σχεδόν σε κάθε παραλλαγής του

Η άνοδος των reverse logistics και του ηλεκτρονικού εμπορίου έχουν αναδείξει πως κατά την διακίνηση εντός της εφοδιαστικής αλυσίδας κατά τη διάρκεια δρομολόγιων συχνά απαιτείται συνδυασμός παραδόσεων και φορτώσεων. Τα VRP στα οποία πραγματοποιούνται φορτώσεις και παραδόσεις χαρακτηρίζονται ως VRP συλλογής παράδοσης ή VRPPD (VRP with Pickup and Delivery). Τα VRPPD χωρίζονται σε δύο μεγάλες υποκατηγορίες ανάλογα με την προέλευση και προορισμό των διακινούμενων προϊόντων. Η πρώτη κατηγορία είναι τα VRP οπισθόζευξης ή VRPB (Vehicle Routing With Backhauls) στα οποία τα προϊόντα μεταφέρονται από το κέντρο διανομής προς τους πελάτες και το αντίστροφο. Η δεύτερη κατηγορία είναι τα προβλήματα συλλογής παράδοσης ή PDP (Pickup and Delivery Problem) στα οποία μεταφορά προϊόντων λαμβάνει χώρο μόνο μεταξύ των πελατών. Ανεξάρτητα της κατηγορίας VRPPD απαιτούνται επιπλέον αποφάσεις σχετικά με την αλληλουχία των σημείων φόρτωσης και παράδοσης, έχοντας υπόψιν και την δυναμική αλλαγή της χωρητικότητας των οχημάτων κατά μήκος ενός δρομολογίου.



Σχήμα 2.3: Οι βασικές παραλλαγές του VRP.

Μια ακόμα αρκετά σημαντική παράμετρος των VRP, ειδικά τα τελευταία χρόνια με την ανάδειξη της σημασίας του «τελευταίου χιλιόμετρου» της εφοδιαστικής αλυσίδας, είναι η διάσταση του χρόνου. Ρεαλιστικά οι περισσότεροι πελάτες μπορούν να εξυπηρετηθούν μόνο εντός ενός χρονικού πλαισίου ενώ χρονικά όρια εντοπίζονται επίσης και στην διάρκεια

των δρομολογίων, είτε λόγω του προγράμματος των οδηγών είτε λόγω της νομοθεσίας. Με την προσθήκη χρονικής διάστασης, μεγέθη όπως χρόνοι φόρτωσης, εκφόρτωσης και αναμονής πρέπει να ληφθούν υπόψιν. Τα VRP με χρονική διάσταση χαρακτηρίζονται ως VRP χρονικών παραθύρων ή VRPTW (Vehicle Routing Problem with Time Windows). Στα VRPTW πρέπει να παρθούν αποφάσεις για το πότε θα γίνουν συγκεκριμένες ενέργειες , όπως η ώρα αναχώρησης των οχημάτων από το κέντρο διανομής και η ώρα εξυπηρέτησης του κάθε πελάτη.

2.3.3 Μοντελοποίηση του CVRP

Σε αυτή την ενότητα γίνεται παρουσίαση δύο μοντέλων του ασύμμετρου CVRP ως πρόβλημα γραμμικού ακεραίου προγραμματισμού. Το πρώτο μοντέλο αποτελεί επέκταση του TSP μοντέλου των Miller, Tucker και Zemlin [3]. Τα δεδομένα του προβλήματος είναι:

$\mathbf{n} \rightarrow$ ο αριθμός των πελατών
$\mathbf{N} \rightarrow$ το σύνολο των πόλεων, όπου $\mathrm{N} = \{1,2,\ldots,\mathrm{n}\}$
$\mathbf{k} \rightarrow$ ο αριθμός των οχημάτων
$\mathbf{K} \rightarrow$ σύνολο οχημάτων, όπου $\mathrm{K} = \{1,2,\ldots,\mathrm{k}\}$
$\mathbf{O} \rightarrow$ η πόλη αφετηρία, όπου $\mathrm{O} = \{0\}$
$\mathbf{V} \rightarrow$ το σύνολο των κόμβων, όπου $\mathrm{V} = \mathrm{O} \cup \mathrm{N}$
$\mathbf{A} \rightarrow$ σύνολο τόξων μεταξύ κόμβων, όπου $\mathrm{A} = \{(\mathrm{i},\mathrm{j}) \in \mathrm{V}^2 : \mathrm{i} \neq \mathrm{j}\}$
$\mathbf{c_{ij}} \rightarrow$ το κόστος μετάβασης διαμέσου του τόξου $(\mathrm{i},\mathrm{j}) \in \mathrm{A}$
$\mathbf{Q} \rightarrow$ η μέγιστη χωρητικότητα ενός οχήματος
$\mathbf{q_i} \rightarrow$ η ζήτηση του πελάτη $\mathrm{i} \in \mathrm{N}$

Οι μεταβλητές απόφασης και οι βοηθητικές μεταβλητές του μοντέλου ορίζονται ως :

$\boldsymbol{x_{ij}} \rightarrow$ *δυαδική μεταβλητή απόφασης που παίρνει την τιμή* 1 *όταν το τόξο* (i,j) *χρησιμοποιήται,* $(i,j) \in A$

$\boldsymbol{u_i} \rightarrow$ μη αρνητική ακέραια βοηθητική μεταβλητή, $i = (1,2,\ldots,n)$. Η μεταβλητή αυτή εκφράζει την ποσότητα προϊόντος που πρέπει να φορτωθεί στο που επισκέπτεται τον πελάτη i , και είναι ίση με τη ζήτηση του πελάτη i συν τις ζητήσεις όλων των πελατών που προηγούνται

Το μοντέλο βελτιστοποίησης του προβλήματος θα είναι :

$$minimize \sum_{(i,j \in A)} c_{ij} x_{ij} \quad (1)$$

Υπό τους περιορισμούς:

$$\sum_{j \in \delta^+(i)} x_{ij} = 1 \ , \forall \ i \in N \quad (2)$$

$$\sum_{i \in \delta^-(j)} x_{ij} = 1 \ , \forall \ j \in N \quad (3)$$

$$\sum_{j \in \delta^+(0)} x_{0j} = |K| \quad (4)$$

$$u_i - u_j + Q x_{ij} \leq Q - q_j \ , \forall \ (i,j) \in A(N) \quad (5)$$

$$q_i \leq u_i \leq Q \ , \forall \ i \in N \quad (6)$$

$$x_{i,j} \ binary \quad και \ u_i > 0 \ integer \quad (7)$$

Πέρα από κάποιες τροποποιήσεις των περιορισμών (4) και (5) καθώς και η προσθήκη του περιορισμού (6) το CVRP μοντέλο είναι παρόμοιο με το TSP. Η αντικειμενική συνάρτηση (1) ορίζεται ως η εύρεση του ελάχιστου κόστους δρομολόγησης. Οι περιορισμοί (2) και (3) εξασφαλίζουν πως σε κάθε σημείο (πελάτη) αντιστοιχεί μόνο ένα τόξο εξόδου και εισόδου αντίστοιχα. Ο περιορισμός (4) φροντίζει πως θα δημιουργηθούν τόσες διαδρομές όσες ο αριθμός των οχημάτων. Άμα δεν είναι απαραίτητο κάθε όχημα να συμμετάσχει στην δρομολόγηση, το «=» μπορεί να αντικατασταθεί με «≤». Ο περιορισμός (5) εξασφαλίζει πως κατά μήκος μίας διαδρομής η αθροιστική ζήτηση των πελατών δεν θα υπερβαίνει την μέγιστη χωρητικότητα οχήματος και ταυτόχρονα αποτρέπει τον σχηματισμό των κλειστών διαδρομών που δεν επικοινωνούν με το {0} (subtour elimination). Ο περιορισμός (6) θέτει ένα κάτω και άνω όριο στην βοηθητική μεταβλητή u_i.

Για την απόδειξη της λειτουργίας του περιορισμού (5), παρομοίως με το TSP, έστω μία κλειστή διαδρομή S_m η οποία αποτελείται από m σημεία και δεν διέρχεται από το 0. Ξεκινώντας από ένα τυχαίο σημείο της διαδρομής, έστω m_1, δημιουργώντας τις ανισότητες του περιορισμού (5) για κάθε σημείο και αθροίζοντας τες κατά μέλη προκύπτει:

$$Q\sum_{(i,j)\in S_m} x_{ij} \le Qm - \sum_{j\in S_m} q_j \to Qm \le Qm - \sum_{j\in S_m} q_j \to 0 \le -\sum_{j\in S_m} q_j \quad \text{άτοπο}$$

Ακόμα έστω μία κλειστή διαδρομή S_m της μορφής $(0,1,2,\dots,m,0)$ στην οποία $\sum q > Q$. Εφαρμόζοντας τον περιορισμό (5), αρχίζοντας από το σημείο 1, σε κάθε σημείο της διαδρομής δημιουργούνται m-1 ανισότητες (τα τόξα που περιέχουν το 0 δεν παράγουν περιορισμούς) και προσθέτοντας τες κατά μέλη προκύπτει:

$$u_1 - u_m + (m-1)Q \le (m-1)Q - \sum q \to u_1 - u_m \le -\sum q \to u_m - u_1 \ge \sum q > Q$$

Το οποίο λόγω του περιορισμού (6) είναι άτοπο.

Η παραπάνω μοντελοποίηση του CVRP παρουσιάζει παρόμοια προβλήματα με το TSP, του οποίου αποτελεί επέκταση. Οι Toth και Vigo [8] αναφέρουν πως το συγκεκριμένο μοντέλο κατά την γραμμική του χαλάρωση δίνει αδύναμα άνω και κάτω όρια, καθιστώντας την εύρεση της βέλτιστης λύσης αρκετά δύσκολη για μεγάλα n. Θετικό στοιχείο της συγκεκριμένης μοντελοποίησης είναι πως χωρίς πολλές αλλαγές στο μοντέλο με την πρόσθεση ενός επιπλέον δείκτη στις μεταβλητές x και u είναι δυνατή ή μοντελοποίηση η ενός CVRP με στόλο ανόμοιων οχημάτων [9].

Μία λύση στην αδύναμη γραμμική χαλάρωση του μοντέλου CVRP των Miller, Tucker και Zemlin αποτελεί το εναλλακτικό μοντέλο των Laporte, Mercure και Nobert το οποίο βασίζεται στη δυναμική δημιουργία *περιορισμών callback*. Για το συγκεκριμένο μοντέλο πρέπει να γίνει η εισαγωγή κάποιων επιπλέων μεγεθών :

$\boldsymbol{S} \to$ *μία διαδρομή, όπου* $S \subseteq V$

$\boldsymbol{q_s} \to$ *η συνολική ζήτηση μίας διαδρομής* $S = \sum_{i\in S} q_i$

Το νέο CVRP μοντέλο βελτιστοποίησης του προβλήματος είναι :

$$minimize \sum_{(i,j \in A)} c_{ij} x_{ij} \quad (1)$$

Υπό τους περιορισμούς

$$\sum_{j \in \delta^+(i)} x_{ij} = 1 \ , \forall \ i \in N \quad (2)$$

$$\sum_{i \in \delta^-(j)} x_{ij} = 1 \ , \forall \ j \in N \quad (3)$$

$$\sum_{j \in \delta^+(0)} x_{0j} = |K| \quad (4)$$

$$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq r(s) \ \ , \forall \ S \subseteq N \ , S \neq \emptyset \ , \text{όπου } r(s) = \left\lceil {}^{q(s)}/_{Q} \right\rceil \quad (5)$$

$$x_{ij} \ binary \quad (6)$$

Σε σύγκριση με το μοντέλο των Miller, Tucker και Zemlin στο μοντέλο των Laporte, Mercure και Nobert δεν γίνεται χρήση της βοηθητικής μεταβλητής u και οι περιορισμοί που την περιχέουν αντικαταστάθηκαν από τον περιορισμό (5). Ο περιορισμός (5) εξασφαλίζει πως κατά μήκος μίας διαδρομής η αθροιστική ζήτηση των πελατών δεν θα υπερβαίνει την μέγιστη χωρητικότητα οχήματος και ταυτόχρονα αποτρέπει τον σχηματισμό των κλειστών διαδρομών που δεν επικοινωνούν με το {0} (subtour elimination).

Το αριστερό μέλος του περιορισμού (5) αποτελεί το άθροισμα των μεταβλητών x_{ij} των τόξων που εξέρχονται από τα σημεία της διαδρομή S (εκτός του {0}) και δεν καταλήγουν σε κάποιο άλλο σημείο της διαδρομής S. Στην περίπτωση όπου δεν υπάρχει ζήτημα παράβασης της μέγιστης χωρητικότητας οχήματος, άρα $r(s) = 1$, και η διαδρομή S δεν επικοινωνεί με το {0}, τότε αναγκαστικά τουλάχιστον ένα από τα τόξα εξόδου θα πρέπει να χρησιμοποιηθεί και άρα αναγκαστικά η συγκεκριμένη διαδρομή δεν είναι εφικτή. Διαδρομές

που των οποίων το άθροισμα των ζητήσεων υπερβαίνει την μέγιστη χωρητικότητα Q, αυξάνουν την τιμή του $r(s)$ με αποτέλεσμα δύο ή παραπάνω τόξα εξόδου να πρέπει να χρησιμοποιηθούν καθιστώντας την S μη εφικτή.

Ο περιορισμός (5) είναι *περιορισμός callback*. Όπως και στο TSP η εύρεση των S γίνεται αλγοριθμικά, λύνοντας το μοντέλο χωρίς τον περιορισμό (5) και εξετάζοντας εάν οι διαδρομές που προκύπτουν από την λύση του τον παραβιάζουν. Για κάθε διαδρομή που τον παραβιάζει γίνεται προσθήκη του αντίστοιχου περιορισμού στο αρχικό μοντέλο. Η διαδικασία της επίλυσης και πρόσθεσης νέων περιορισμών επαναλαμβάνεται και τερματίζει όταν στην λύση του μοντέλου καμία διαδρομή δεν παραβιάζει τον περιορισμό (5). Η λύση αυτή είναι η βέλτιστη λύση του CVRP.

2.4 Το πρόβλημα συλλογής-παράδοσης (GPDP)

2.4.1 Περιγραφή και ιστορία του GPDP

Το γενικό πρόβλημα συλλογής-παράδοσης ή GPDP (General Pickup and Delivery Problem) αποτελεί μία συνήθης παραλλαγή του VRP, η οποία συναντάται και με την ονομασία VRPPD (VRP with Pickups and Deliveries). Το GPDP αφορά την εύρεση των βέλτιστων δρομολογίων παράδοσης και συλλογής σε ένα δίκτυο σημείων προέλευσης ή φόρτωσης (pickup) και προορισμού ή παράδοσης (delivery) υπό διάφορους περιορισμούς.

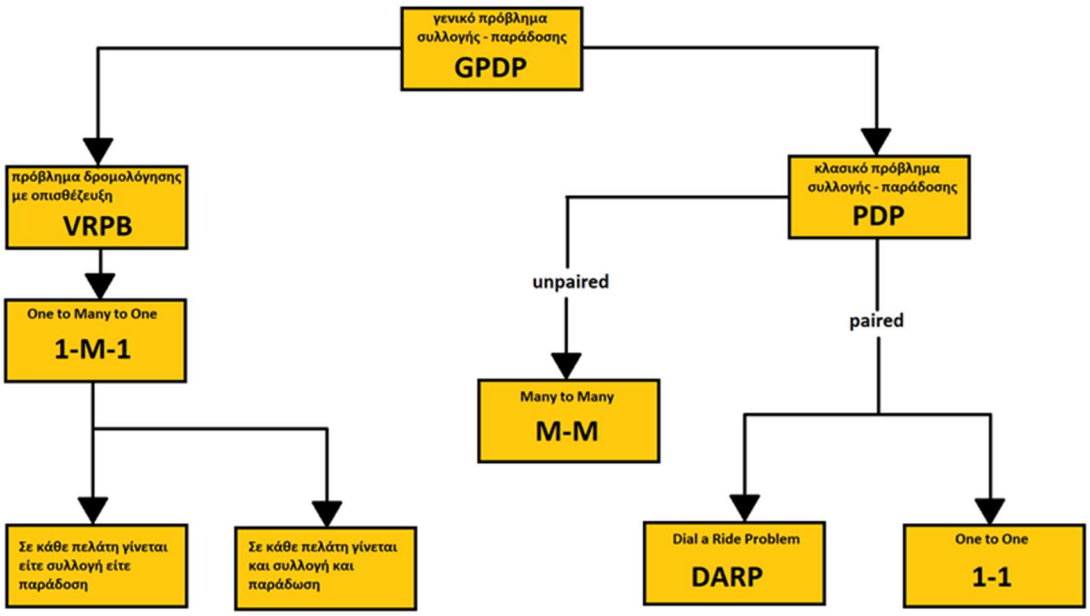
Η πρώτη αναφορά στο πρόβλημα γίνεται το 1985 από τους Kalantari et al [11] και αφορά την αλγοριθμική επίλυση του TSP με σημεία συλλογής και παράδοσης. Η εκτενής έρευνα, τις ακόλουθες δεκαετίες, πάνω στην μοντελοποίηση και βελτιστοποίηση του VRP συντελεί στην ανάπτυξη έντονου ερευνητικού ενδιαφέροντος και γύρω από το GPDP, κυρίως στον τομέα των εμπορευματικών μεταφορών. Την τελευταία δεκαετία η άνοδος του ηλεκτρονικού εμπόριού σε συνδυασμό με την ανάδειξη της σημασίας του τελευταίου μιλίου (last mile) της εφοδιαστικής αλυσίδας, καθιστούν την επίλυση του GPDP επίκαιρη πρόκληση για όσες επιχειρήσεις επιθυμούν να αξιοποιήσουν πλήρως το δίκτυο λειτουργίας τους, μειώνοντας ταυτόχρονα το κόστος μεταφοράς.

2.4.2 Παραλλαγές του GPDP

Η ονομασία GPDP καλύπτει ένα εκτενές εύρος αρκετά διαφορετικών εφαρμογών της εφοδιαστικής αλυσίδας, δημιουργώντας την ανάγκη του ορισμού κάποιων κύριων

υποκατηγοριών για την αποτελεσματικότερη μελέτη και κατανόηση του προβλήματος. Ο ορισμός των υποκατηγοριών θα γίνει με βάση τη σχέση μεταξύ των σημείων συλλογής και παράδοσης, υπό την προϋπόθεση πως υφίστανται περιορισμοί μέγιστης χωρητικότητας των οχημάτων και ύπαρξη χρονικών παραθύρων συλλογής και παράδοσης.

Το GPDP (ή VRPPD) έχει δύο βασικές υποκατηγορίες. Τα προβλήματα που ανήκουν στην πρώτη υποκατηγορία ονομάζονται VRP οπισθέζευξης ή VRPB (VRP with Backhauls) και αφορούν τις εφαρμογές κατά τις οποίες τα αντικείμενα προς παράδοση φορτώνονται στα οχήματα σε ένα η περισσότερα κέντρα διανομής και όλα τα αντικείμενα που συλλέγονται κατά την διαδρομή μεταφέρονται πίσω στα κέντρα διανομής. Όταν το προϊόν προς παράδοση και το προϊόν συλλογής είναι διαφορετικά το πρόβλημα συναντάται και με την ονομασία one-to many-to one ή (1-M-1). Το VRPB επίσης εμφανίζει παραλλαγές, οι οποίες ορίζονται ανάλογα με τον ρόλο των πελατών, οι οποίοι είτε δέχονται μόνο να παραλάβουν/φορτώσουν προϊόντα είτε πραγματοποιούν και τα δύο.



Σχήμα 2.4: Οι βασικές παραλλαγές του GPDP

Η δεύτερη υποκατηγορία αφορά τα κλασικά προβλήματα συλλογής-παράδοσης ή PDP κατά τα οποία η μεταφορά προϊόντων (συλλογή-παράδοση) πραγματοποιείται μεταξύ των πελατών. Τα PDP περεταίρω διαχωρίζονται με βάση το εάν υπάρχει σύζευξη (pairing) των σημείων συλλογής και παράδοσης, δηλαδή εάν σε κάθε σημείο συλλογή αντιστοιχεί

συγκεκριμένο σημείο παράδοσης. Όταν υπάρχει σύζευξη τα προβλήματα χαρακτηρίζονται ως one-to-one ή (1-1), όταν αφοράν μεταφορά προϊόντων, και Dial-A-Ride Problem (DARP) όταν αφοράν τη μεταφορά επιβατών μεταξύ σημείων. Τα PDP που δεν παρουσιάζουν σύζευξη αποκαλούνται many-to-many ή M-M, στα οποία συνήθως πραγματοποιείται συλλογή και διανομή ενός ενιαίου προϊόντος ελεύθερα μεταξύ των πελατών.

2.4.3 Μοντελοποίηση του (1-1) PDP

Σε αυτή την ενότητα γίνεται παρουσίαση δύο μοντέλων του ασύμμετρου (1-1) PDP ως πρόβλημα γραμμικού ακεραίου προγραμματισμού. Το πρώτο μοντέλο βασίζεται στην μορφοποίηση του VRPTW από τους Cordeau et al. [12] , στην οποία έχει γίνει προσθήκη κάποιων περιορισμών σχετικών με το PDP [13]. Τα δεδομένα του προβλήματος είναι :

$\boldsymbol{n}$ → ο αριθμός των κόμβων που απαιτούν pick up

$\boldsymbol{\hat{n}}$ → ο αριθμός των κόμβων που απαιτούν delivery, στο $(1-1)$PDP ισχύει $n = \hat{n}$

$\boldsymbol{P}$ → το σύνολο των κόμβων που απαιτούν pick up, όπου $\mathrm{P} = \{1,2,\ldots,\mathrm{n}\}$

$\boldsymbol{D}$ → το σύνολο των κόμβων που απαιτούν delivery, όπου $\mathrm{D} = \{\mathrm{n}+1,\ldots,\mathrm{n}+\hat{n}\}$

$\mathbf{V}$ → το σύνολο των κόμβων , όπου $\mathrm{V} = \{0, \mathrm{n}+\hat{\mathrm{n}}+1\} \cup \mathrm{P} \cup \mathrm{DV}$, με 0 το σημείο εκκίνησης και $\mathrm{n}+\hat{\mathrm{n}}+1$ το σημείο τερματισμο

$\mathbf{A}$ → το σύνολο των τόξων , όπου $\mathrm{A} = \{(\mathrm{i},\mathrm{j}) \in \mathrm{V2}: \mathrm{i} \neq \mathrm{j}, \mathrm{i} \neq \mathrm{n}+\hat{\mathrm{n}}+1, \mathrm{j} \neq 0\}$

$\boldsymbol{K}$ → το σύνολο οχημάτων

$\boldsymbol{c_{ij}}$ → το κόστος μετάβασης διαμέσου του τόξου $(\mathrm{i},\mathrm{j}) \in \mathrm{A}(\mathrm{i},\mathrm{j}) \in \mathrm{A}$

$\boldsymbol{q_i}$ → η ποσότητα συλλογής $(\mathrm{q_i} > 0)$ ή παράδοσης $(\mathrm{q_i} < 0)$στον κόμβο i, όπου $\mathrm{i} \in \mathrm{V}$ και $\mathrm{q}_0 = \mathrm{q}_{\mathrm{n}+\hat{\mathrm{n}}+1} = 0$

$\boldsymbol{e_i}$ → ο ελάχιστος χρόνος στον οποίο είναι δυνατή η εξυπηρέτηση του κόμβου i, με $\mathrm{i} \in \mathrm{V}$

$\boldsymbol{l_i}$ → ο μέγιστος χρόνος στον οποίο είναι δυνατή η εξυπηρέτηση του κόμβου i, με $\mathrm{i} \in \mathrm{V}$

$\boldsymbol{d_i}$ → η διάρκεια εξυπηρέτησης στον κόμβο i, με $\mathrm{i} \in \mathrm{V}$

$\boldsymbol{t_{ijk}}$ → *ο χ*ρόνος μετάβασης του οχήματος k από τον κόμβο i στον j

$\boldsymbol{c_{ijk}}$ → το κόστος μετάβασης του οχήματος k από τον κόμβο i στον j

$\boldsymbol{C_k}$ → η μέγιστη χωρητικότητα του οχήματος k

$\boldsymbol{T_k}$ → η μέγιστη επιτρεπτή διάρκεια δρομολογίου του οχήματος k

Οι μεταβλητές απόφασης και οι βοηθητικές μεταβλητές του μοντέλου ορίζονται ως :

$\boldsymbol{x_{ijk}}$ → δυαδική μεταβλητή απόφασης, η οποία πέρνει την τιμή 1 όταν το τόξο $(\mathrm{i}, \mathrm{j}) \in \mathrm{A}$ χρησιμοποιείται από το όχημα k

$\boldsymbol{Q_{ik}}$ → η χωρητικότητα του οχήματος $\mathrm{k} \in \mathrm{K}$ όταν εγκαταλείπει τον κόμβο $\mathrm{i} \in \mathrm{V}$

$\boldsymbol{D_{ik}}$ → ο χρόνος έναρξης της εξυπηρέτησης του κόμβου $\mathrm{i} \in \mathrm{V}$ από το όχημα $\mathrm{k} \in \mathrm{K}$

Το μοντέλο βελτιστοποίησης του προβλήματος θα είναι :

$$minimize \sum_{k \in K} \sum_{(i,j) \in A} c_{ijk} x_{ijk} \quad (1)$$

Υπό τους περιορισμούς

$$\sum_{k \in K} \sum_{j \in (i,j) \in A} x_{ijk} = 1 \,, \forall i \in P \cup D \quad (2)$$

$$\sum_{j \in (0,j) \in A} x_{0jk} = 1 \,, \forall k \in K \quad (3)$$

$$\sum_{i \in (i,n+\hat{n}+1) \in A} x_{i(n+\hat{n}+1)k} = 1 \,, \forall k \in K \quad (4)$$

$$\sum_{i \in (i,j) \in A} x_{ijk} - \sum_{i \in (j,i) \in A} x_{jik} = 0 \;\,, \forall j \in P \cup D, \forall k \in K \quad (5)$$

$$if\ x_{ijk} = 1 \rightarrow D_{jk} \geq D_{ik} + d_i + t_{ijk} \,, \forall (i,j) \in A, \forall k \in K \quad (6)$$

$$if\ x_{ijk} = 1 \rightarrow Q_j \geq Q_i + q_j \;\,, \forall (i,j) \in A, \forall k \in K \quad (7)$$

$$max(q_i, 0) \leq Q_{ik} \leq min(C_k, C_k + q_i) \quad (8)$$

$$\sum_{j\in(i,j)\in A} x_{ijk} - \sum_{j\in(n+1,j)\in A} x_{(n+i)jk} = 0 \ , \forall i \in P, \forall k \in K \quad (9)$$

$$D_{ik} \leq D_{(n+1)k} \ , \forall i \in P, \forall k \in K \quad (10)$$

$$e_i \leq D_{ik} \leq l_i \ , \forall i \in V, \forall k \in K \quad (11)$$

$$B_{(n+\hat{n}+1)k} - B_{0k} \leq T_k \ , \forall k \in K \quad (12)$$

$$x_{ijk} \ binary \quad (13)$$

$$Q_{ik}, D_{ik} > 0 \quad (14)$$

Η αντικειμενική συνάρτηση (1) και οι περιορισμοί από (1) έως (8) είναι ουσιαστικά η μοντελοποίηση 3 δεικτών του VRPTW [12]. Η αντικειμενική (1) ελαχιστοποιεί το συνολικό κόστος δρομολόγησης όλων των οχημάτων. Οι ισότητες που προκύπτουν από το (2) εξασφαλίζουν πως κάθε σημείο θα δεχθεί μία μόνο επίσκεψη από μόνο ένα όχημα. Οι περιορισμοί (3) και (4) φροντίζουν κάθε πως όχημα ξεκινάει από την αφετηρία και καταλήγει στον τερματισμό, χωρίς αυτό ταυτόχρονα να συνεπάγεται πως κάθε όχημα χρησιμοποιείται αναγκάστηκα. Εάν κατά την επίλυση η χρήση κάποιου οχήματος θεωρηθεί ασύμφορη τότε αυτό χρησιμοποιεί την διαδρομή $(0, \mathrm{n} + \hat{\mathrm{n}} \ + 1)$. Ο (5) είναι τυπικός περιορισμός διατήρησης ροής. Ο περιορισμός (6) εισάγει την χρονική μεταβλητή D_{ik} στο μοντέλο και ο ρόλος του είναι η αποτροπή λειτουργίας κλειστών διαδρομών, βάση του ότι $\left(d_{ij} + t_{ijk}\right) > 0$ για κάθε $(i, j) \in A$. Οι περιορισμοί (7) και (8) διασφαλίζουν πως το όριο μέγιστης χωρητικότητας κάθε οχήματος τηρείται κατά τη διάρκεια του δρομολογίου του. Αν και οι περιορισμοί (6) και (7) δεν είναι γραμμικοί, μπορούν ευκολά να γίνουν μέσω της μεθόδου M. Ο περιορισμός (9) τροποποιεί το μοντέλο ώστε να καλύπτει τις ανάγκες του (1-1)PDP επιβάλλοντας σε κάθε όχημα που επισκέπτεται ένα σημείο συλλογής $i \in P$ να επισκεφθεί και το αντίστοιχο σημείο παράδοσης $i + n$, ενώ ο περιορισμός (10) εξασφαλίζει πως η κάθε συλλογή γίνεται πριν την αντίστοιχη παράδοση. Τελικώς ο περιορισμός (11) αφορά την τήρηση χρονικών παραθύρων και ο περιορισμός (12) την μέγιστη διάρκεια δρομολογίου.

Η παραπάνω μοντελοποίηση έχει το πλεονέκτημα πως μπορεί να χρησιμοποιηθεί σε προβλήματα με στόλο ανόμοιων οχημάτων, τόσο στην μέγιστη χωρητικότητα C_k όσο και στο κόστος λειτουργίας τους (κατανάλωση) c_{ijk}. Μειονέκτημα του μοντέλου είναι πως η αύξηση του n επιφέρει τετραγωνική αύξηση του αριθμού των μεταβλητών και περιορισμών μειώνοντας σημαντικά την απόδοση του μοντέλου. Στην περίπτωση όμοιου στόλου οχημάτων οι Ropke et al. [14] προτείνουν μία εναλλακτική μορφοποίηση του προβλήματος, η οποία περιέχει λιγότερες μεταβλητές και δημιουργεί στενότερα όρια.

Επειδή το πρόβλημα είναι (1-1)PDP θεωρείται πως $n = \hat{n} \rightarrow n + \hat{n} + 1 = 2n + 1$. Αν και τα περισσότερα δεδομένα και οι συμβολισμοί τους είναι ίδια με αυτά του πρώτου μοντέλου, χωρίς τον δείκτη k, θα χρειαστεί να γίνει η εισαγωγή ορισμένων επιπλέων μεγεθών:

$\mathbf{S}$ → μία διαδρομή (αλληλουχία σημείων) στην οποία $0 \in \mathrm{S}$, $2\mathrm{n} + 1 \notin \mathrm{S}$
και που περιέχει τουλάχιστον ένα σημείο $\mathrm{i} \in \mathrm{P}$ τέτοιο ώστε $\mathrm{i} \notin \mathrm{S}$ και $\mathrm{n} + \mathrm{i} \in \mathrm{S}$

$\boldsymbol{\mathcal{S}}$ → το σύνολο που περιέχει τις διαδρομές S

$\boldsymbol{\mathcal{R}}$ → το σύνολο που περιέχει τις ανέφικτες διαδρομές, ώσον αναφορά τους χρονικούςπεριορισμούς (χρονικά παράθυρα, μέγιστη διάρκεια διαδρομής)

$\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{R}})$ → το σύνολο που περιέχει τα τόξα $(\mathrm{i}, \mathrm{j}) \in \mathrm{A}$ τα οποία περιέχει μια διαδρομή $\mathrm{R} \in \mathcal{R}$

Ακόμα ορίζω: $q(S) = \sum_{i \in S} q_i$

Το μοντέλο βελτιστοποίησης του προβλήματος θα είναι :

$$minimize \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \quad (1)$$

Υπό τους περιορισμούς :

$$\sum_{i \in N} x_{ij} = 1 \, , \forall j \in P \cup D \quad (2)$$

$$\sum_{j \in N} x_{ij} = 1 \, , \forall i \in P \cup D \quad (3)$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \, , \forall S \in \mathcal{S} \quad (4)$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - max\left\{1, \left\lceil \left\| \frac{q(S)}{Q} \right\| \right\rceil \right\} \, , \forall S \subseteq N \backslash \{0, 2n+1\}, |S| \geq 2 \quad (5)$$

$$\sum_{(i,j) \in A(R)} x_{ij} \leq |A(R)| - 1 \, , \forall R \in \mathcal{R} \quad (6)$$

$$x_{ij} \;\; binary \quad (7)$$

Η αντικειμενική συνάρτηση (1) ελαχιστοποιεί το συνολικό κόστος δρομολόγησης. Οι περιορισμοί (1) και (2) φροντίζουν πως σε κάθε σημείο, εκτός της αφετηρίας και του τερματισμού, υπάρχει μόνο ένα τόξο εισόδου και εξόδου. Ο περιορισμός (4) εξασφαλίζει το ότι εάν κατά μήκος μίας διαδρομής γίνει επίσκεψη σε έναν κόμβο συλλογής $\mathrm{i} \in \mathrm{P}$ πως στη συνέχεια κατά το μήκος της ίδιας διαδρομής γίνεται επίσκεψη του αντίστοιχου κόμβου συλλογής $i + n$. Ο περιορισμός (5) αφορά την τήρηση τού ορίου χωρητικότητας των οχημάτων κατά μήκος της διαδρομής καθώς και την εξάλειψη κλειστών διαδρομών. Τελικώς ο περιορισμός (6) διασφαλίζει πως οι διαδρομές σέβονται του χρονικούς περιορισμούς, δηλαδή την τήρηση των χρονικών παράθυρων εξυπηρέτησης και της μέγιστης διάρκειας δρομολογίου.

Οι περιορισμοί (4),(5) και (6) είναι περιορισμοί callback. Η διαδικασία που ακολουθείται στην επίλυση του προβλήματος δεν διαφέρει πολύ από αυτή που περιεγράφηκε στο TSP και CVRP. Αρχικά γίνεται η επίλυση του προβλήματος μόνο με τους περιορισμούς (2) και (3) και στις διαδρομές που προκύπτουν εφαρμόζονται οι κατάλληλοι περιορισμοί. Εάν μια διαδρομή πληροί τις προϋποθέσεις ώστε να ανήκει στο σύνολο $\mathcal{S}$, τότε προστίθεται ο περιορισμός (4) για τα σημεία της συγκεκριμένης διαδρομής. Ο περιορισμός (5) προστίθεται για τα σημεία κάθε κλειστής διαδρομής καθώς και για τα κομμάτια μη κλειστών διαδρομών που παραβιάζουν τους περιορισμούς χωρητικότητας (εάν ληφθούν

υπόψιν όλα τα σημεία της διαδρομής τότε ο περιορισμός δεν έχει κάποια επίδραση). Ο περιορισμός (6) προστίθεται για κάθε διαδρομή που παραβιάζει τους χρονικούς περιορισμούς. Αφού γίνει η πρόσθεση των κατάλληλων περιορισμών η διαδικασία επαναλαμβάνεται ως ότου οι διαδρομές που προκύπτουν από κάποια λύση δεν παραβιάζουν κανέναν από τους περιορισμούς, γεγονός που σημαίνει πως έχει βρεθεί η λύση του PDP.

Αν και η λειτουργία των περιορισμών (4) και (6) είναι προφανής, δεν μπορεί να ειπωθεί το ίδιο και για τον περιορισμό (5). Έστω μια κλειστή διαδρομή S, η οποία δεν έχει ζητήματα παραβίασης χωρητικότητας ($q(S) = 0$). Έστω η διαδρομή περιέχει m σημεία και άρα m ενεργά τόξα. Ο περιορισμός (5) παραβιάζεται καθώς $m \leq m - \max(1,0) \rightarrow m \leq m - 1$.

Έστω τώρα τμήμα μίας μη κλειστής διαδρομής S, το οποίο περιέχει m σημεία θα περιέχει και συνεπώς m-1 ενεργά τόξα. Στην περίπτωση που στο τμήμα αυτό η χωρητικότητα του οχήματος υπερβαίνει την μέγιστη τότε $\lceil |q(S)/Q| \rceil \geq 2$. Ο περιορισμός (5) παραβιάζεται καθώς: $m - 1 \leq m - \max(1,2) \rightarrow m - 1 \leq m - 2$.

2.5 Το πρόβλημα δρομολόγησης αποθέματος (IRP)

2.5.1 Περιγραφή και ιστορία του IRP

Για την βελτίωση της απόδοσης της εφοδιαστικής αλυσίδας, οι προμηθευτές σε ένα μεγάλο εύρος του παραγωγικού τομέα στρέφονται σε πρακτικές VMI (Vendor-Managed Inventory). Οι πρακτικές αυτές αναφέρονται σε μία συμφωνία μεταξύ προμηθευτή και πελάτη, στην οποία ο πελάτης επιτρέπει στον προμηθευτή να επιλέξει τον χρόνο και όγκο παράδοσης φροντίζοντας ταυτοχρόνως τον μη μηδενισμό του αποθέματος του πελάτη. Η υιοθέτηση μίας VMI πολιτικής ωφελεί τον προμηθευτή, καθώς έχει την δυνατότητα να συνδυάσει καλύτερα παραδώσεις αξιοποιώντας με μεγαλύτερη απόδοση τον στόλο και πόρους του και μειώνοντας εντέλει το κόστος διανομής της εφοδιαστικής του αλυσίδας. Ταυτόχρονα κέρδος έχει και ο πελάτης, ο οποίος δεν αναλώνει χρόνο και πόρους για την διαχείριση του αποθέματος του.

Η εφαρμογή μίας πολιτικής VMI, παρόλα αυτά, δεν είναι απλή, γιατί απαιτεί την ταυτόχρονη επίλυση δύο απαιτητικών προβλημάτων της εφοδιαστικής αλυσίδας, τον έλεγχο αποθέματος (inventory control) και την δρομολόγηση παραδόσεων (distribution routing). Στον έλεγχο του αποθέματος ο στόχος είναι ο προσδιορισμός του χρόνου και ποσότητας

παραγγελίας κάθε πελάτη ενώ η δρομολόγηση παραδόσεων αφορά την δημιουργία προγράμματος για την ικανοποίηση των παραπάνω ζητήσεων. Η συνεπίλυση των προβλημάτων, που επηρεάζει δραματικά την απόδοση του συνολικού συστήματος, οδηγεί στον ορισμό του προβλήματος δρομολόγησης αποθέματος (Inventory Routing Problem) ή IRP το οποίο είναι η καρδιά όλων των VMI πολιτικών.

Αρχικά το IRP αντιμετωπίζονταν ως μία παραλλαγή VRP μοντέλων, στα οποία οι απαιτήσεις αποθέματος εξασφαλίζονταν ευρετικά. Η πρώτη μελέτη του προβλήματος εντοπίζεται από τους Bell et al. (1983) και καλύπτει τα προβλήματα που παρουσιάζουν αποκλειστικά κόστη μεταφοράς και στοχαστικές ζητήσεις. Μελέτες που ακολούθησαν επεκτάθηκαν σε προβλήματα IRP με τυχαίες ζητήσεις και διάφορα επιπρόσθετα κόστη, όπως κόστη αποθήκευσης, έλλειψης και εκκίνηση παραγωγής.

2.5.2 Παραλλαγές του IRP

Το IRP εντοπίζεται σε πλήθος διαφορετικών εφαρμογών, από την διανομή βενζίνης και πετροχημικών μέχρι και τον ανεφοδιασμό σούπερ μάρκετ και υποκαταστημάτων, η κάθε με τα δικά της ξεχωριστά χαρακτηριστικά. Λόγω της ιδιαιτερότητας κάθε εφαρμογής είναι δύσκολο να προσδιοριστεί ένα συγκεκριμένο πρότυπο IRP και αντί αυτού είναι καταλληλότερο να γίνει αναφορά σε κάποιες βασικές παραλλαγές του προβλήματος, οι οποίες συγκεντρώνουν το μεγαλύτερο ερευνητικό ενδιαφέρον. Οι παραλλαγές αυτές μπορούν να ταξινομηθούν βάση των παρακάτω κριτηρίων.

- Χρονικός ορίζοντας: Ο χρονικός ορίζοντας μπορεί να είναι πεπερασμένος ή άπειρος.
- Δομή: Σχετίζεται με τον αριθμό των προμηθευτών και πελατών (one-to-one, one-to-many, many-to-many).
- Δρομολόγηση: Διαφέρει ανάλογα το εάν κατά την διάρκεια ενός δρομολογίου εξυπηρετείται μόνο ένας ή παραπάνω από ένας πελάτες. Η δρομολόγηση επίσης μπορεί να είναι συνεχής χωρίς την ύπαρξη κάποιου κέντρου διανομής.
- Πολιτική αποθέματος: Οι πολιτικές που υιοθετούνται για τον ανεφοδιασμό των πελατών. Οι πιο συνηθισμένες πολιτικές είναι ή maximum level (ML) και η order up to level (OU). Η χρήση ML πολιτικής επιτρέπει ευελιξία όσον αφορά την ποσότητα ανεφοδιασμού στον πελάτη, με μόνο περιορισμό την χωρητικότητα σε απόθεμα του κάθε πελάτη. Χρήση OU πολιτικής σημαίνει πως η ποσότητα ανεφοδιασμού κατά την

επίσκεψη στον πελάτη πρέπει να είναι τέτοια ώστε να αναπληρώσει πλήρως την χωρητικότητα του σε απόθεμα.

• Αποφάσεις αποθέματος: Επηρεάζουν την μοντελοποίηση του αποθέματος, δηλαδή εάν επιτρέπεται να πάρει ή όχι αρνητικές τιμές (backorders ή χαμένες πωλήσεις).

• Σύσταση στόλου: Εάν τα οχήματα είναι ομοιογενή ή όχι.

• Μέγεθος στόλου: Ο αριθμός των οχημάτων.

• Ζήτηση: Εάν η ζήτηση είναι ντετερμινιστική, στοχαστική ή δυναμική.

2.5.3 Μοντελοποίηση του κλασικού IRP

Ως κλασικό ορίζεται το IRP του οποίου οι ζητήσεις είναι ντετερμινιστικές και γνωστές για έναν πεπερασμένο χρονικό ορίζοντα, ακολουθεί ML πολιτική αποθέματος, το οποίο απαγορεύεται να πάρει αρνητικές τιμές, και αφορά έναν στόλο ανόμοιων οχημάτων, τα οποία κατά τη διάρκεια ενός δρομολογίου μπορούν να εξυπηρετήσουν παραπάνω από έναν πελάτες. Το κλασικό IRP είναι συμμετρικό και αφορά τη δρομολόγηση στόλου από ένα μόνο κέντρο διανομής (one to many). Τελικώς σημειώνεται πως έχει γίνει η παραδοχή ότι ο προμηθευτής διαθέτει επαρκές απόθεμα για την εξυπηρέτηση όλων των πελατών εντός του χρονικού ορίζοντα του προβλήματος.

Παρακάτω παρουσιάζεται η μορφοποίηση του προβλήματος από τους Coelho και Laporte [16] ως πρόβλημα μεικτού ακέραιου προγραμματισμού. Τα δεδομένα του προβλήματος είναι:

$\mathbf{n} \rightarrow$ ο αριθμός των πελατών

$\mathbf{N} \rightarrow$ το σύνολο των πόλεων, όπου $N = \{1,2,\dots,n\}$

$\mathbf{V} \rightarrow$ το σύνολο των κόμβων, όπου $V = \{0\} \cup N$ με το $\{0\}$ να είναι η αφετηρία

$\mathbf{A} \rightarrow$ σύνολο τόξων μεταξύ κόμβων, όπου $A = \{(i,j) \in V^2 : i \neq j\}$

$\mathbf{c_{ij}} \rightarrow$ το κόστος μετάβασης διαμέσου του τόξου $(i,j) \in A$

$\mathbf{k} \rightarrow$ ο αριθμός των οχημάτων

$\mathbf{K} \rightarrow$ σύνολο οχημάτων, όπου $K = \{1,2,\dots,k\}$

$\mathbf{Q_k} \rightarrow$ η μέγιστη χωρητικότητα σε μονάδες προιόντος του οχήματος k

$\mathbf{p} \rightarrow$ διάρκεια του ορίζοντα σχεδίασης

$\mathbf{T} \rightarrow$ το σύνολο περιόδων, όπου $T = \{1,2,\dots,p\}$, όπου t μία συγκεκριμένη περίοδος του T

$\mathbf{r_t}$ → μονάδες προιόντος διαθέσιμες στον προμηθευτή την περίοδο t

$\mathbf{d_{it}}$ → η ζήτηση σε μονάδες προιόντος του πελάτη i ∈ N την περίοδο t

$\mathbf{C_i}$ → η μέγιστη χωρητικότητα αποθέματος σε μονάδες προιόντος i ∈ V

$\mathbf{h_i}$ → το κόστος αποθήκευσης μία μονάδας προιόντος για μία περίοδο στο απόθεμα του πελάτη i ∈ N

Οι μεταβλητές απόφασης και βοηθητικές μεταβλητές του μοντέλου ορίζονται:

$\mathbf{x_{ijkt}}$ → εκφράζει πόσες φορές το τόξο (i, j) ∈ A, όπου i < j, χρησιμοποιείται από το όχημα k την περίοδο t. Το x_{ijkt} μπορεί να πάρει τις τιμές {0,1,2}. Το x_{ijkt} μπορεί να πάρει την τιμή 2 μόνο εάν το i = 0.

$\mathbf{y_{ikt}}$ → βοηθητική δυαδική μεταβλητή, η οποία πέρνει την τιμή 1 όταν ο πελάτης i εξυπηρετείται απο το όχημα k την περίοδο t

$\mathbf{I_{it}}$ → απόθεμα σε μονάδες προιόντος του πελάτη i στο τέλος της περιόδου t

$\mathbf{q_{ikt}}$ → ποσότητα προιόντων που παραδίδει το όχημα k στον πελάτη i την περίοδο t

Το μοντέλο βελτιστοποίησης του προβλήματος θα είναι :

$$minimize \sum_{i\in V}\sum_{t\in T} h_i I_{it} + \sum_{i\in V}\sum_{j\in V, j>i}\sum_{k\in K}\sum_{t\in T} c_{ij} x_{ijkt} \quad (1)$$

Υπό τους περιορισμούς

$$I_{0t} = I_{0(t-1)} + r_t - \sum_{k\in K}\sum_{i\in N} q_{ikt} \ , \forall t \in T \quad (2)$$

$$I_{it} = I_{i(t-1)} - d_{it} + \sum_{k\in K} q_{ikt} \ , \forall i \in N, \forall t \in T \quad (3)$$

$$0 \le I_{it} \le C_i \ , \forall i \in V, \forall t \in T \quad (4)$$

$$\sum_{k\in K} q_{ikt} \le C_i - I_{i(t-1)} \ , \forall i \in N, \forall t \in T \quad (5)$$

$$q_{ikt} \leq C_i \, y_{ikt} \quad , \forall i \in N \, , \forall k \in K, \forall t \in T \quad (6)$$

$$\sum_{i \in N} q_{ikt} \leq Q_k \, y_{0kt} \quad , \forall k \in K, \forall t \in T \quad (7)$$

$$\sum_{i \in V, i<j} x_{ijkt} + \sum_{j \in V, j<i} x_{ijkt} = 2 \, y_{ikt} \, , \forall i \in V \, , \forall k \in K, \forall t \in T \quad (8)$$

$$\sum_{i \in S} \sum_{j \in S, i<j} x_{ijkt} \leq \sum_{i \in S} y_{ikt} - y_{mkt} \quad , S \subseteq N, \forall k \in K, \forall t \in T, \text{for some } m \in S \quad (9)$$

$$x_{ijkt} \leq 1 \quad , \forall \, i \in N, \forall \, j > i \in N, \forall \, k \in K, \forall t \in T \; (10)$$

$$x_{ijkt} \quad \in \{0,1,2\} \quad (11)$$

$$I_{it}, q_{ikt}, y_{ikt} \geq 0 \quad (12)$$

Η αντικειμενική συνάρτηση (1) ελαχιστοποιεί το συνολικό κόστος, το οποίο αποτελείται από το συνολικό κόστος διατήρησης αποθέματος και το συνολικό κόστος δρομολόγησης. Ο περιορισμός (2) καθορίζει το επίπεδο αποθέματος του προμηθευτή, ενώ ο περιορισμός (3) καθορίζει το επίπεδο αποθέματος του πελάτη. Ο περιορισμός (4) φροντίζει το απόθεμα να είναι θετικό και μικρότερο ή ίσο της μέγιστης χωρητικότητας του προμηθευτή ή πελάτη. Οι περιορισμοί (5) και (6) συνδέουν τις μεταβλητές q_{ikt} , y_{ikt} , I_{it} μεταξύ τους επιτρέποντας την παράδοση προϊόντων σε έναν πελάτη από ένα όχημα μόνο εάν αυτό έχει επισκεφτεί τον πελάτη. Ο περιορισμός (7) σχετίζεται με την τήρηση της μέγιστης χωρητικότητας των οχημάτων. Ο περιορισμός (8) αποτελεί degree constraint, ενώ ο περιορισμός (9) αποτρέπει την δημιουργία κλειστών διαδρομών. Τέλος ο περιορισμός (10) θέτει ως άνω όριο την μονάδα στις μεταβλητές x_{ijkt} που σχετίζονται με τα τόξα που δεν περιέχουν το 0, καθιστώντας τες ουσιαστικά δυαδικές μεταβλητές.

Ο περιορισμός (9) είναι περιορισμός τύπου callback και τα S που εισάγονται σε αυτόν είναι οι διαδρομές που προκύπτουν από την λύση του μοντέλου χωρίς τον περιορισμό. Εάν

κάποιες από τις διαδρομές αυτές παραβιάζουν τον περιορισμό, τότε γίνεται πρόσθεση του περιορισμού στο μοντέλο για τα σημεία κάθε μίας από αυτές και ακολουθεί ξανά επίλυση του. Η διαδικασία επαναλαμβάνεται έως ότου καμία από τις διαδρομές που προκύπτουν από την λύση του μοντέλου δεν παραβιάζουν τον περιορισμό (9), γεγονός που υποδηλώνει πως έχει βρεθεί η λύση του IRP.

Κεφάλαιο 3. Παρουσίαση του solver και interface της GUROBI

3.1 Σύντομη ιστορία της ανάπτυξης και χρήσης λογισμικών μαθηματικής βελτιστοποίησης.

Η αναγνώριση της αξίας των λογισμικών μαθηματικής βελτιστοποίησης (solvers) στον επιχειρηματικό τομέα και την βιομηχανία είναι σχετικά πρόσφατη. Αν και το θεωρητικό υπόβαθρο της μαθηματικής βελτιστοποίησης είχε αναπτυχθεί ήδη από τα μέσα του αιώνα από τον Leonid Kantorovich (1939) [17] και τον George Dantzig (1947) [18], η πρώτη πρακτική εφαρμογή της έγινε στις αρχές του 1960 σε ένα πλήθος προβλημάτων των βιομηχανιών πετρελαίου, οι οποίες αποτέλεσαν τους κύριους επενδυτές στην έρευνα πάνω στον υπολογιστικό γραμμικό προγραμματισμό [19]. Η ανάπτυξη της τεχνολογίας των υπολογιστών τη δεκαετία του 70 επέτρεψε την εφαρμογή της τεχνολογίας μαθηματικής βελτιστοποίησης στον ευρύτερο επιχειρηματικό κόσμο, αν και σε περιορισμένη ακόμη κλίμακα [18][19]. Σημείο καμπής, στις αρχές της δεκαετίας του 90, αποτέλεσε η έρευνα του Narendra Karmarkar[20] και άλλων πάνω σε αλγόριθμους επίλυσης μεγάλων πρακτικών προβλημάτων, επιτρέποντας εκτενέστερη χρήση εφαρμογών γραμμικού προγραμματισμού, ιδιαίτερα στις αεροπορικές εταιρείες.

Το 1987 ο Robert Bixby αναπτύσσει τον CPLEX Optimizer και γίνεται συνιδρυτής της CPLEX Optimization Inc, την οποία το 1997 αγοράζει η ILOG. Αρχικά η CPLEX ήταν μία εφαρμογή της μεθόδου simplex στην γλώσσα προγραμματισμού C και αφορούσε την επίλυση γραμμικών προβλημάτων συνδυαστικής βελτιστοποίησης. Σήμερα είναι από τα πιο γνωστά λογισμικά μαθηματικής βελτιστοποίησης, με δυνατότητες επίλυσης γραμμικών και μη προβλημάτων σε πλήθος διεπαφών και γλωσσών προγραμματισμού πέρα της C. Αναπτύσσοντας περεταίρω τον κώδικα της CPLEX, ο Bixby συνειδητοποίησε [21] την ανάγκη δημιουργίας μιας λειτουργίας, η οποία θα έδινε στους ερευνητές τη δυνατότητα εύκολου και απλού ορισμού ή αλλαγής των παραμέτρων των προβλημάτων τους καθώς και την αποτελεσματική επίλυση αυτών των προβλημάτων.

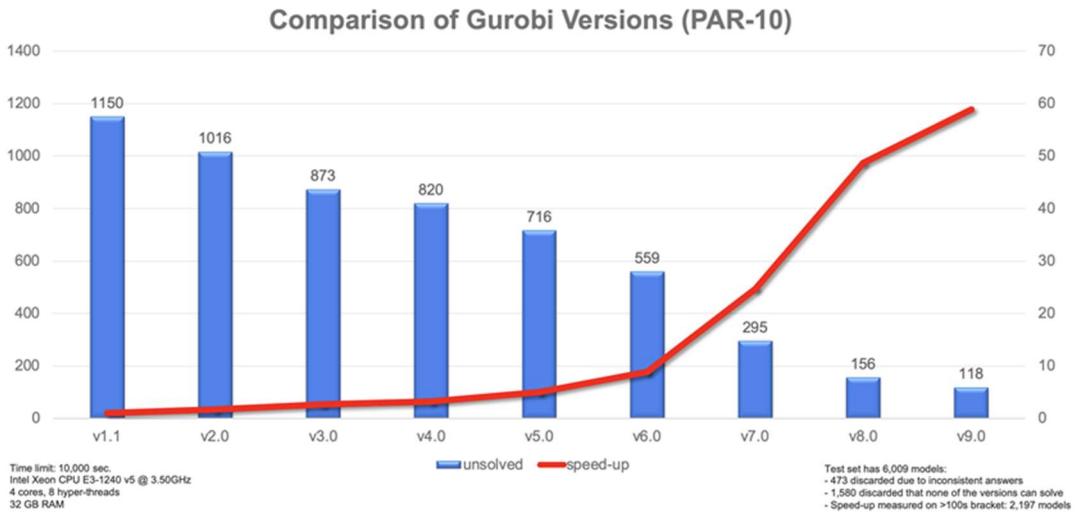
Η πραγματική εκτόξευση στην απόδοση της CPLEX έγινε το 1998 [21] όταν οι, πρόσφατα ενταγμένοι στην ομάδα της CPLEX, Ed Rothberg και Zonghao Gu εφάρμοσαν ένα πλήθος ιδεών, εμπνευσμένες από την ήδη 30ετή θεωρητική έρευνα πάνω στην μαθηματική

βελτιστοποίηση, οι οποίες δεν είχαν εφαρμοστεί ακόμα στους εμπορικούς solvers. Η έκδοση 6.5 της CPLEX, η πρώτη που περιείχε τις παραπάνω ιδέες, δεκαπλασίασε την ταχύτητα του solver και αποτέλεσε την αρχή μίας νέας γενιάς λογισμικών μεικτού ακέραιου προγραμματισμού (MIP) γενικής χρήσης. Η αύξηση της ταχύτητας και της ευελιξίας των solvers κατέστησε δυνατή πλέον τη χρήση τους σε αληθινά προβλήματα επιχειρήσεων.

Το 2008 οι Robert Bixby ,Ed Rothberg και Zonghao Gu εγκαταλείπουν την ομάδα της CPLEX και ιδρύουν τη δική τους εταιρεία λογισμικού μαθηματικής βελτιστοποίησης, την οποία ονομάζουν από τα αρχικά των επιθέτων τους GUROBI. Το όραμα των ιδρυτών της GUROBI ήταν η δημιουργία του πιο ισχυρού solver στην αγορά δίνοντας βάρος στην συνεχή βελτίωση της απόδοσης του [22]. Αν και το 2008 η GUROBI ξεκίνησε ουσιαστικά από το μηδέν, πολύ σύντομα κατέλαβε θέση μεταξύ των κορυφαίων solver της αγοράς, θέση που διατηρεί μέχρι και σήμερα.

3.2 Τα πλεονεκτήματα χρήσης του solver της Gurobi

Σήμερα στην αγορά κυκλοφορούν πολλοί ισχυροί solvers, είτε επαγγελματικοί είτε ανοιχτού λογισμικού. Είναι δύσκολο να γίνει μία συγκριτική κατάταξη των λογισμικών βελτιστοποίησης, καθώς πολλές φορές η απόδοση ενός solver ποικίλει ανάλογα τον τύπο του προβλήματος και τις προδιαγραφές ποιότητας (π.χ. μέγιστός επιτρεπτός χρόνος επίλυσης). Λαμβάνοντας υπόψιν όμως την χρησιμότητα των solvers στην επίλυση αληθινών πρακτικών προβλημάτων οι solvers που ξεχωρίζουν είναι της CPLEX, της XPRESS και της GUROBI [23].

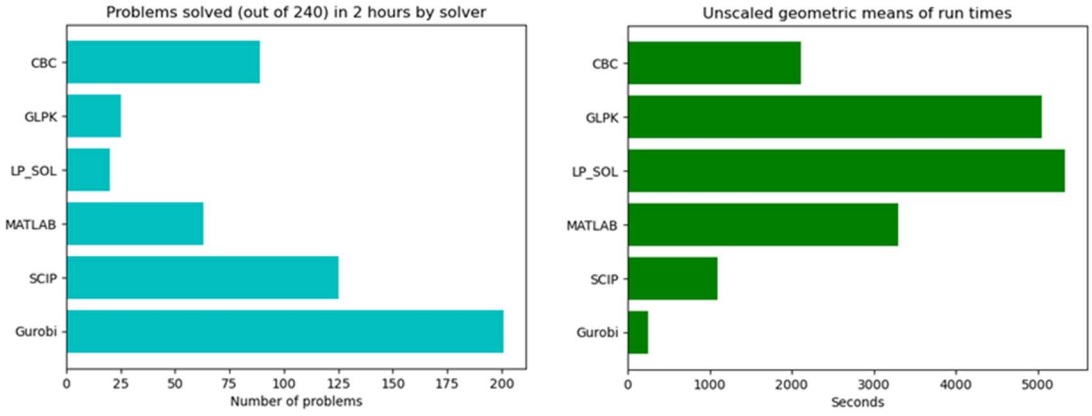


Σχήμα 3.1: Σύγκριση των εκδόσεων της Gurobi βάση του αριθμού των άλυτων προβλημάτων (αριστερός άξονας) και της ποσοστιαίας αύξησης της ταχύτητας του solver (δεξιός άξονας).

Το πιο σημαντικό χαρακτηριστικό που πρέπει να έχει ένας εμπορικός solver είναι η ικανότητα επίλυσης μεγάλων σε μέγεθος προβλημάτων σε λογικό χρονικό διάστημα. Από την ίδρυση της η GUROBI OPTIMIZATION στόχευε στην συνεχή βελτίωση της ισχύος και ταχύτητας του solver της, τις οποίες βελτιώνει με κάθε νέα έκδοση.

Η ισχύς της GUROBI επιβεβαιώνεται επίσης από τις πολλές συμμέτοχές σε δημόσιους διαγωνισμούς επιδόσεων (benchmarks) μεταξύ solvers [23][25], στους οποίους καταλαμβάνει σταθερά τις πρώτες θέσεις. Ο τελευταίος διαγωνισμός που συμμετείχαν μαζί η GUROBI, η CPLEX και η XPRESS οργανώθηκε το 2018 από τον Hans D. Mittelman [24]. Σε αυτόν τον διαγωνισμό η GUROBI αποδείχθηκε:

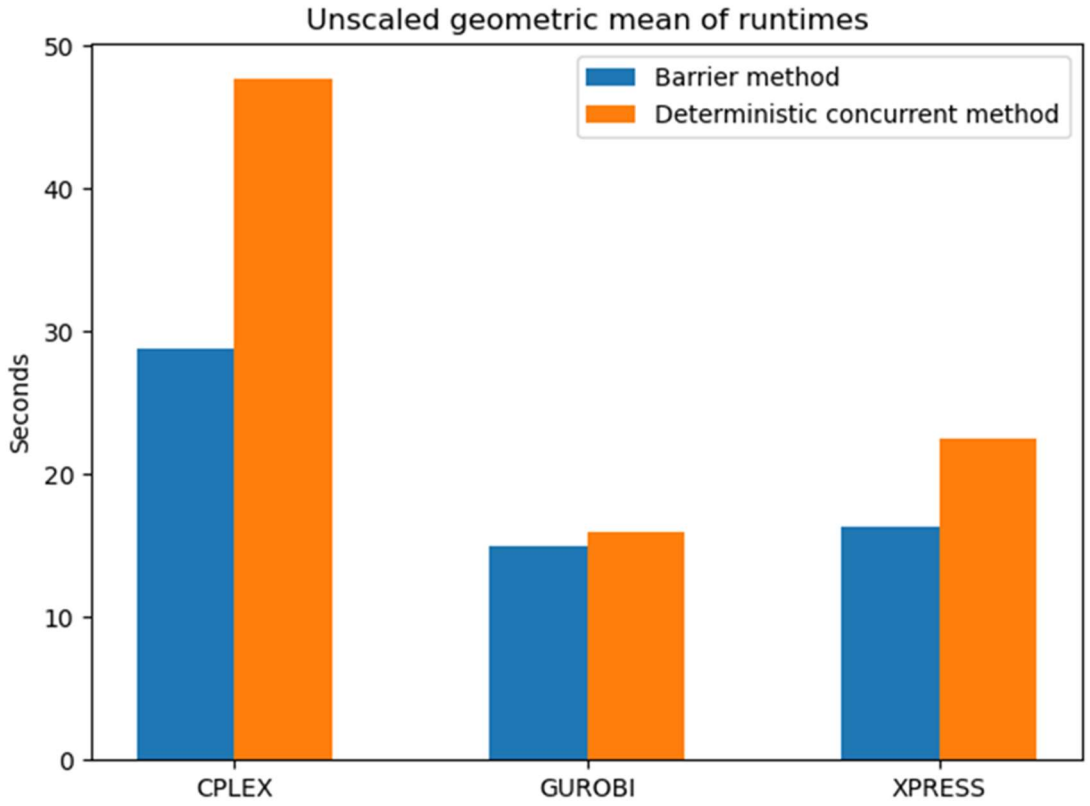
- Ο γρηγορότερος LP (Linear Programming) solver
- Ο γρηγορότερος MIP (Mixed Integer Programming) solver
- Ο γρηγορότερος QP (Quadratic Programming) και QCP (Quadratically Constrained Programming) solver
- Ο γρηγορότερος solver στην ανίχνευση άλυτων μοντέλων



Σχήμα 3.2: Αποτελέσματα benchmarks (5/2/2021) για προβλήματα γραμμικού ακεραίου προγραμματισμού. Οι solvers που συμμετάσχουν είναι οι CBC-2.10.5 (CBC), GLPK-5.0 (GPLK), LP_SOLVE-5.5.2 (LP_SOL), MATLAB-2020a (MATLAB), (F)SCIP/spx-7.0.0 (SCIP) και Gurobi-9.1.0 (GUROBI). Πηγή http://plato.asu.edu/ftp/milp.html .

Η αναφορά του Robert Bixby [26], κατά τη διάρκεια συνέντευξης το 2020, στα μεγαλύτερα MIP μοντέλα από προβλήματα πελατών, που έχει λύσει το λογισμικό, δίνει μία καλή εικόνα για την δύναμη του solver. Με βάση το πλήθος των μεταβλητών, το μεγαλύτερο μοντέλο που έχει αντιμετωπίσει ο solver περιείχε 150 εκατομμύρια μεταβλητές, 1,3 εκατομμύρια περιορισμούς και ο χρόνος επίλυσης του ήταν περίπου 13000 δευτερόλεπτα (3,6 ώρες). Με βάση το πλήθος των περιορισμών, το μεγαλύτερο μοντέλο περιείχε 57

εκατομμύρια περιορισμούς, 41 εκατομμύρια μεταβλητές και ο χρόνος επίλυσης του ήταν περίπου 331 δευτερόλεπτα (5,5 λεπτά). Αναφέρει επίσης πως τουλάχιστον 797 από όλα τα μοντέλα πελατών ξεπερνούσαν τους 1 εκατομμύρια περιορισμούς ή μεταβλητές και από αυτά τα περίπου τα 500 λύνονταν σε λιγότερο από 30000 δευτερόλεπτα (8,3 ώρες).



Σχήμα 3.3: Αποτελέσματα benchmarks (17/10/2018) για προβλήματα γραμμικού προγραμματισμού μεταξύ εμπορικών solvers. Οι solvers που συμμετάσχουν είναι οι CPLEX-12.8.0 (CPLEX), GUROBI-8.1.0 (GUROBI) και XPRESS-8.5.1 (XPRESS). Πηγή http://plato.asu.edu/ftp/lpcom.html .

Η μεγάλες ταχύτητες της GUROBI, γενικότερα των επαγγελματικών solvers, δεν πρέπει να αποδίδονται στην αύξηση τις υπολογιστικής ισχύος. Αν και η αύξηση της μνήμης και της δύναμης των επεξεργαστών βοηθούν στη μείωση των χρόνων επίλυσης, οι υψηλές επιδώσεις οφείλονται κυρίως στο πλήθος ευφυών αλγορίθμων τομής επιπέδων (cutting planes), διακλάδωσης (branching), προ επίλυσης (presolving), προ επεξεργασίας (preprocessing) και ευρετικών αλγόριθμων (heuristics) που περιλαμβάνει ο κάθε solver [27].

Ένα ακόμα πλεονέκτημα της GUROBI είναι η δυνατότητα χρήσης της σε πολλές προγραμματιστικές γλώσσες. Γλώσσες που υποστηρίζει η GUROBI είναι οι : Java, Python, C++, .Net, C, MATLAB και R. Επιτρέπει επιπλέον σύνδεση με γλώσσες μοντελοποίησης όπως η

AMPL, MPL και AIMS και με προγράμματα όπως το Excel. Παρέχει επίσης δυνατότητα για επίλυση είτε μέσω μίας συσκευής, είτε μέσω πολλαπλών συσκευών είτε μέσω σύννεφου (cloud) [28].

3.3 Η Python και το Jupyter Notebook

Η γλώσσα προγραμματισμού Python είναι διερμηνευόμενη, γενικού σκοπού, υψηλού επιπέδου και αναπτύσσεται ως ανοιχτό λογισμικό από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα από ότι θα ήταν δυνατόν σε άλλες γλώσσες και διακρίνεται τόσο για το πλήθος των βιβλιοθηκών της όσο και για την ταχύτητα εκμάθησης της.

Το Jupyter Notebook είναι ένα web-based διαδραστικό περιβάλλον δημιουργίας εγγράφων σημειώσεων και παρουσιάσεων, τα οποία είναι έγγραφα JSON διαμορφωμένης δομής, που περιέχουν μία διατεταγμένη λίστα input και output κελίων, τα οποία μπορούν να περιέχουν κώδικα, κείμενο, γραφικές παραστάσεις και rich media. Τα έγγραφα Jupyter έχουν την κατάληξη .ipynb και μπορούν εύκολα να μετατραπούν και σε άλλους τύπους εγγράφων (HTML, LaTeX, PDF, Python, διαφάνειες παρουσίασης). Το περιβάλλον του Jupyter υποστηρίζει επιπλέον πάνω από 40 από διαφορετικές γλώσσες προγραμματισμού όπως Python, R, Julia και Scala.

3.4 Εισαγωγή στη χρήση του solver της GUROBI μέσω Python και Jupyter Notebook

Όπως αναφέρθηκε, η χρήση του solver της GUROBI μπορεί να γίνει μέσω πλήθους διεπαφών σε διάφορες γλώσσες προγραμματισμού. Στα πλαίσια της τρέχουσας διπλωματικής γίνεται χρήση της διεπαφής GUROBI-Python στο περιβάλλον του Jupyter Notebook.

Η διεπαφή Python του solver της GUROBI επιδιώκει να ακολουθήσει την φιλοσοφία απλότητας και αναγνωσιμότητας της Python, επιτρέποντας την εύκολη κατασκευή και επεξεργασία μοντέλων βελτιστοποίησης, με κώδικα που παρομοιάζει εμφανισιακά το μαθηματικό μοντέλο. Η επίλυση του μοντέλου γίνεται εξίσου εύκολα, όπως και η πρόσβαση

σε πληροφορίες σχετικά με αυτό. Η βιβλιοθήκη της Gurobi ονομάζεται gurobipy και εισάγει νέα αντικείμενα και συναρτήσεις στο περιβάλλον της Python.

3.4.1 Η λογική της διεπαφής GUROBI-Python

Οι περισσότερες ενέργειες στην διεπαφή του GUROBI-Python πραγματοποιούνται με την επίκληση μεθόδων στα αντικείμενα GUROBI. Το πιο διαδεδομένο σε χρήση αντικείμενο είναι το *Model*, το οποίο αποτελείται από ένα σύνολο μεταβλητών απόφασης (αντικείμενα τύπου *Var* και *MVar*), ένα σύνολο περιορισμών (αντικείμενα τύπου *Constr*, *QConstr*, *SOS*, ή *GenConstr*) και μία γραμμική ή τετραγωνική αντικειμενική συνάρτησή (ορισμένη με την μέθοδο *Model.set Objective*). Σε κάθε μεταβλητή αντιστοιχεί ένας τύπος (continuous, integer, binary, κτλ.), ένα κάτω και άνω όριο ενώ σε κάθε περιορισμό αντιστοιχεί μία ισότητα ή ανισότητα και μία σταθερά «δεξιού μέρους». Τα models είναι δυναμικές οντότητες, δίνοντας τη δυνατότητα πρόσθεσης ή αφαίρεσης μεταβλητών και περιορισμών οποιαδήποτε στιγμή.

Ένα μοντέλο βελτιστοποίησης γίνεται να οριστεί είτε φορτώνοντας το απευθείας ολόκληρο από ένα αρχείο (χρησιμοποιώντας την συνάρτηση *read()*), είτε κατασκευάζοντας το σταδιακά, δημιουργώντας αρχικά ένα κενό αντικείμενο τύπου *Model* και στη συνέχεια χτίζοντάς το με την κλήση μεθόδων. Κάποιες από τις βασικές μεθόδους κατασκευής είναι οι:

- *Model.addVar*, *Model.addVars*, με τις οποίες γίνεται η προσθήκη μεταβλητών απόφασης.
- *Model.addConstr*, *Model.addConstrs*, *Model.addSOS*, με τις οποίες γίνεται η εισαγωγή περιορισμών. Ο ορισμός των γραμμικών περιορισμών πραγματοποιείται με την κατασκευή γραμμικών εκφράσεων (αντικείμενα τύπου *LinExpr*) και στην συνέχεια προσδιορίζοντας τις σχέσεις μεταξύ των παραπάνω εκφράσεων.

Αφού έχει πραγματοποιηθεί η κατασκευή ενός μοντέλου, η λύση του γίνεται μέσω της μεθόδου *Model.optimize*. Ο solver της GUROBI παρακολουθεί στενά την κατάσταση του μοντέλου, συνεπώς η επίκληση της παραπάνω μεθόδου θα πραγματοποιηθεί μόνο εφόσον έχει παρέλθει αλλαγή των σχετιζόμενων δεδομένων από την τελευταία βελτιστοποίηση του μοντέλου. Η διαγραφή των πληροφοριών μίας υπολογισμένης λύσης και επανεκκίνηση της βελτιστοποίησης από το μηδέν χωρίς την αλλαγή του μοντέλου γίνεται μέσω της μεθόδου *Model.reset*.

Ιδιαίτερης προσοχής χρήζει, το γεγονός ότι οι αλλαγές στο μοντέλο αντιμετωπίζονται «τεμπέλικα» (lazy updates), δηλαδή δεν επηρεάζουν άμεσα το μοντέλο αλλά δρομολογούνται ώστε να εφαρμοσθούν αργότερα όταν γίνει ενημέρωση του μοντέλου. Η ενημέρωση του μοντέλου μπορεί να γίνεται με τις μεθόδους *Model.update*, *Model.optimize* και *Model.write*.

Ο κύριος μηχανισμός εύρεσης ή αλλαγής χαρακτηριστικών σε ένα μοντέλο γίνεται μέσω της διεπαφής *attribute*. Ορισμένα από αυτά τα χαρακτηριστικά παίρνουν τιμές μόνο συγκεκριμένες στιγμές (π.χ. αυτά που συνδέονται με την βέλτιστη λύση του μοντέλου), ενώ άλλα είναι διαθέσιμα συνεχώς (π.χ. ο αριθμός των μεταβλητών του μοντέλου). Τα χαρακτηριστικά μπορεί να σχετίζονται με μεταβλητές, περιορισμούς ή ακόμα και ολόκληρο το μοντέλο. Η εκχώρηση χαρακτηριστικών σε μεταβλητές γίνεται μέσω της μεθόδου *model.getAttr* ή να τις προβάλω απευθείας μέσω της μεθόδου *model.printAttr*. Από τα πιο χρήσιμα χαρακτηριστικά, είναι το χαρακτηριστικό *X* το οποίο περιέχει τις τιμές των μη μηδενικών μεταβλητών στη λύση και το χαρακτηριστικό *ObjVal* το οποίο περιέχει την τιμή της αντικειμενικής στη λύση.

Όλα τα παραπάνω μπορούν να γίνουν κατανοητά με ένα απλό παράδειγμα. Έστω ότι είναι επιθυμητή η δημιουργία και επίλυση του παρακάτω μοντέλου:

$$maximize\ X_1 + 3X_2$$

Υπό τους περιορισμούς:

$$X_1 + 2X_2 \leq 200 \quad (1)$$
$$X_1 + 5X_2 \leq 300 \quad (2)$$
$$X_1, X_2 > 0\ and\ integer\ (3)$$

Όπως σε κάθε κώδικα Python αρχικά εισάγονται οι απαιτούμενες βιβλιοθήκες. Για την κατασκευή και επίλυση μοντέλου μέσω της GUROBI εισάγω την αντίστοιχη βιβλιοθήκη:

```
from gurobipy import *
```

Το επόμενο βήμα είναι η δημιουργία μίας κλάσης τύπου μοντέλου Gurobi.

```
m=Model()
```

Στη συνέχεια γίνεται η προσθήκη των μεταβλητών απόφασης.

```
x1=m.addVar(vtype=GRB.INTEGER,name="x1")
x2=m.addVar(vtype=GRB.INTEGER,name="x2")
```

Ακολουθεί η προσθήκη των περιορισμών.

```
C1=m.addConstr(x1+2*x2<=200,name="Condition1")
C2=m.addConstr(x1+5*x2<=300,name="Condotion2")
```

Τελευταίο βήμα της κατασκευής είναι ο καθορισμός της αντικειμενικής συνάρτησης.

```
Ob=m.setObjective(x1+3*x2,GRB.MAXIMIZE)
```

Τελικώς ακολουθεί η εύρεση της βέλτιστης λύσης του μοντέλου. Μαζί με την επίλυση γίνεται αυτόματα και η ενημέρωση του μοντέλου με τις μεταβλητές, περιορισμούς και αντικειμενική που δόθηκαν.

```
m.optimize()
```

Αφού ολοκληρωθεί η διαδικασία της επίλυσης ανακτάται η βέλτιστη τιμή της αντικειμενικής και η τιμή των μεταβλητών στην λύση με τη χρήση της μεθόδου *printAttr*.

```
m.printAttr('ObjVal')
```

```
    ObjVal
------------
       233
```

```
m.printAttr('X')
```

```
    Variable            X
-------------------------
          x1          134
          x2           33
```

3.4.2 Ενσωματωμένες δομές δεδομένων της Python

Πριν την περεταίρω εμβάθυνση στα αντικείμενα της gurobipy και των μεθόδων τους, πρέπει να γίνει μία σύντομη αναφορά στις ενσωματωμένες δομές δεδομένων της Python

καθώς χρησιμοποιούνται εκτενώς για την εισαγωγή και εξαγωγή δεδομένων στο μοντέλο. Οι δομές αυτές είναι:

- Οι πλειάδες (tuples)
- Οι λίστες (lists)
- Τα λεξικά (dicts)

Κατά τη μοντελοποίηση με GUROBI οι πλειάδες είναι ιδανικές για την αναπαράσταση πολυδιάστατων δεικτών. Κύριο χαρακτηριστικό τους είναι ότι από τη στιγμή της δημιουργίας τους δεν μπορούν να μεταβληθούν. Η δημιουργία μιας πλειάδας καθώς και η επίκληση σε ένα στοιχείο της γίνεται ώς εξής:

```
destinations=('Volos','Larisa','Trikala')
```

```
destinations[0]
```

```
Out[10]:
'Volos'
```

Σε αντίθεση με τις πλειάδες, οι λίστες γίνεται να μεταβληθούν κατά τη διάρκεια του κώδικα. Κατά την μοντελοποίηση οι λίστες ενδείκνυνται για την αναπαράσταση συνόλων. Η δημιουργία μιας πλειάδας καθώς και η επίκληση σε ένα στοιχείο της γίνεται ώς εξής:

```
customers=[1,2,3]
```

```
customers[0]
```

```
Out[12]:
1
```

Τα λεξικά αντιστοιχίζουν τιμές σε κλειδιά, τα οποία είναι αριθμοί (floats, integers) ή λέξεις (strings). Στην μοντελοποίηση τα λεξικά είναι πολύ χρήσιμα για την αναπαράσταση δεδομένων με δείκτες (π.χ. κόστη). Η δημιουργία λεξικού καθώς και η επίκληση σε ένα στοιχείο του γίνεται ώς εξής:

```
travel_costs={'Volos':100,'Larisa':200,'Trikala':150}
```

```
travel_costs['Volos']
```

```
Out[15]:
100
```

3.4.3 Δημιουργία νέου μοντέλου

Η κλήση της εντολής *Model()* δημιουργεί ένα κενό αντικείμενο τύπου *Model*, το οποίο για την ευκολότερη επεξεργασία του συνηθίζεται η εκχώρηση του σε μία καθολική μεταβλητή. Η συνάρτηση συντάσσεται ώς :

$$Model(\, name = "\ ", env = desfaultEnv\,)$$

- $name$: Το όνομα του μοντέλου. Το όνομα θα αποθηκευτεί σαν ASCII string και γι' αυτό καλό είναι το όνομα να αποτελείται μόνο από χαρακτήρες ASCII. Επίσης συνίσταται τα κενά να αποφεύγονται καθώς υπάρχει αδυναμία γραφής τους σε lp αρχεία.
- env : Το περιβάλλον στο οποίο δημιουργείται το μοντέλο. Εξ' ορισμου επιλέγεται το περιβάλλον της Gurobi.

Παράδειγμα δημιουργίας και επίκλησης αντικειμένων *Model.*

```
sm=Model("Serious_Model")
cm=Model("Casual_Model")
```

```
sm
```

```
Out[17]:
```

```
<gurobi.Model Continuous instance Serious_Model: 0 constrs, 0 vars, No param
```

3.4.4 Πρόσθεση μεταβλητών απόφασης στο μοντέλο

Αφού γίνει η δημιουργία ενός κενού μοντέλου μέσω της Model() , το επόμενο βήμα στην κατασκευή του μοντέλου είναι η εισαγωγή των μεταβλητών απόφασης, καθώς προφανώς χωρίς αυτές δεν γίνεται να προστεθούν περιορισμοί ούτε να ορισθεί αντικειμενική. Οι μεταβλητές εκφράζονται μέσω των αντικειμένων τύπου Var και MVar που εισάγει η gurobipy.

3.4.4.1 Εισαγωγή μίας μεταβλητής απόφασης

Η πρόσθεση μίας μεταβλητής σε ένα μοντέλο γίνεται με την μέθοδο *Model.addVar()*, η οποία συντάσσεται ώς:

$$addVar(lb = 0.0, ub = GRB.INFINITYY, obj = 0.0, vtype = GRB.CONTINUOUS,$$
$$name = \text{" "}, column = None)$$

- lb (προαιρετικό) : Το κάτω όριο της νέας μεταβλητής. Η εξ' ορισμού τιμή είναι μηδέν.
- ub (προαιρετικό) : Το άνω όριο τής νέας μεταβλητής. Η εξ' ορισμού τιμή είναι άπειρο (GRB.INFINITY).
- obj (προαιρετικό) : Ο συντελεστής της μεταβλητής στην αντικειμενική. Εξ' ορισμού ίσος με 0, που σημαίνει πως η αντικειμενική συνάρτηση θα πρέπει να ορισθεί μέσω της μεθόδου *model.setObjective()*. Εάν το obj πάρει μη μηδενική τιμή ο συντελεστής και η μεταβλητή προστίθενται αυτόματα στην αντικειμενική και δεν υπάρχει ανάγκη ορισμού της. Η μόνη επιπλέον ενέργεια που απαιτείται είναι να προσδιοριστεί με την χρήση της μεθόδου *model.setAttr("ModelSense", -1)* ή *model.setAttr("ModelSense", 1)* πως επιθυμητή είναι αντίστοιχα η μεγιστοποίηση ή ελαχιστοποίηση της αντικειμενικής συνάρτησης. Οποιαδήποτε αναφορά στην μέθοδο *model.setObjective()* κάνει όλες τις μεταβλητές να αντιμετωπίζονται σαν να έχουν obj=0.0.
- $vtype$ (προαιρετικό) : Τύπος της μεταβλητής (GRB.CONTINUOUS, GRB.BINARY, GRB.INTEGER, GRB.SEMICONT, GRB.SEMIINT). Εξορισμού η μεταβλητή είναι τύπου GRB.CONTINUOUS.
- $name$ (προαιρετικό) : Το όνομα της μεταβλητής σε χαρακτήρες ASCII. Η χρήση κενών να αποφεύγεται λόγω της αδυναμίας αποθήκευσης σε lp αρχεία.
- $column$ (προαιρετικό) : Αντικείμενο στήλης που υποδηλώνει το σύνολο των περιορισμών στο οποίο συμμετέχει ή νέα μεταβλητή και οι σχετιζόμενοι συντελεστές. Εξ' ορισμού δεν παίρνει κάποια τιμή.

Παράδειγμα δημιουργίας και επίκλησης μίας ακέραιας μεταβλητής με όνομα *SV*.

```
sv=cm.addVar(vtype=GRB.INTEGER,name="SV")
```

```
cm.update()
sv
```

```
Out[20]:

<gurobi.Var SV>
```

3.4.4.2 Εισαγωγή πολλαπλών μεταβλητών απόφασης

Τα μοντέλα βελτιστοποίησης πρακτικών εφαρμογών περιέχουν συνήθως μεγάλο πλήθος μεταβλητών, των οποίων η εισαγωγή κάθε μίας ξεχωριστά είναι αν όχι αδύνατη σίγουρα χρονοβόρα και καθόλου αποδοτική. Η ανάγκη ταυτόχρονης εισαγωγής πολλών μεταβλητών καλύπτεται μέσω της μεθόδου *Model.addVars()*. Η μέθοδος συντάσσεται ως :

$$addVars(indices, lb = 0.0, ub = GRB.INFINITYY, obj = 0.0,$$
$$vtype = GRB.CONTINUOUS, name = \text{" "})$$

- lb (προαιρετικό) : Το κάτω όριο των νέων μεταβλητών. Η εξ' ορισμού τιμή είναι μηδέν.
- ub (προαιρετικό) : Το άνω όριο των νέων μεταβλητών. Η εξ' ορισμού τιμή είναι άπειρο (GRB.INFINITY).
- obj (προαιρετικό) : Ο συντελεστής των νέων μεταβλητών στην αντικειμενική. Εξ' ορισμού ίσος με 0, που σημαίνει πως η αντικειμενική συνάρτηση θα πρέπει να κατασκευαστεί μέσω της *model.setObjective*(). Εάν το obj πάρει μη μηδενική τιμή οι συντελεστές και οι μεταβλητές προστίθενται αυτόματα στην αντικειμενική και δεν υπάρχει ανάγκη για δημιουργία της. Η μόνη ενέργεια που ίσως χρειαστεί είναι με την *model.setAttr("ModelSense", -1)* να δηλωθεί πως επιθυμητή είναι η μεγιστοποίηση της αντικειμενικής ή με την με την *model.setAttr("ModelSense", 1)* η ελαχιστοποίηση της αντικειμενικής. Οποιαδήποτε αναφορά στην μέθοδο *model.setObjective()* κάνει όλες τις μεταβλητές να αντιμετωπίζονται σαν να έχουν obj=0.0.
- $vtype$ (προαιρετικό) : Τύπος των νέων μεταβλητών (GRB.CONTINUOUS, GRB.BINARY, GRB.INTEGER, GRB.SEMICONT, GRB.SEMIINT). Εξορισμού οι μεταβλητές είναι τύπου GRB.CONTINUOUS.

- $name$ (προαιρετικό) : Το όνομα των νέων μεταβλητών σε χαρακτήρες ASCII. Η χρήση κενών να αποφεύγεται λόγω της αδυναμίας αποθήκευσης σε lp αρχεία. Το τελικό όνομα της μεταβλητής θα περιέχει και τους αντίστοιχους δείκτες. Εάν δεν δοθεί κάποιο συγκεκριμένο όνομα οι μεταβλητές θα πάρουν αυτόματα το όνομα Cx, όπου x η θέση της μεταβλητής στη λίστα των συνολικών μεταβλητών.

Το πιο σημαντικό στοιχείο εισαγωγής είναι το indices, το οποίο μπορεί να συνταχθεί με διάφορους τρόπους επιτρέποντας έτσι πέντε εναλλακτικούς τρόπους εισαγωγής των μεταβλητών. Η επιλογή της κατάλληλης σύνταξης εξαρτάται από το επιθυμητό αποτέλεσμα το οποίο συνήθως έχει να κάνει με τους δείκτες και τις διαστάσεις των μεταβλητών.

Ο πρώτος τρόπος εισαγωγής των δεικτών είναι βάζοντας στη θέση της παραμέτρου εισαγωγής indices απλά τις τιμές του μεγέθους των διαστάσεών τους. Έστω απαιτείται η εισαγωγή δυαδικών μεταβλητών $aprt_{i,j,k}$ όπου $i,k \in (0,1)$ και $j \in (0,1,2)$.

```
aprt=sm.addVars(2,3,2,vtype=GRB.BINARY,name='aprt')
sm.update()
aprt
```

```
Out[21]:

{(0, 0, 0): <gurobi.Var aprt[0,0,0]>,
 (0, 0, 1): <gurobi.Var aprt[0,0,1]>,
 (0, 1, 0): <gurobi.Var aprt[0,1,0]>,
 (0, 1, 1): <gurobi.Var aprt[0,1,1]>,
 (0, 2, 0): <gurobi.Var aprt[0,2,0]>,
 (0, 2, 1): <gurobi.Var aprt[0,2,1]>,
 (1, 0, 0): <gurobi.Var aprt[1,0,0]>,
 (1, 0, 1): <gurobi.Var aprt[1,0,1]>,
 (1, 1, 0): <gurobi.Var aprt[1,1,0]>,
 (1, 1, 1): <gurobi.Var aprt[1,1,1]>,
 (1, 2, 0): <gurobi.Var aprt[1,2,0]>,
 (1, 2, 1): <gurobi.Var aprt[1,2,1]>}
```

Όπως φαίνεται το πλεονέκτημα της μεθόδου αυτής είναι η απλότητα της καθώς απαιτεί μόνο εισαγωγή των διαστάσεων. Το μειονέκτημα της είναι πως η ονομασία των μεταβλητών γίνεται με βάση την αρίθμηση της Python, η οποία ξεκινάει από το μηδέν. Εάν η ονομασία των μεταβλητών και η σωστή εμφάνιση των δεικτών είναι σημαντική καλό είναι να χρησιμοποιηθεί κάποιος από του άλλους τρόπους εισαγωγής. Η πρόσβαση σε μία συγκεκριμένη μεταβλητή $aprt_{i,j,k}$ γίνεται ως:

```
aprt[(0,1,1)]
```

```
Out[22]:
```

```
<gurobi.Var aprt[0,1,1]>
```

```
aprt[0,1,1]
```

```
Out[23]:
```

```
<gurobi.Var aprt[0,1,1]>
```

Στον δεύτερο τρόπο εισαγωγής δεικτών στην θέση της παραμέτρου εισόδου indices εισάγονται λίστες όπου περιέχουν ή integer ή float ή string στοιχεία. Οι δείκτες προκύπτουν από το διανυσματικό γινόμενο των λιστών αυτών. Έστω είναι επιθυμητή εισαγωγής ακέραιων μεταβλητών FS, η κάθε μία από τις οποίες εκφράζει τον αριθμό ενός τύπου καταστημάτων μίας αλυσίδας σε κάποιες περιοχές :

```
regions=['Region_A','Region_B']
types=['Fast_Food','Restaurant','Pizzeria']
fs=cm.addVars(regions,types,vtype=GRB.INTEGER,name='FS')
cm.update()
fs
```

```
Out[24]:
```

```
{('Region_A', 'Fast_Food'): <gurobi.Var FS[Region_A,Fast_Food]>,
 ('Region_A', 'Restaurant'): <gurobi.Var FS[Region_A,Restaurant]>,
 ('Region_A', 'Pizzeria'): <gurobi.Var FS[Region_A,Pizzeria]>,
 ('Region_B', 'Fast_Food'): <gurobi.Var FS[Region_B,Fast_Food]>,
 ('Region_B', 'Restaurant'): <gurobi.Var FS[Region_B,Restaurant]>,
 ('Region_B', 'Pizzeria'): <gurobi.Var FS[Region_B,Pizzeria]>}
```

Η παραπάνω μέθοδος είναι ιδανική για την κατασκευή ενός πλήθους μεταβλητών των οποίων οι δείκτες αποτελούν όλους τους δυνατούς συνδυασμούς ενός αριθμού συνόλων. Η πρόσβαση σε μία συγκεκριμένη μεταβλητή γίνεται ως:

```
fs['Region_A','Pizzeria']
```

```
Out[26]:
```

```
<gurobi.Var FS[Region_A,Pizzeria]>
```

Ένας άλλος τρόπος εισαγωγής δεικτών γίνεται εάν στην παράμετρο εισόδου indices εισαχθεί μια λίστα από πλειάδες δεικτών ιδίου μεγέθους. Οι πλειάδες πρέπει να περιέχουν είτε τιμές (integer, float) είτε κείμενο και δεν μπορούν να περιέχουν αντικείμενα όπως άλλες πλειάδες, λίστες ή λεξικά. Δημιουργούνται τόσες μεταβλητές όσες οι εισαγόμενες πλειάδες με δείκτες το περιεχόμενο τους. Έστω απαιτείται εισαγωγή δυαδικών μεταβλητών

$RT_{pointA,pointB}$, η κάθε μία από τις οποίες εκφράζει το εάν θα πραγματοποιηθεί ένα δρομολόγιο μεταξύ δύο σημείων :

```
possible_routes=[('port','city_center'),('port','airport')
                 ,('city_center','port'),('city_center','airport')]
rt=cm.addVars(possible_routes,vtype=GRB.BINARY,name='RT')
cm.update()
rt
```

```
Out[27]:

{('port', 'city_center'): <gurobi.Var RT[port,city_center]>,
 ('port', 'airport'): <gurobi.Var RT[port,airport]>,
 ('city_center', 'port'): <gurobi.Var RT[city_center,port]>,
 ('city_center', 'airport'): <gurobi.Var RT[city_center,airport]>}
```

Το μεγάλο πλεονέκτημα της παραπάνω μεθόδου είναι ότι τα ονόματα των δεικτών μπορούν να καθοριστούν πλήρως από το χρήστη. Η πρόσβαση σε μία συγκεκριμένη μεταβλητή γίνεται ως:

```
rt['port','airport']
```

```
Out[29]:

<gurobi.Var RT[port,airport]>
```

Δεν είναι λίγες οι φορές που τα δεδομένα έχουν μορφή λεξικών με τα κλειδιά να αποτελούν τους δείκτες των μεταβλητών απόφασης. Ο τέταρτος τρόπος εισαγωγής δεικτών επιτρέπει πραγματοποιείται εισάγοντας στη θέση της παραμέτρου εισόδου indices ένα λεξικό Python. Έστω χρειάζεται γίνει ορισμός ακέραιων μεταβλητών $PA_{product}$, η κάθε μία από τις οποίες εκφράζει την ποσότητα παραγωγής ενός τύπου προϊόντος :

```
production_cost={"Product1":2000,"Product2":1800,"Product3":1900}
pa=cm.addVars(production_cost,vtype=GRB.BINARY,name="PA")
cm.update()
pa
```

```
Out[30]:

{'Product1': <gurobi.Var PA[Product1]>,
 'Product2': <gurobi.Var PA[Product2]>,
 'Product3': <gurobi.Var PA[Product3]>}
```

Πρόσβαση σε μία συγκεκριμένη τιμή γίνεται με την εκτέλεση:

```
pa["Product1"]
```

```
Out[31]:

<gurobi.Var PA[Product1]>
```

Τελικώς δημιουργία δεικτών γίνεται και εισάγοντας στη θέση της παραμέτρου εισόδου indices μία λίστα με τους δείκτες. Έστω απαιτείται εισαγωγή ακέραιων μεταβλητών $BC_{product\ barcode}$, η κάθε μία από τις οποίες εκφράζει την ποσότητα παραγωγής ενός τύπου προϊόντος βάση του barcode του προϊόντος:

```
barcodes=[123,156,122,190,145,157,110]
bc=cm.addVars(barcodes,vtype=GRB.INTEGER,name="BC")
cm.update()
bc
```

```
Out[35]:

{123: <gurobi.Var BC[123]>,
 156: <gurobi.Var BC[156]>,
 122: <gurobi.Var BC[122]>,
 190: <gurobi.Var BC[190]>,
 145: <gurobi.Var BC[145]>,
 157: <gurobi.Var BC[157]>,
 110: <gurobi.Var BC[110]>}
```

Η μέθοδος αυτή είναι ιδανική για μεταβλητές μίας διάστασης. Σαν δείκτες μπορούν να χρησιμοποιηθούν και λέξεις. Η πρόσβαση σε μία συγκεκριμένη μεταβλητή γίνεται ως :

```
bc[190]
```

```
Out[36]:

<gurobi.Var BC[190]>
```

3.4.5 Πρόσθεση περιορισμών στο μοντέλο

Αφού γίνει η εισαγωγή των μεταβλητών απόφασης, το επόμενο βήμα συνήθως είναι η κατασκευή των περιορισμών. Οι περιορισμοί εκφράζονται μέσω των αντικειμένων τύπου *Constr*, *QConstr*, *SOS* και *GenConstr* που εισάγει ή gurobipy.

3.4.5.1 Κατασκευή εκφράσεων μεταβλητών

Οι περιορισμοί αποτελούνται από σχέσεις ισότητας ή ανισότητας μεταξύ εκφράσεων μεταβλητών, οι οποίες μπορεί να είναι είτε γραμμικές $(\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n)$ είτε τετραγωνικές οι οποίες περιέχουν τετραγωνικούς όρους $(x_i^2$ ή $x_n x_m)$. Η GUROBI αντιμετωπίζει τις εκφράσεις μεταβλητών ως ξεχωριστά αντικείμενα, με τα αντικείμενα LinExpr και QuadExpr να εκφράζουν τις γραμμικές και τετραγωνικές εκφράσεις αντίστοιχα. Η κατασκευή εκφράσεων μεταβλητών (χρησιμοποιηθούν οι μεταβλητές $BC_{product\ barcode}$ της ενότητας *3.4.4.2*) γίνεται ως εξής:

Κατασκευή γραμμικής έκφρασης.

```
lin_expr=bc[123]+4*bc[145]+7*bc[157]
lin_expr
```

```
Out[40]:

<gurobi.LinExpr: BC[123] + 4.0 BC[145] + 7.0 BC[157]>
```

Κατασκευή τετραγωνικής έκφρασης.

```
quad_expr=bc[123]*bc[123]+3*bc[145]*bc[157]+10*bc[145]
quad_expr
```

```
Out[41]:

<gurobi.QuadExpr: 10.0 BC[145] + [ BC[123] ^ 2 + 3.0 BC[145] * BC[157] ]>
```

Στα προβλήματα μεικτού ακέραιου προγραμματισμού (MIP) οι περιορισμοί και αντικειμενική συνάρτηση αποτελούνται από γραμμικές εκφράσεις μεταβλητών. Οι εκφράσεις αυτές γίνεται να ορισθούν απευθείας, όπως στο παραπάνω παράδειγμα, όμως στα περισσότερα πρακτικά προβλήματα ο μεγάλος αριθμός των μεταβλητών προτάσσει χρήση αποδοτικότερων μεθόδων σύνταξης γραμμικών εκφράσεων.

Ο πρώτος τρόπος είναι αξιοποιώντας την συνάρτηση *quicksum* την οποία εισάγει η *gurobipy*. Η συνάρτηση *quicksum* είναι ουσιαστικά μία εκδοχή της συνάρτησης *sum* της Python ή οποία είναι πολύ πιο αποδοτική στην κατασκευή μεγάλων εκφράσεων GUROBI (αντικείμενα τύπου LinExpr και QuadExpr). Η συνάρτηση συντάσσεται ώς:

$$quicksum(data)$$

- $data$: Μία λίστα αντικείμενων τύπου *Var* πολλαπλασιασμένα με τους συντελεστές τους (αν απουσιάζουν θεωρούνται μονάδα).

Έστω απαιτείται η δημιουργία της παρακάτω γραμμικής έκφρασης η οποία αποτελείται από τις μεταβλητές $aprt_{i,j,k}$ της ενότητας 3.4.4.2 και κάποιων συντελεστών κέρδους.

$$\sum_i \sum_j \sum_k profits_{i,j,k} aprt_{i,j,k}$$

```
profits={(0, 0, 0):2, (0, 0, 1):4, (0, 1, 0):3, (0, 1, 1):2, (0, 2, 0):5, (0, 2, 1):2,
         (1, 0, 0):3, (1, 0, 1):1, (1, 1, 0):4, (1, 1, 1):4, (1, 2, 0):2, (1, 2, 1):4}
```

```
index_i=[0,1]
index_j=[0,1,2]
index_k=[0,1]
```

```
prof_x_aprt=[profits[i,j,k]*aprt[i,j,k] for i in index_i for j in index_j for k in index_k]
prof_x_aprt
```

```
Out[45]:

[<gurobi.LinExpr: 2.0 aprt[0,0,0]>,
 <gurobi.LinExpr: 4.0 aprt[0,0,1]>,
 <gurobi.LinExpr: 3.0 aprt[0,1,0]>,
 <gurobi.LinExpr: 2.0 aprt[0,1,1]>,
 <gurobi.LinExpr: 5.0 aprt[0,2,0]>,
 <gurobi.LinExpr: 2.0 aprt[0,2,1]>,
 <gurobi.LinExpr: 3.0 aprt[1,0,0]>,
 <gurobi.LinExpr: aprt[1,0,1]>,
 <gurobi.LinExpr: 4.0 aprt[1,1,0]>,
 <gurobi.LinExpr: 4.0 aprt[1,1,1]>,
 <gurobi.LinExpr: 2.0 aprt[1,2,0]>,
 <gurobi.LinExpr: 4.0 aprt[1,2,1]>]
```

```
lin_aprt_expr_2=quicksum(prof_x_aprt)
lin_aprt_expr_2
```

```
Out[46]:

<gurobi.LinExpr: 2.0 aprt[0,0,0] + 4.0 aprt[0,0,1] + 3.0 aprt[0,1,0] + 2.0 a
prt[0,1,1] + 5.0 aprt[0,2,0] + 2.0 aprt[0,2,1] + 3.0 aprt[1,0,0] + aprt[1,0,
1] + 4.0 aprt[1,1,0] + 4.0 aprt[1,1,1] + 2.0 aprt[1,2,0] + 4.0 aprt[1,2,1]>
```

Αν και ή *quicksum* είναι αποδοτικότερη απ' ότι η *sum*, δεν είναι ο γρηγορότερος τρόπος κατασκευής μεγάλων εκφράσεων. Για γρηγορότερο χρόνο κατασκευής συνιστάται η χρήση της *LinExpr()* η οποία συντάσσεται ώς:

$$LinExpr(argument1, argument2)$$

- $argument1$: Σε αυτή την παράμετρο εισόδου μπορεί να εισαχθεί
  - Σταθερά : Σε αυτήν την περίπτωση στην παράμετρο εισόδου $argument2$ δεν εισάγεται τίποτα.
  - Μεταβλητή : Σε αυτήν την περίπτωση στην παράμετρο εισόδου $argument2$ δεν εισάγεται τίποτα.
  - Γραμμική έκφραση : Σε αυτήν την περίπτωση στην παράμετρο εισόδου $argument2$ δεν εισάγεται τίποτα.

- Λίστα συντελεστών : Σε αυτήν την περίπτωση στην παράμετρο εισόδου $argument2$ πρέπει να εισαχθεί λίστα μεταβλητών ίσου μεγέθους.
- Λίστα πλειάδων συντελεστή-μεταβλητής. Σε αυτήν την περίπτωση στην παράμετρο εισόδου $argument2$ δεν εισάγεται τίποτα.

- $argument2$:Εξαρτάται απο τι εισάγεται στην παράμετρο εισόδου $argument1$.

Η ίδια έκφραση με πριν (άθροισμα $profits_{i,j,k} * aprt_{i,j,k}$) δημιουργείται ως:

```
list_profits=[profits[i,j,k] for (i,j,k) in list(profits)]
aprt_list=[aprt[i,j,k] for (i,j,k) in list(aprt)]
lin_aprt_expr_4=LinExpr(list_profits,aprt_list)
lin_aprt_expr_4
```

```
Out[47]:

<gurobi.LinExpr: 2.0 aprt[0,0,0] + 4.0 aprt[0,0,1] + 3.0 aprt[0,1,0] + 2.0 a
prt[0,1,1] + 5.0 aprt[0,2,0] + 2.0 aprt[0,2,1] + 3.0 aprt[1,0,0] + aprt[1,0,
1] + 4.0 aprt[1,1,0] + 4.0 aprt[1,1,1] + 2.0 aprt[1,2,0] + 4.0 aprt[1,2,1]>
```

3.4.5.2 Εισαγωγή ενός περιορισμού

Από την στιγμή που έχουν δημιουργηθεί οι απαραίτητες γραμμικές εκφράσεις η εισαγωγή ενός περιορισμού γίνεται μέσω της μέθοδού *model.addConsr(),* η οποία συντάσσεται ώς:

$$addConstr(lhs, sense = None, rhs = None, name = \text{" "})$$

- lhs : Το αριστερό μέρος του νέου περιορισμού. Μπορεί να είναι μία σταθερά, μία μεταβλητή, μία γραμμική έκφραση ή μία τετραγωνική έκφραση.
- $sense$: Η σχέση μετάξυ του αριστερού και δεξιού μέρους του περιορισμού. Πιθανές εκφράσεις που μπορεί να πάρει είναι GRB.LESS_EQUAL (γιά <=), GRB.GREATER_EQUAL (για >=) και GRB.EQUAL (για =).
- rhs : Το δεξιό μέρος του νέου περιορισμού. Μπορεί να είναι μία σταθερά, μία μεταβλητή, μία γραμμική έκφραση ή μία τετραγωνική έκφραση.
- $name$: Το όνομα του νέου περιορισμού σε χαρακτήρες ASCII. Η χρήση κενών να αποφεύγεται λόγω της αδυναμίας αποθήκευσης σε lp αρχεία.

Εναλλακτικά συντάσσεται και ως:

$$addConstr(\gamma\rho\alpha\mu\mu\iota\kappa\acute{\eta}\ \acute{\epsilon}\kappa\phi\rho\alpha\sigma\eta\ (>=\ \acute{\eta}\ <=\ \acute{\eta}\ ==)\ \gamma\rho\alpha\mu\mu\iota\kappa\acute{\eta}\ \acute{\epsilon}\kappa\phi\rho\alpha\sigma\eta\ , name = \text{" "})$$

Έστω απαιτείται η δημιουργία του παρακάτω περιορισμού, ο οποίος βασίζεται στην γραμμική έκφρασης της ενότητας 3.4.5.1:

$$\sum_i \sum_j \sum_k profits_{i,j,k} aprt_{i,j,k} \geq 10$$

```
con_1=sm.addConstr(lin_aprt_expr_2,sense=GRB.GREATER_EQUAL,rhs=10,name="Con_1")
sm.update()
```

```
con_1
```

```
Out[104]:
```

```
<gurobi.Constr Con_1>
```

Στην GUROBI η κλήση ενός περιορισμού μας εμφανίζει μόνο τον τύπο του αντικειμένου (*Constr*) και το όνομά του. Εάν είναι επιθυμητή η οπτική επιθεώρηση του περιορισμού που δημιουργήθηκε συνίσταται η δημιουργία και ανάγνωση lp αρχείου, στο οποίο στο οποίο ο περιορισμός καταγράφεται αναλυτικά.

3.4.5.3 Εισαγωγή πολλαπλών περιορισμών

Κατά την κατασκευή μοντέλων βελτιστοποίησης τις περισσότερες φορές είναι επιθυμητή η πρόσθεση πολλαπλών περιορισμών η οποία είναι δυνατή με τη χρήση της μεθόδου *model.addConstrs()* η οποία συντάσσεται ώς:

$$addConstr(generator, name = \text{" "})$$

- $generator$: Μία γεννήτρια εκφράσεων Python. Η γεννήτρια αποτελείται από μία έκφραση η οποία επαναλαμβάνεται για κάποιες συγκεκριμένες αλλαγές τιμών (ουσιαστικά μία δομή επανάληψης). Η γεννήτρια μπαίνει μέσα σε παρένθεση.
- $name$: Το όνομα του νέου περιορισμού σε χαρακτήρες ASCII. Η χρήση κενών να αποφεύγεται λόγω της αδυναμίας αποθήκευσης σε lp αρχεία. Το τελικό όνομα

κάθε περιορισμού θα είναι $name[i]$, οπού το i πέρνει τιμές ανάλογα με την επαναληπτική έκφραση της γεννήτριας εκφράσεων.

Έστω είναι επιθυμητή η εισαγωγή των παρακάτω περιορισμών, οι οποίοι χρησιμοποιούν την μεταβλητή $aprt_{i,j,k}$ της ενότητας 3.4.4.2:

$$\sum_{j}\sum_{k} space_{i,j,k} aprt_{i,j,k} \leq 400 \quad \forall i$$

```
space={(0, 0, 0):125, (0, 0, 1):215, (0, 1, 0):100, (0, 1, 1):75,
       (0, 2, 0):230, (0, 2, 1):115, (1, 0, 0):140, (1, 0, 1):55,
       (1, 1, 0):270, (1, 1, 1):195, (1, 2, 0):120, (1, 2, 1):225}
```

```
gen_expr_1=(quicksum(space[i,j,k]*aprt[i,j,k] for j in index_j for k in index_k)
            <=400 for i in index_i)
```

```
con_2=sm.addConstrs(gen_expr_1,name="Con_2")
sm.update()
```

```
con_2
```

```
Out[55]:

{0: <gurobi.Constr Con_2[0]>, 1: <gurobi.Constr Con_2[1]>}
```

Η επίκληση του περιορισμού επιστρέφει ένα λεξικό που περιέχει τα αντικείμενα *Constr* που δημιουργήθηκαν καθώς και το όνομά τους. Όπως και στην εισαγωγή ενός περιορισμού εάν είναι επιθυμητή η οπτική επιθεώρηση των περιορισμών που δημιουργήθηκαν συνίσταται η δημιουργία lp αρχείου.

3.4.6 Ορισμός της αντικειμενικής συνάρτησης

Αφού έχει γίνει η εισαγωγή των μεταβλητών και περιορισμών, σειρά έχει ο ορισμός της αντικειμενικής συνάρτησης. Η αντικειμενική συνάρτηση αντίθετα με τις μεταβλητές και τους περιορισμούς δεν είναι αντικείμενο της *gurobipy* αλλά αποτελεί ιδιότητα του κάθε μοντέλου. Κατά την εισαγωγή του, ένα αντικείμενο τύπου model δημιουργεί αυτόματα την αντικειμενική του συνάρτηση η οποία είναι η ελαχιστοποίηση του αθροίσματος όλων των μεταβλητών που περιέχονται στο μοντέλο, με όλες τις μεταβλητές να έχουν συντελεστή το μηδέν. Για τον ορισμό της επιθυμητής αντικειμενική απαιτείται η χρήση της μεθόδου

model.setObjective() η οποία επιτρέπει τον ορισμό γραμμικών ή τετραγωνικών γραμμικών συναρτήσεων και συντάσσεται ως:

$$setObjective(expr, sense = None)$$

- $expr$: Η έκφραση της αντικειμενικής συνάρτησης. Η συγκεκριμένη παράμετρος εισόδου πρέπει να είναι μια γραμμική ή τετραγωνική έκφραση (δηλαδή αντικείμενο τύπου LinExpr ή QuadExpr).
- $sense$: Τύπος της βελτιστοποίησης. Εισάγεται GRB.MINIMIZE για ελαχιστοποίηση και GRB.MAXIMIZE για μεγιστοποίηση.

Έστω απαιτείται ο ορισμός της παρακάτω αντικειμενικής η οποία χρησιμοποιεί την γραμμική έκφραση της ενότητας 3.4.5.1:

$$maximize \sum_i \sum_j \sum_k profits_{i,j,k} aprt_{i,j,k}$$

```
sm.setObjective(lin_aprt_expr_2,sense=GRB.MAXIMIZE)
sm.update()
```

Η επιθεώρηση της αντικειμενικής συνάρτησης ενός μοντέλου μπορεί να γίνει με την μέθοδο *model.getObjective()*.

```
obj=sm.getObjective()
obj
```

```
Out[63]:

<gurobi.LinExpr: 2.0 aprt[0,0,0] + 4.0 aprt[0,0,1] + 3.0 aprt[0,1,0] + 2.0 a
prt[0,1,1] + 5.0 aprt[0,2,0] + 2.0 aprt[0,2,1] + 3.0 aprt[1,0,0] + aprt[1,0,
1] + 4.0 aprt[1,1,0] + 4.0 aprt[1,1,1] + 2.0 aprt[1,2,0] + 4.0 aprt[1,2,1]>
```

3.4.7 Αποθήκευση και ανάγνωση μοντέλων από αρχεία

Συχνά η αποθήκευση ενός μοντέλου είναι επιθυμητή για πλήθος λόγων, όπως η επεξεργασία του σε αργότερο χρόνο, η εισαγωγή του σε κάποιο άλλο λογισμικό και η δυνατότητα αποστολής του σε τρίτους. Αποθήκευση ενός μοντέλου σε αρχείο γίνεται μέσω της μεθόδου *model.write()*, η οποία επιπλέον ενημερώνει το μοντέλο και συντάσσεται ως:

$$Model.write(filename)$$

- $filename$: Το όνομα (μαζί με την κατάληξη) του αρχείου που θα αποθηκευτούν τα δεδομένα.
    - .mps/.rew/.lp/.rlp : Καταλήξεις αρχείων για εγγραφή του μοντέλου.
    - .ilp : Καταλήξεις αρχείων για εγγραφή infeasible μοντέλου.
    - .sol : Καταλήξεις αρχείων για εγγραφή της λύσης του μοντέλου.
    - .prm : Καταλήξεις αρχείων για εγγραφή των ρυθμίσεων στις παραμέτρους.
    - .attr : Καταλήξεις αρχείων για εγγραφή των ρυθμίσεων στις ιδιότητες.
    - Άλλοι τύποι αρχείων είναι : .mst/.hnt/.bas/.json

```
sm.write("Serious_Model.lp")
```

Η ανάγνωση αρχείων που περιέχουν μοντέλα βελτιστοποίησης πραγματοποιείται με την συνάρτηση *read()* η οποία συντάσσεται ώς:

$$read(filename)$$

- $filename$: Το όνομα του αρχείου που περιέχει το μοντέλο μαζύ με την κατάληξη τύπου αρχείου. Έγκυρες καταλήξεις είναι .mps, .rew, .lp, .rlp, .ilp και .opb. Εάν βρεθούν παραπάνω από ένα αρχεία με το ίδιο όνομα, πραγματοποιείται ανάγνωση μόνο του πρώτου από αυτά.

Ανάγνωση ενός μοντέλου προβλήματος δρομολόγησης οχημάτων.

```
CVRP_model=read("gurobi_overview_files/CVRP.lp")
```

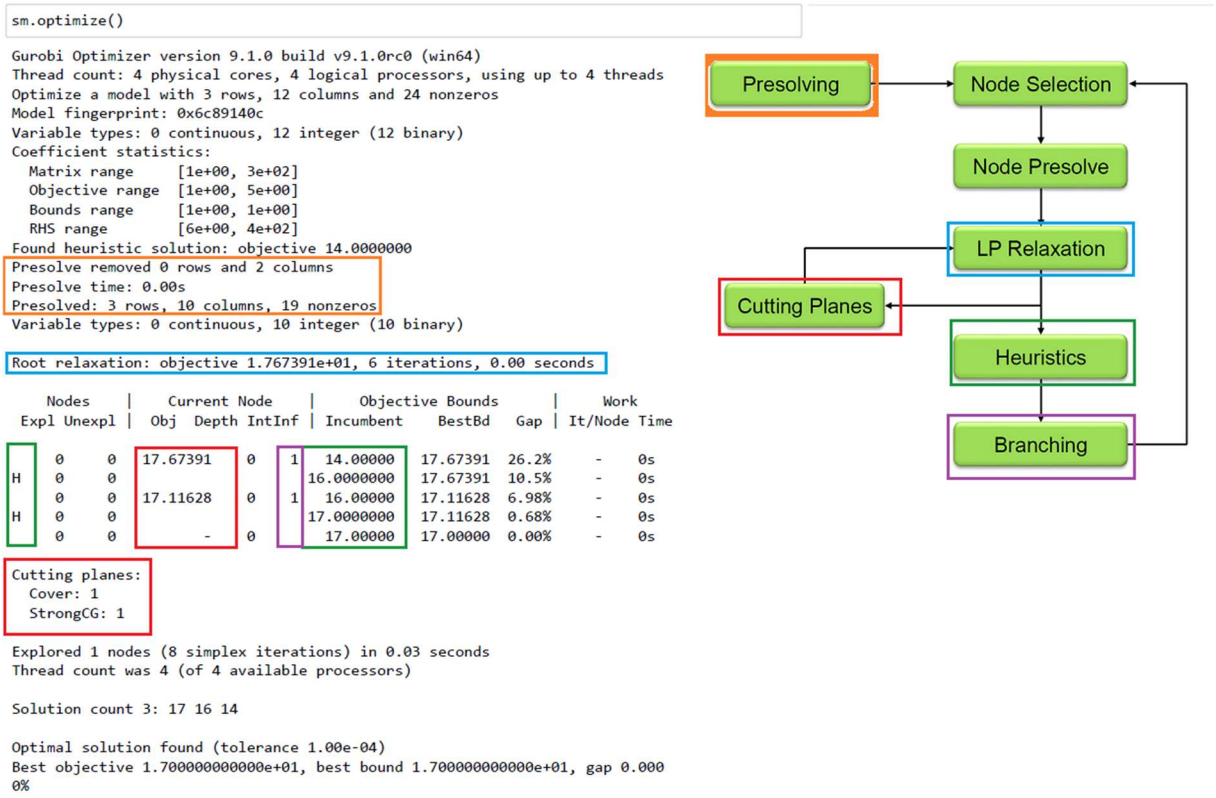
```
Read LP format model from file gurobi_overview_files/CVRP.lp
Reading time = 0.00 seconds
: 130 rows, 120 columns, 490 nonzeros
```

3.4.8 Επίλυση μοντέλων και επιθεώρηση της λύσης

Αφού έχει ολοκληρωθεί η κατασκευή του μοντέλου (εισαγωγή μεταβλητών και περιορισμών, ορισμός αντικειμενικής) η επίλυση του είναι πλέον δυνατή μέσω της μεθόδου *model.optimize()*, με την επίκληση της οποίας πραγματοποιείται και ενημέρωση του

μοντέλου. Ο solver της GUROBI αυτόματα συμπυκνώνει και μικραίνει σε μέγεθος το μοντέλο με σύσφιξη περιορισμών (presolve) και στη συνέχεια ανάλογα τον τύπο του μοντέλου (Linear, Quadratic, Quadratically-Constrained, Mixed Integer) ακολουθεί την αντίστοιχη μέθοδο επίλυσης (π.χ. στα γραμμικά μοντέλα μία βελτιωμένη εκδοχή της SIMPLEX).

Πέρα από την επίλυση του μοντέλου η *model.optimize()* επιστρέφει και ένα report με πληροφορίες για όλα τα στάδια της επίλυσης. Η επίλυση τερματίζει όταν βρεθεί η βέλτιστη λύση ή όταν ικανοποιείται κάποια συνθήκη (π.χ. υπέρβαση ενός χρονικού ορίου). Με την εύρεση της βέλτιστης λύσης εκχωρούνται οι κατάλληλές τιμές στα χαρακτηριστικά (attributes) των αντικειμένων της GUROBI, που σχετίζονται με την λύση, όπως οι τιμές των μεταβλητών ή της αντικειμενικής στη λύση.



Σχήμα 3.4: Αριστερά εικονίζεται το report που επιστρέφει η Model.optimize(). Δεξιά απεικονίζεται η αλγοριθμική διαδικασία επίλυσης ενός MIP προβλήματος (πηγή: https://www.gurobi.com/pdfs/user-events/2017-frankfurt/Algorithms-I.pdf). Τα χρωματιστά πλαίσια στο report αντιστοιχούν στα αντίστοιχα χρώματα των σταδίων της αλγοριθμικής διαδικασίας: πορτοκαλί (presolving), μπλε (lp relaxation), κόκκινο (cutting planes), πράσινο (heuristics), μοβ (branching).

Αφού έχει πραγματοποιηθεί η επίλυση του μοντέλου, η πρόσβαση στην τιμή της αντικειμενικής γίνεται με τους εξής τρόπους:

```
sm.ObjVal
```

```
Out[67]:

17.0
```

```
sm.printAttr('ObjVal')
```

```
      ObjVal
------------
          17
```

Η επιθεώρηση της τιμής μίας μεταβλητής στην λύση γίνεται ώς:

```
aprt[1,1,0].X
```

```
Out[69]:

0.0
```

Η λίστα με τις τιμές όλων των μεταβλητών απόφασης (με τη σειρά που εκχωρήθηκαν) στο μοντέλο μπορεί να αποκτηθεί ώς:

```
values_in_solution=sm.X
values_in_solution
```

```
Out[70]:

[-0.0, 1.0, 1.0, 1.0, -0.0, -0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0]
```

Εάν είναι επιθυμητή και η προβολή του ονόματος των μεταβλητών σε συνδυασμό με τις τιμές που παίρνουν στην αντικειμενική μπορεί να χρησιμοποιηθεί η παρακάτω εντολή η οποία εμφανίζει το όνομα και τις τιμές των μη μηδενικών μεταβλητών στη λύση.

```
sm.printAttr('X')
```

```
    Variable            X
-------------------------
 aprt[0,0,1]            1
 aprt[0,1,0]            1
 aprt[0,1,1]            1
 aprt[1,0,0]            1
 aprt[1,0,1]            1
 aprt[1,1,1]            1
```

Εάν είναι επιθυμητή η επιθεώρηση όλων των μεταβλητών (και των μηδενικών) μπορεί να γίνει χρήση της μεθόδου *model().getVars*:

```
sm.getVars()
```

```
Out[74]:

[<gurobi.Var aprt[0,0,0] (value -0.0)>,
 <gurobi.Var aprt[0,0,1] (value 1.0)>,
 <gurobi.Var aprt[0,1,0] (value 1.0)>,
 <gurobi.Var aprt[0,1,1] (value 1.0)>,
 <gurobi.Var aprt[0,2,0] (value -0.0)>,
 <gurobi.Var aprt[0,2,1] (value -0.0)>,
 <gurobi.Var aprt[1,0,0] (value 1.0)>,
 <gurobi.Var aprt[1,0,1] (value 1.0)>,
 <gurobi.Var aprt[1,1,0] (value 0.0)>,
 <gurobi.Var aprt[1,1,1] (value 1.0)>,
 <gurobi.Var aprt[1,2,0] (value 0.0)>,
 <gurobi.Var aprt[1,2,1] (value 0.0)>]
```

3.4.9 Χαρακτηριστικά (Attributes)

Ο κύριος μηχανισμός για απόκτηση πληροφοριών ή τροποποίησης ενός μοντέλου GUROBI γίνεται μέσω των attributes. Ορισμένα attributes παίρνουν τιμές μόνο συγκεκριμένες υπό συγκεκριμένες συνθήκες (π.χ. όταν έχει βρεθεί η λύση του μοντέλου), ενώ άλλα είναι διαθέσιμα συνεχώς (π.χ. ο αριθμός των μεταβλητών του μοντέλου). Επισκόπηση όλων των διαθέσιμων attributes γίνεται μέσω της εντολής *help(GRB.Attr)*.Τα attributes πάντα συσχετίζονται με κάποιο από τα αντικείμενα που εισάγει η gurobipy, τα οποία είναι:

- Μοντέλα (αντικείμενο τύπου model)
- Περιορισμοί (αντικείμενο τύπου Constr, QConstr, SOS και GenConstr)
- Μεταβλητές (αντικείμενο τύπου Var)

Αν και κάθε αντικείμενο έχει ξεχωριστά Attributes, επειδή ένα μοντέλο GUROBI περιέχει όλες τις μεταβλητές και περιορισμούς, η επίκληση ενός attribute που αντιστοιχεί σε μεταβλητές ή περιορισμούς σε ένα αντικείμενο model θα επιστρέψει λίστα με τις τιμές του συγκεκριμένου attribute για όλες τις μεταβλητές ή περιορισμούς που περιέχονται στο συγκεκριμένο μοντέλο.

Η επιθεώρηση ενός attribute ενός αντικειμένου κλάσης model μπορεί να γίνει με τρεις τρόπους. Ο πρώτος τρόπος, ο οποίος επιστρέφει την τιμή του attribute (δίνοντας έτσι την δυνατότητα να αποθηκεύει σε μεταβλητή) γίνεται ώς:

```
name=sm.modelName
name
```

```
Out[75]:

'Serious_Model'
```

Ο δεύτερος τρόπος είναι μέσω της μεθόδου *model.getAttr()*:

```
name=sm.getAttr("modelName")
name
```

```
Out[76]:

'Serious_Model'
```

Ο τρίτος τρόπος είναι μέσω της μεθόδου *model.printAttr()*. Η μέθοδος αυτή εκτυπώνει με όμορφο τρόπο τα αποτελέσματα στην οθόνη.

```
sm.printAttr("modelName")
```

```
   modelName
------------
Serious_Model
```

Η απόκτηση της τιμής ενός attribute μίας μεταβλητής ή περιορισμού γίνεται με παρόμοιο τρόπο:

```
name=aprt[0,0,1].varName
name
```

```
Out[79]:

'aprt[0,0,1]'
```

```
name=con_2[0].constrName
name
```

```
Out[80]:

'Con_2[0]'
```

Άλλος ένας τρόπος είναι μέσω της μεθόδου *var.getAttr()* για τις μεταβλητές και αντίστοιχα *constr.getAttr()* για τους περιορισμούς.

```
name=aprt[0,0,1].getAttr("varName")
name
```

```
Out[81]:

'aprt[0,0,1]'
```

```
name=con_2[0].getAttr("constrName")
name
```

```
Out[82]:

'Con_2[0]'
```

Οι μεταβλητές και οι περιορισμοί αποθηκεύονται μέσα στα αντικείμενα τύπου model, με την επίκληση attributes μεταβλητών (αντικείμενα τύπου Var) και περιορισμών (αντικείμενα τύπου Constr) σε αντικείμενα model επιστρέφουν τις τιμές των attributes όλων των μεταβλητών ή περιορισμών αντίστοιχα:

```
names=sm.constrName
names
```

```
Out[83]:

['Con_1', 'Con_2[0]', 'Con_2[1]']
```

```
names=sm.getAttr("constrName")
names
```

```
Out[84]:

['Con_1', 'Con_2[0]', 'Con_2[1]']
```

3.4.10 Τροποποίηση και αφαίρεση στοιχείων μοντέλων

Κατά τη δημιουργία επίλυσης ενός προβλήματος βελτιστοποίησης πιθανώς χρειαστεί κάποια τροποποίηση του μοντέλου. Στην GUROBI για γίνει επιτυχής τροποποίηση το μοντέλο πρέπει να είναι άλυτο, δηλαδή τα attributes που σχετίζονται με την λύση να είναι κενά. Επιστροφή ενός μοντέλου στην άλυτη του κατάσταση γίνεται μέσω της μεθόδου *model.reset()*.

```
sm.reset()
```

```
Discarded solution information
```

Τροποποιήσεις γίνονται εύκολα με την αλλαγή συγκεκριμένων attributes. Συνηθισμένα τέτοια attributes είναι:

- lb: Κάτω όριο μεταβλητής.
- ub: Άνω όριο μεταβλητής.
- obj: Συντελεστής της μεταβλητής στην αντικειμενική.
- $vType$: Τύπος της μεταβλητής.

- $sense$: Η σχέση ισότητας/ανισότητας ενός περιορισμού.
- rhs: Constraint right-hand side value

Η αλλαγή της τιμής ενός χαρακτηριστικού μίας μεταβλητής και περιορισμού γίνεται με τις μεθόδους *var.setAttr()* και *constr.setAttr()* αντίστοιχα (*model.setAttr()* για μοντέλα).

```
print(aprt[0,0,1].getAttr("vType"))
aprt[0,0,1].setAttr("vType",GRB.INTEGER)
sm.update()
print(aprt[0,0,1].getAttr("vType"))
```

```
B
I
```

```
print(con_2[0].getAttr("sense"))
con_2[0].setAttr("sense","=")
sm.update()
print(con_2[0].getAttr("sense"))
```

```
<
=
```

Εάν απαιτείται η αφαίρεση ενός ολόκληρου μοντέλου, τότε γίνεται χρήση της μεθόδου *Model.dispose()*, συνήθως όμως επιθυμητή είναι η αφαίρεση συγκεκριμένων μεταβλητών ή περιορισμών ενός μοντέλου. Σε αυτήν την περίπτωση γίνεται η χρήση της μεθόδου *Model.remove()* η οποία συντάσσεται ως:

$$remove(items)$$

- $items$: Τα αντικείμενα που θέλω να διαγράψω. Δεκτά είναι αντικείμενα μεταβλητών, περιορισμών η και λεξικά που περιέχουν μεταβλητές και περιορισμούς. Στην περίπτωση του λεξικού, διαγράφονται οι μεταβλητές και όχι τα κλειδιά.

```
cons=smcopy.getConstrs()
cons
```

```
Out[95]:

[<gurobi.Constr Con_1>, <gurobi.Constr con2-0>, <gurobi.Constr Con_2[1]>]
```

```
smcopy.remove([cons[0],cons[1]])
smcopy.update()
```

```
cons=smcopy.getConstrs()
cons
```

```
Out[97]:

[<gurobi.Constr Con_2[1]>]
```

3.4.11 Παράμετροι (Parameters)

Πολλές φορές είναι επιθυμητός ο έλεγχος της διαδικασίας επίλυσης καθώς η αποδοτικότητα ορισμένων αλγορίθμων διαφέρει για κάποιους τύπους προβλημάτων. Ακόμα συχνά απαιτείται ο πρόωρος τερματισμός του solver σε κάποια δυσεπίλυτα προβλήματα, οπού αρκεί η λύση να πληρεί ορισμένες προϋποθέσεις ή υπάρχει όριο στον χρόνο επίλυσης. Στην Gurobi ο έλεγχος της διεργασίας βελτιστοποίησης γίνεται μέσω των παράμετρών (parameters). Όλες οι παράμετροι έχουν κάποιες εξ' ορισμού τιμές οι οποίες μπορούν να αλλάξουν ανάλογα με τις ανάγκες επίλυσης. Μία αναλυτική λίστα με τα ονόματα και τον ρόλο των παραμέτρων μπορώ να πάρω μέσω των εντολών *help(GRB.param)*.

Πληροφορίες για την λειτουργία μίας συγκεκριμένης παραμέτρου είναι προσβάσιμες με την συνάρτηση *paramHelp("name of parameter").*

```
paramHelp("TimeLimit")
```

```
PARAMETER NAME:
  TimeLimit

TYPE:
  Double

PURPOSE:
  Terminate after the specified time has been expended optimizing a model.

VALUES:
  Minimum : 0
  Maximum : 1e100
  Default : 1e100
```

Όταν είναι επιθυμητή η αλλαγή κάποιας παραμέτρου χρησιμοποιείται η συνάρτηση *serParam()*, η οποία συντάσσεται ώς :

$$serParam(paramname, newvalue)$$

- $paramname$: Το όνομα της παραμέτρου πού αλλάζει τιμή.
- *newvalue* : Η νέα τιμή της παραμέτρου.

```
setParam("TimeLimit",2)
```

```
Changed value of parameter TimeLimit to 2.0
   Prev: inf  Min: 0.0  Max: inf  Default: inf
```

3.4.12 Εισαγωγή περιορισμών callback

Οι ρουτίνες callback της GUROBI αποτελούν μία αρκετά προχωρημένη λειτουργία της διεπαφής. Η εκτενής χρήση τους στα μοντέλα βελτιστοποίησης αυτής της διπλωματικής καθιστά αναγκαία μια συνοπτικής παρουσίασης τους. Οι ρουτίνες αυτές κάνουν χρήση ενός ζεύγους arguments,where και what. Όταν γίνεται χρήση μίας callback συνάρτησης του χρήστη, το where argument προσδιορίζει σε ποια φάση της βελτιστοποίησης λαμβάνει μέρος η callback συνάρτηση (presolve, simplex, barrier, MIP). Εάν το where πάρει την τιμή *GRB.Callback.MIPSOL* σημαίνει πως η κλήση της συνάρτησης callback πραγματοποιείται αφού έχει γίνει ο υπολογισμός της λύσης του μοντέλου.

Εντός της συνάρτησης μπορεί να γίνει ανάκτηση πληροφοριών σχετικές με την λύση, όπως οι τιμές των μεταβλητών, καθώς και η πρόσθεση περιορισμών callback (ή lazy). Η χρήση των περιορισμών callback σε ένα MIP μοντέλο γίνεται τυπικά όταν ένα σύνολο περιορισμών είναι υπερβολικά μεγάλο για να ορισθεί αναλυτικά. Προσθέτοντας στο μοντέλο μόνο τους περιορισμού που παραβιάζονται στις τις λύσεις κατά την έρευνα διακλάδωσης και τομής (branch and cut), είναι δυνατό να γίνει η εύρεση της βέλτιστης λύσης με την εισαγωγή μόνο ενός μικρού αριθμού περιορισμών. Η πρόσθεση αυτών των περιορισμών γίνεται μέσω της μεθόδου *Model.cbLazy(),* η οποία συντάσσεται ως:

$$cbLazy(lhs, sense = None, rhs = None)$$

- lhs : Το αριστερό μέρος του νέου περιορισμού. Μπορεί να είναι μία σταθερά, μία μεταβλητή, μία γραμμική έκφραση ή μία τετραγωνική έκφραση.
- $sense$: Η σχέση μεταξύ του αριστερού και δεξιού μέρους του περιορισμού. Πιθανές εκφράσεις που μπορεί να πάρει είναι GRB.LESS_EQUAL (για <=), GRB.GREATER_EQUAL (για >=) και GRB.EQUAL (για =).

- rhs : Το δεξιό μέρος του νέου περιορισμού. Μπορεί να είναι μία σταθερά, μία μεταβλητή, μία γραμμική έκφραση ή μία τετραγωνική έκφραση.

Για την χρήση callbacks κατά την επίλυση πρέπει αρχικά η global μεταβλητή που σχετίζεται με τους περιορισμούς callback να εκχωρηθεί στο *model._vars* και η τιμή της παραμέτρου *lazyConstraints* να γίνει ίση με την μονάδα. Το όνομα της συνάρτησης callback θα μπει μέσα στην μέθοδο model.optimize().

```
m._vars = x
m.Params.lazyConstraints = 1
m.optimize(subtourelim)
```

Η συνάρτηση callback πρέπει να έχει αυτή την μορφή:

```
def subtourelim(model, where):
    if where == GRB.Callback.MIPSOL:
        # make a list of edges selected in the solution
        vals = model.cbGetSolution(model._vars)
        selected = [(i, j) for i, j in model._vars.keys() if vals[i,j]>0.5]
        ...
        Κώδικας
        ...
        model.cbLazy(quicksum(model._vars[i,j] for i,j in A)>=rs)
```

Κεφάλαιο 4. Κατασκευή και αξιολόγηση μοντέλων

Στο τρέχον κεφάλαιο παρουσιάζεται ο κώδικας κατασκευής των μοντέλων βελτιστοποίησης του Κεφαλαίου 2 με τη χρήση της διεπαφής Python της GUROBI σε περιβάλλον Jupyter Notebooks, ενώ στην συνέχεια πραγματοποιείται σύγκριση και σχολιασμός της απόδοσης τους.

Κατά την παρουσίαση του κώδικα, βάρος θα δοθεί κυρίως στο κομμάτι της κατασκευής και επίλυσης του κάθε μοντέλου βελτιστοποίησης με κάποια βασική αναφορά της μορφής των απαραίτητων δεδομένων εισόδου. Αναλυτικοί κώδικες που περιλαμβάνουν την δημιουργία τυχαίων προβλημάτων, μοντελοποίηση και οπτικοποίηση των αποτελεσμάτων δίνονται στο παράρτημα Α.

Το αντικείμενο της τρέχουσας διπλωματικής δεν είναι η μεθοδική ανάλυση της απόδοσης των αλγορίθμων αλλά μία συγκριτική αξιολόγηση της συμπεριφοράς τους, καθώς και μία εικόνα των δυνατοτήτων του solver της Gurobi. Επίσης γίνεται αναλυτική αναφορά στον τρόπο δημιουργίας των προβλημάτων πάνω στα οποία τρέχουν οι κώδικες, ενώ τα δεδομένα και αποτελέσματα επίλυσης (περιγραφή της καλύτερης λύσης) των προβλημάτων καταγράφονται στο παράρτημα Β.

Τα χαρακτηριστικά του συστήματος, στο οποίο γίνονται οι μετρήσεις επιδόσεων, έχει τα εξής χαρακτηριστικά:

- Λειτουργικό σύστημα : Windows 10 Home (64 bit)
- Επεξεργαστής : Intel® Core™ i5-4460 @ 3.20 GHz
- Εγκατεστημένη RAM : 8 Gb

Τα χρησιμοποιούμενα λογισμικά και οι εκδόσεις τους είναι:

- Python 3.9
- Jupyter Notebook 6.1.4
- Gurobi Optimizer version 9.1.0

4.1 Κατασκευή και αξιολόγηση του VRP

Στην *Ενότητα 2.3.3* παρουσιάζονται δύο μοντελοποιήσεις του CVRP ως πρόβλημα ακέραιου γραμμικού προγραμματισμού. Η πρώτη αποτελεί επέκταση του μοντέλου του TSP των Miller, Tucker και Zemlin και είναι παράδειγμα ενός κλασικού προβλήματος ακέραιου προγραμματισμού, ενώ η δεύτερη μοντελοποίηση, των Laporte, Mercure και Nobert, κάνει χρήση δυναμικά προστιθέμενων περιορισμών (περιορισμοί callbacks). Για ευκολία τα προβλήματα θα αναφέρονται ως CVRP1 και CVRP2 αντίστοιχα.

Πρώτού γίνει η παρουσίαση κατασκευής των μοντέλων, θα γίνει αναφορά στα δεδομένα εισόδου και την μορφή τους για την ευκολότερη κατανόηση του κώδικα.

- n : Αριθμός πελατών
- NV : Αριθμός οχημάτων
- Q : Η μέγιστη χωρητικότητα οχήματος.
- N : Λίστα της μορφής $[1,2,\dots,n]$. Εκφράζει το σύνολο των πελατών.
- V : Λίστα της μορφής $[0,1,2,\dots,n]$. Εκφράζει το σύνολο των κόμβων.
- A : Λίστα πλειάδων της μορφής $[(0,1),(0,2),\dots,(n-1,n)]$. Εκφράζει το σύνολο των τόξων μετάβασης μεταξύ σημείων.
- c : Λεξικό με κλειδιά πλειάδες της μορφής $\{(i,j): c_{ij},\dots\}$ όπου $(i,j) \in A$. Περιέχει το κόστος μετάβασης κάθε τόξου (i,j).
- q : Λεξικό με κλειδιά τιμές της μορφής $\{i: d_i,\dots\}$ όπου $i \in N$. Περιέχει πληροφορίες για την ζήτηση κάθε πελάτη.

4.1.1 Ο κώδικας κατασκευής και επίλυσης του CVRP1

Αρχικά δημιουργείται ένα κενό αντικείμενο Model.

```
m=Model("Capacitated Vehicle Routing Problem")
```

Στη συνέχεια ακολουθεί ο ορισμός των μεταβλητών x και u του προβλήματος. Οι μεταβλητές είναι εξ' ορισμού θετικές.

```
x=m.addVars(A,vtype=GRB.BINARY,name='x')
u=m.addVars(N,vtype=GRB.CONTINUOUS,name='u')
```

Αφού έχει γίνει η εισαγωγή των μεταβλητών οι ορίζονται οι περιορισμοί. Η αρίθμηση στον κώδικά γίνεται βάση της σειράς του κάθε περιορισμού. Η αντιστοίχιση με τις εκφράσεις της *Ενότητα 2.3.3* γίνεται ώς : (*αρίθμηση κώδικά) = (αρίθμηση ενότητας 2.3.3) - 1*

```
con1=m.addConstrs((quicksum(x[i,j] for j in V if j!=i)==1 for i in N),name='con1')
con2=m.addConstrs((quicksum(x[i,j] for i in V if i!=j)==1 for j in N),name='con2')
con3=m.addConstr((quicksum(x[0,j] for j in N)==NV),name="con3")
con4=m.addConstrs((u[i]-u[j]+Q*x[i,j]<=Q-q[j] for i,j in A if i!=0 and j!=0),name='con4')
con5_1=m.addConstrs((q[i]<=u[i] for i in N),name='con5_1')
con5_2=m.addConstrs((u[i]<=Q for i in N),name='con5_2')
```

Ο περιορισμός 4 εναλλακτικά ορίζεται και ώς:

```
con4=m.addConstrs((x[i,j]==1) >> (u[i]+q[j]==u[j]) for i,j in A if i!=0 and j!=0)
```

Τελικώς γίνεται ο ορισμός της αντικειμενικής συνάρτησης.

```
m.setObjective(quicksum(x[i,j]*c[i,j] for i,j in A),sense=GRB.MINIMIZE)
```

4.1.2 Ο κώδικας κατασκευής και επίλυσης του CVRP2

Αρχικά δημιουργείται ένα κενό αντικείμενο Model.

```
m=Model("Capacitated Vehicle Routing Problem")
```

Ακολουθεί ο ορισμός της μεταβλητής x.

```
x=m.addVars(A,vtype=GRB.BINARY,name='x')
```

Γίνεται ο ορισμός των σταθερών περιορισμών του προβλήματος.

```
con1=m.addConstrs((quicksum(x[i,j] for j in V if j!=i)==1 for i in N),name='con1')
con2=m.addConstrs((quicksum(x[i,j] for i in V if i!=j)==1 for j in N),name='con2')
con3=m.addConstr((quicksum(x[0,j] for j in N)==len(K)),name="con3")
```

Στην συνέχεια γίνεται ο ορισμός της αντικειμενική συνάρτησης.

```
m.setObjective(quicksum(x[i,j]*c[i,j] for i,j in A),sense=GRB.MINIMIZE)
```

Οι callback περιορισμοί, όπως αναλύθηκε στην *Ενότητα 3.4.12*, προστίθενται στο μοντέλο κατά την διαδικασία επίλυσης του προβλήματος μέσω μίας συνάρτησης callback (εδώ ονομάζεται *subtourlim*).

```
m._vars = x
m.Params.lazyConstraints = 1
m.optimize(subtourelim)
```

Η callback συνάρτηση καλείται να εντοπίσει της διαδρομές στην λύση του προβλήματος και να εφαρμόσει σε αυτές τον περιορισμό (5) του CVRP2.

```
def subtourelim(model, where):
    if where == GRB.Callback.MIPSOL:
        # make a list of edges selected in the solution
        vals = model.cbGetSolution(model._vars)
        selected = [(i, j) for i, j in model._vars.keys() if vals[i,j]>0.5]
```

- $rute$: Λίστα της μορφής $[k_1, k_2, \dots, k_r]$. Η λίστα περιέχει τα σημεία της διαδρομής (εκτός του σημείου εκκίνησης), τα οποία δεν είναι απαραίτητο να είναι στην σωστή αλληλουχία. Οι διαδρομές μπορεί να είναι είτε κανονικές, είτε κλειστές (δεν επικοινωνούν με το σημείο αφετηρίας).
- $rs1$: Ο αμέσως επόμενος μεγαλύτερος ακαριαίος του αριθμού που προκύπτει από την διαίρεση συνολικής ζήτησης όλων των σημείων της διαδρομής $rute$ διά της μέγιστης χωρητικότητας οχήματος.

```
model.cbLazy(quicksum(model._vars[i,j] for i,j in A if i in rute if j not in rute)>=rs1)
```

4.1.3 Δημιουργία τυχαίου προβλήματος

Τα προβλήματα πάνω στα οποία θα τρέξουνε οι κώδικες των CVRP1 και CVRP2 δημιουργούνται προσομοιώνοντας ένα πραγματικό πρόβλημα. Αρχικά ορίζεται η μέγιστη χωρητικότητα οχήματος Q, ο αριθμός των πελατών n καθώς και μία μέγιστη τιμή των συντεταγμένων του x και y, max_x και max_y. Στην συνέχεια δημιουργούνται τυχαία οι συντεταγμένες (x_i, y_i) του σημείου εκκίνησης και των πελατών, με τα x_i και y_i να παίρνουν τυχαίες τιμές μεταξύ $(0, max\_x)$ και $(0, max\_y)$ αντίστοιχα. Οι ζητήσεις q κάθε πελάτη παίρνουν τυχαίες τιμές μεταξύ μιας μέγιστης και ελάχιστης τιμής ζήτησης, οι οποίες προκύπτουν από την Q και δύο προκαθορισμένες παραμέτρους b_1, b_2 ως:

$$q_{max} = round\left(\frac{Q}{b_1}\right) \ και \ q_{min} = round\left(\frac{Q}{b_2}\right) \ όπου \ b_1 < b_2$$

Το κόστος μετάβασης μεταξύ των σημείων ισούται με την γεωμετρική τους απόσταση:

$$d_{ij} = c_{ij} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}$$

Τελικώς ο αριθμός των οχημάτων NV (ή $|K|$) προκύπτει από το άθροισμα $evnv + tol$, όπου tol ένας προκαθορισμένος ακέραιος (όχι απαραίτητα θετικός) αριθμός και $evnv$ ο αναμενόμενος αριθμός απαραίτητων οχημάτων ο οποίος υπολογίζεται από:

$$evnv = \left\lceil \frac{n * (q_{max} + q_{min})}{2Q} \right\rceil = \frac{n}{2}\left(\frac{1}{b_1} + \frac{1}{b_2}\right)$$

4.1.4 Εύρεση αρχικής εφικτής λύσης

Αρχίζοντας την διαδικασία επίλυσης ενός προβλήματος ο solver της GUROBI ίσως χρειάζεται κάποιο χρονικό διάστημα μέχρι να εντοπίσει κάποια αρχική λύση, το μέγεθος του οποίου εξαρτάται από την δυσκολία του προβλήματος. Για την αποφυγή μεγάλων χρόνων αναμονής συνίσταται, εάν είναι εφικτό, η εισαγωγή μίας αρχικής εφικτής λύσης στο μοντέλο ώστε να αρχίσει άμεσα η διαδικασία της βελτιστοποίησης. Καλής ποιότητας αρχικές εφικτές λύσεις μειώνουν αισθητά τον χρόνο επίλυσης του προβλήματος.

Στο CVRP η ανάγκη μιας εύκολης αρχικής λύσης καλύπτεται από μία βασική ανάθεση πελατών σε δρομολόγια, τέτοια ώστε η συνολική ζήτηση των πελατών σε κάθε δρομολόγιο να μην ξεπερνάει την μέγιστη χωρητικότητα οχήματος Q. Μία τέτοια ανάθεση προκύπτει από την επίλυση του παρακάτω προβλήματος ακέραιου προγραμματισμού:

$\mathbf{N} \rightarrow$ το σύνολο των πόλεων, όπου $\mathrm{N} = \{1,2,\ldots,\mathrm{n}\}$ (ήδη γνωστό)
$\mathbf{K} \rightarrow$ σύνολο οχημάτων, όπου $\mathrm{K} = \{1,2,\ldots,\mathrm{k}\}$ (ήδη γνωστό)
$\mathbf{Q} \rightarrow$ η μέγιστη χωρητικότητα ενός οχήματος (ήδη γνωστό)
$\mathbf{q_i} \rightarrow$ η ζήτηση του πελάτη $\mathrm{i} \in \mathrm{N}$ (ήδη γνωστό)

Οι μεταβλητές απόφασης και οι βοηθητικές μεταβλητές του μοντέλου ορίζονται ως :

$\boldsymbol{RQ_j}$ → ακεραία μεταβλητή απόφασης ίση με την συνολική ζήτηση του δρομολογίου $j \in K$.

$\boldsymbol{a_{ij}}$ → δυαδική μεταβλητή, η οποία παίρνει την τιμή 1 όταν ο πελάτης $i \in N$ αναθέτεται στο όχημα $j \in K$.

Το μοντέλο βελτιστοποίησης θα είναι:

$$minimize \quad |K|Q - \sum_{j \in K} RQ_j \quad (1)$$

$$\sum_{i \in N} q_i a_{ij} = RQ_j \,, \forall\, j \in K \quad (2)$$

$$\sum_{j \in K} a_{ij} = 1 \;\,, \forall\, i \in N \quad (3)$$

$$0 < RQ_j \leq \; Q \,, \forall\, j \in K \quad (4)$$

Η (1) φροντίζει την ελαχιστοποίηση του συνολικού κενού χώρου στα οχήματα. Ο περιορισμός (2) καθορίζει την σχέση των μεταβλητών απόφασης και εκφράζει την σχέση που προκύπτει μεταξύ της συνολικής ζήτησης που εξυπηρετεί το δρομολόγιο και από την ανάθεση πελάτη στο δρομολόγιο. Ο περιορισμός, ο (3) εξασφαλίζει πως κάθε πελάτης ανατίθεται μόνο σε ένα δρομολόγιο και ο (4) φροντίζει πως σε κάθε δρομολόγιο ανατίθεται τουλάχιστον ένας πελάτης και πως τηρούνται τα όρια χωρητικότητας.

Οι αναθέσεις που προκύπτουν από το παραπάνω μοντέλο εισάγονται στο CVRP ως αρχικά εφικτά δρομολόγια. Η εύρεση της βέλτιστης αλληλουχίας επίσκεψης των σημείων κάθε αρχικού δρομολογίου, άρα και βελτίωση της αρχικής εφικτής λύσης, επιτυγχάνεται με την επίλυση ενός TSP, παρόλα αυτά για λόγους απλότητας γίνεται χρήση αλληλουχίας αύξουσας σειράς βάσει της αρίθμησης των σημείων.

4.1.5 Αποτελέσματα επιδόσεων CVRP

Σε αυτή την ενότητα γίνεται παρουσίαση των επιδόσεων των CVRP1 και CVRP2 σε προβλήματα μικρού (10,20 πελάτες), μεσαίου (30,50 πελάτες) και μεγάλου (100 πελάτες) μεγέθους για διάφορους τύπους ζήτησης με μέγιστη χωρητικότητα οχημάτων Q=400. Οι μέγιστες τιμές συντεταγμένων $max\_x = 200$ και $max\_y = 100$. Ό μέγιστος επιτρεπτός χρόνος εκτέλεσης κώδικα είναι 10 λεπτά (600 δευτερόλεπτα).

Πίνακας 4-1: Αποτελέσματα επιδόσεων CVRP σε πρόβλημα 10 πελατών.

10 πελάτες	Οχήματα	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$b1$=4 , $b2$=10					
CVRP1 tol=0	2	120	0,3	0%	452,65
CVRP2 tol=0	2	110	0,3	0%	452,64
CVRP1 tol=1	3	120	0,17	0%	469,73
CVRP2 tol=1	3	110	0,03	0%	469,73
$b1$=10 , $b2$=40					
CVRP1 tol=0	1	120	0,2	0%	385,54
CVRP2 tol=0	1	110	0,03	0%	385,54
CVRP1 tol=1	2	120	0,13	0%	416,98
CVRP2 tol=1	2	110	0,03	0%	416,98

Πίνακας 4-2: Αποτελέσματα επιδόσεων CVRP σε πρόβλημα 20 πελατών.

20 πελάτες	Οχήματα	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$b1$=4 , $b2$=10					
CVRP1 tol=0	4	440	600	9,14%	1101,51
CVRP2 tol=0	4	420	600	8,85%	1101,51
CVRP1 tol=1	5	440	83,97	0%	1110,74
CVRP2 tol=1	5	420	76,24	0%	1110,74
$b1$=10 , $b2$=40					
CVRP1 tol=0	2	440	600	6,10%	685,15
CVRP2 tol=0	2	420	0,08	0,00%	685,15
CVRP1 tol=1	3	440	0,36	0%	691,32
CVRP2 tol=1	3	420	0,04	0%	691,32

Πίνακας 4-3: Αποτελέσματα επιδόσεων CVRP σε πρόβλημα 30 πελατών.

30 πελάτες	Οχήματα	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$b1$=4 , $b2$=10					
CVRP1 tol=0	6	960	600	15,14%	1135,79
CVRP2 tol=0	6	930	600	8,86%	1163,98
CVRP1 tol=1	7	960	600	10,41%	1151,46
CVRP2 tol=1	7	930	600	2,40%	1151,46
$b1$=10 , $b2$=40					
CVRP1 tol=0	2	960	600	10,85%	753,09
CVRP2 tol=0	2	930	0,36	0,00%	746,66
CVRP1 tol=1	3	960	600	4,89%	765,19
CVRP2 tol=1	3	930	0,21	0,00%	765,19

Πίνακας 4-4: Αποτελέσματα επιδόσεων CVRP σε πρόβλημα 50 πελατών.

50 πελάτες	Οχήματα	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$b1$=4 , $b2$=10					
CVRP1 tol=0	9	2600	600	47,98%	2555,14
CVRP2 tol=0	9	2550	600	62,82%	5109,02
CVRP1 tol=1	10	2600	600	44,14%	2584,28
CVRP2 tol=1	10	2550	600	61,10%	5151,66
$b1$=10 , $b2$=40					
CVRP1 tol=0	4	2600	600	13,13%	1046,42
CVRP2 tol=0	4	2550	600	3,86%	1048,71
CVRP1 tol=1	5	2600	600	11,05%	1065,3
CVRP2 tol=1	5	2550	600	0,88%	1055,94
$b1$=20 , $b2$=100					
CVRP1 tol=0	2	2600	13,24	0,00%	869,38
CVRP2 tol=0	2	2550	0,65	0,00%	869,38
CVRP1 tol=1	3	2600	2,18	0,00%	878,38
CVRP2 tol=1	3	2550	0,29	0,00%	878,48

Πίνακας 4-5: Αποτελέσματα επιδόσεων CVRP σε πρόβλημα 100 πελατών.

100 πελάτες	Οχήματα	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$b1$=4 , $b2$=10					
CVRP1 tol=0	18	10200	600	51,54%	4238,86
CVRP2 tol=0	18	10100	600	70%	8937,61
CVRP1 tol=1	19	10200	600	53,27%	4582,33
CVRP2 tol=1	19	10100	600	71,93%	10053,13
$b1$=10 , $b2$=40					
CVRP1 tol=0	7	10200	600	44,84%	2218,98
CVRP2 tol=0	7	10100	600	83%	8657,63
CVRP1 tol=1	8	10200	600	42,00%	2177,41
CVRP2 tol=1	8	10100	600	83,95%	9041,07
$b1$=20 , $b2$=100					
CVRP1 tol=0	3	10200	600	37,10%	1760,67
CVRP2 tol=0	3	10100	600	84%	8335
CVRP1 tol=1	4	10200	600	37,17%	1838,61
CVRP2 tol=1	4	10100	600	84,90%	8891,86

Τα συμπεράσματα που προκύπτουν βάσει των επιδόσεων των CVRP1 και CVRP2 είναι τα εξής:

- Για μικρά και μεσαία προβλήματα το CVRP2 είναι ταχύτερο του CVRP1, ενώ στα μεγάλα προβλήματα παρατηρείται το αντίθετο.
- Η αύξηση των $b1, b2$ συνδέεται με γρηγορότερους χρόνους επίλυσης, που υποδηλώνει πως προβλήματα λίγων δρομολογίων πολλών πελατών λύνονται ευκολότερα από προβλήματα πολλών δρομολογίων λίγων πελατών.

4.2 Κατασκευή και αξιολόγηση του (1-1)PDP

Στην *Ενότητα 2.4.3* παρουσιάζονται δύο μοντελοποιήσεις του PDP ως πρόβλημα ακέραιου γραμμικού προγραμματισμού. Η πρώτη αποτελεί τροποποίηση της μοντελοποίησης του VRPTW των Cordeau et al. [12] και κάνει χρήση 3 δεικτών και μεταβλητών απόφασης. Η δεύτερη μοντελοποίηση, των Ropke et al. [14], κάνει χρήση μίας μεταβλητής απόφασης και βασίζεται σε δυναμικά δημιουργούμενους callback περιορισμούς. Για ευκολία τα προβλήματα θα αναφέρονται ως PDP1 και PDP2 αντίστοιχα.

Πρώτού γίνει η παρουσίαση κατασκευής των μοντέλων, θα γίνει αναφορά στα δεδομένα εισόδου και την μορφή τους για την ευκολότερη κατανόηση του κώδικα. Τα δεδομένα που χρησιμοποιούνται και στις δύο μοντελοποιήσεις είναι:

- n : Αριθμός πελατών
- P : Λίστα της μορφής $[1,2,\dots,n]$. Εκφράζει το σύνολο των πελατών φόρτωσης.
- D : Λίστα της μορφής $[1+n, 2+n, \dots, 2n]$. Εκφράζει το σύνολο των πελατών παράδοσης.
- V : Λίστα της μορφής $[0,1,\dots,2n, 2n+1]$. Εκφράζει το σύνολο όλων των κόμβων.
- A : Λίστα της μορφής $[(0,1),\dots,(2n, 2n+1)]$. Εκφράζει το σύνολο των τόξων.
- q : Λεξικό της μορφής $\{i: q_i, \dots\}$ όπου $i \in V$. Περιέχει πληροφορίες για την ζήτηση κάθε πελάτη (θετική=pick up, αρνητική=delivery). Τα σημεία έναρξης (0) και τερματισμού $(2n+1)$ έχουν μηδενική ζήτηση.
- ei : Λεξικό της μορφής $\{i: e_i, \dots\}$ όπου $i \in V$. Περιέχει πληροφορίες για τον συντομότερο επιτρεπτό χρόνο επίσκεψης κάθε σημείου.
- li : Λεξικό της μορφής $\{i: l_i, \dots\}$ όπου $i \in V$. Περιέχει πληροφορίες για τον αργότερο επιτρεπτό χρόνο επίσκεψης κάθε σημείου.
- di : Λεξικό της μορφής $\{i: d_i, \dots\}$ όπου $i \in V$. Περιέχει πληροφορίες για τον απαιτούμενο χρόνο εξυπηρέτησης κάθε σημείου.

4.2.1 Ο κώδικας κατασκευής και επίλυσης του PDP1

Ο κώδικας του PDP1 χρειάζεται επιπλέον τα εξής δεδομένα.

- NV : Ο αριθμός οχημάτων.
- K : Λίστα της μορφής $[1,2,\dots,NV]$. Εκφράζει το σύνολο των οχημάτων.
- AK : Λίστα της μορφής $[(i,j,k),\dots]$ όπου $(i,j) \in A$ και $k \in K$. Περιέχει όλους του πιθανούς συνδυασμούς τόξων και οχημάτων.
- VK : Λίστα της μορφής $[(i,k),\dots]$ όπου $i \in V$ και $k \in K$.. Περιέχει όλους του πιθανούς συνδυασμούς σημείων και οχημάτων.
- Ck : Λεξικό της μορφής $\{k: C_k, \dots\}$ όπου $k \in K$. Περιέχει πληροφορίες για την μέγιστη χωρητικότητα κάθε οχήματος.
- $cijk$: Λεξικό της μορφής $\{(i,j,k): c_{ijk}, \dots\}$ όπου $(i,j) \in A$ και $k \in K$. Περιέχει πληροφορίες για τα κόστη μετάβασης τόξων κάθε οχήματος.
- $tijk$: Λεξικό της μορφής $\{(i,j,k): t_{ijk}, \dots\}$ όπου $(i,j) \in A$ και $k \in K$. Περιέχει πληροφορίες για τους χρόνους μετάβασης τόξων κάθε οχήματος.

- Tk : Λεξικό της μορφής $\{k: T_k, \dots\}$ όπου $k \in K$. Περιέχει πληροφορίες για την μέγιστη επιτρεπτή διάρκεια δρομολογίου κάθε οχήματος.

Αρχικά δημιουργείται ένα κενό αντικείμενο Model.

```
m=Model("Pick_Up_and_Delivery_Model")
```

Στη συνέχεια ακολουθεί ο ορισμός των μεταβλητών x, Q και B του προβλήματος. Οι μεταβλητές είναι εξ' ορισμού θετικές.

```
x=m.addVars(AK,vtype=GRB.BINARY,name='x')
Q=m.addVars(VK,vtype=GRB.CONTINUOUS,name='Q')
B=m.addVars(VK,vtype=GRB.CONTINUOUS,name='B')
```

Αφού έχει γίνει η εισαγωγή των μεταβλητών οι ορίζονται οι περιορισμοί. Η αρίθμηση στον κώδικα γίνεται βάση της σειράς του κάθε περιορισμού. Η αντιστοίχιση με τις εκφράσεις της *Ενότητα 2.4.3* γίνεται ώς : (*αρίθμηση κώδικά) = (αρίθμηση ενότητας 2.4.3) - 1*

```
con1=m.addConstrs((quicksum(x[i,j,k] for j in V if j!=i if j!=0 for k in K)==1
                   for i in P+D),name="con1")
con2=m.addConstrs((quicksum(x[0,j,k] for j in V if j!=0)==1 for k in K),
                  name="con2")
con3=m.addConstrs((quicksum(x[i,2*n+1,k] for i in V if i!=2*n+1)==1 for k in K),
                  name="con3")
con4=m.addConstrs(((quicksum(x[i,j,k] for i in V if i!=j if i!=2*n+1)-
                    quicksum(x[j,i,k] for i in V if i!=j if i!=0))==0
                    for j in P+D for k in K),name="con4")
con5=m.addConstrs(((x[i,j,k]==1) >> (B[j,k]>=B[i,k]+di[i]+tijk[i,j,k])
                   for i,j in A for k in K),name="con5")
con6=m.addConstrs(((x[i,j,k]==1) >> (Q[j,k]==Q[i,k]+q[j])
                   for i,j in A for k in K),name="con6")
con7_1_1=m.addConstrs((Q[i,k]>=q[i] for i in P for k in K),
                      name="con7_1_1")
con7_1_2=m.addConstrs((Q[i,k]<=Ck[k] for i in P for k in K),
                      name="con7_1_2")
con7_2_1=m.addConstrs((Q[i,k]>=0 for i in [0]+D+[2*n+1] for k in K),
                      name="con7_2_1")
con7_2_2=m.addConstrs((Q[i,k]<=Ck[k]+q[i] for i in [0]+D+[2*n+1] for k in K),
                      name="con7_2_2")
con8=m.addConstrs((quicksum(x[i,j,k] for j in V if j!=i if j!=0)-
                   quicksum(x[n+i,j,k] for j in V if j!=n+i if j!=0)==0
                   for i in P for k in K),name="con8")
con9=m.addConstrs((B[i,k]<=B[i+n,k] for i in P for k in K),name="con9")
con10_1=m.addConstrs((B[i,k]>=ei[i] for i in V for k in K),name="con10_1")
con10_2=m.addConstrs((B[i,k]<=li[i] for i in V for k in K),name="con10_2")
con11=m.addConstrs((B[2*n+1,k]-B[0,k]<=Tk[k] for k in K),name="con11")
```

Τελικώς γίνεται ο ορισμός της αντικειμενικής συνάρτησης.

```
m.setObjective(quicksum(x[i,j,k]*cijk[i,j,k] for i,j in A for k in K),sense=GRB.MINIMIZE)
```

4.2.2 Ο κώδικας κατασκευής και επίλυσης του PDP2

Όπως και το PDP1, το PDP2 χρειάζεται κάποια επιπλέον δεδομένα, τα οποία είναι τα εξής:

- T : Η μέγιστη επιτρεπτή διάρκεια δρομολογίου
- $Qmax$: Η μέγιστη χωρητικότητα των οχημάτων.
- cij : Λεξικό της μορφής $\{(i,j): c_{ij}, \dots\}$ όπου $(i,j) \in A$. Περιέχει πληροφορίες για τα κόστη μετάβασης τόξων κάθε οχήματος.

Πρέπει επίσης να σημειωθεί πως στην λίστα A δεν περιέχονται τα τόξα $(0,i)$ όπου $i \in D$ και τα τόξα $(i, 2n+1)$ όπου $i \in P$.

Αρχικά δημιουργείται ένα κενό αντικείμενο Model.

```
m.setObjective(quicksum(x[i,j]*cij[i,j] for i,j in A),sense=GRB.MINIMIZE)
```

Στη συνέχεια ακολουθεί ο ορισμός της μεταβλητής απόφασης x του προβλήματος, η οποία είναι εξ' ορισμού θετική.

```
x=m.addVars(A,vtype=GRB.BINARY,name='x')
```

Στη συνέχεια γίνεται ο ορισμός των σταθερών περιορισμών του προβλήματος και της αντικειμενικής συνάρτησης.

```
con1=m.addConstrs((quicksum(x[i,j] for i in V if (i,j) in A)==1 for j in P+D),name="con1")
con2=m.addConstrs((quicksum(x[i,j] for j in V if (i,j) in A)==1 for i in P+D),name="con2")
```

```
m.setObjective(quicksum(x[i,j]*cij[i,j] for i,j in A),sense=GRB.MINIMIZE)
```

Οι callback περιορισμοί προστίθενται στο μοντέλο κατά την διαδικασία επίλυσης του προβλήματος μέσω της συνάρτησης *subtourlim*.

```
m._vars = x
m.Params.lazyConstraints = 1
m.optimize(subtourelim)
```

Η συνάρτηση callback του PDP2 πρέπει να εντοπίσει τις διαδρομές στη λύση του προβλήματος, να αναγνωρίσει τον τύπο τους και να εφαρμόσει σε αυτές τους κατάλληλους callback περιορισμούς.

```
def subtourelim(model, where):
    if where == GRB.Callback.MIPSOL:
        # make a list of edges selected in the solution
        vals = model.cbGetSolution(model._vars)
        selected = [(i, j) for i, j in model._vars.keys() if vals[i,j]>0.5]
```

Αρχικά γίνεται ο έλεγχος εάν μία διαδρομή είναι ανοιχτή (περιλαμβάνει τα σημεία έναρξης και τερματισμού) ή κλειστή. Εάν η διαδρομή βρεθεί πως είναι κλειστή τότε εφαρμόζεται σε αυτήν ο περιορισμός (5) του PDP2.

- $route$: Μια λίστα της μορφής $[r_1, r_2, \dots, r_m, r_1]$, η οποία περιέχει τα σημεία της διαδρομής (m σημείων) και την αλληλουχία τους.
- con_arcs : Μια λίστα της μορφής $[(r_1, r_2), (r_1, r_3), \dots, (r_{m-1}, r_m)]$, η οποία περιέχει όλα τα πιθανά τόξα μεταξύ των σημείων της διαδρομής.
- con_value : Ισούται με $max\left(1, \left\lceil \frac{\sum_{i \in route} q_i}{Q_{max}} \right\rceil\right)$

```
model.cbLazy(quicksum(model._vars[i,j] for i,j in con_arcs)<=len(route)-1-con_value)
```

Εάν η διαδρομή βρεθεί πως είναι ανοιχτή στη συνέχει ελέγχεται εάν περιέχει σημεία παράδοσης των οποίων το αντίστοιχο σημείο συλλογής δεν ανήκει στην διαδρομή. Εάν βρεθούν τέτοια σημεία τότε στην διαδρομή εφαρμόζεται ο περιορισμός (4) του PDP2.

- $route$: Μια λίστα της μορφής $[0, r_1, \dots, r_m, 2n+1]$, η οποία περιέχει τα σημεία της διαδρομής και την αλληλουχία τους.
- con_arcs_2 : Μια λίστα της μορφής $[(0, r_1), (0, r_2), \dots, (r_{m-1}, r_m)]$, η οποία περιέχει όλα τα πιθανά τόξα της μεταξύ των σημείων της διαδρομής, εξαιρώντας το σημείο τερματισμού.

```
model.cbLazy(quicksum(model._vars[i,j] for i,j in con_arcs_2)<=len(route)-1-2)
```

Εάν στην διαδρομή δεν έχουν βρεθεί ασύζευκτα σημεία παράδοσης πραγματοποιείται έλεγχος τήρησης των χρονικών περιορισμών της διαδρομής, οι οποίοι είναι:

1. Η επίσκεψη του σημείου συλλογής πρέπει να γίνεται πιο νωρίς από το αντίστοιχο σημείο παράδοσης.
2. Η επίσκεψη σε κάθε σημείο της διαδρομής πρέπει να μπορεί να γίνει εντός των χρονικών παραθύρων του.
3. Η διάρκεια του δρομολογίου δεν πρέπει να είναι μεγαλύτερη της μέγιστης επιτρεπτής διάρκειας δρομολογίου.

Εάν έστω ένας από αυτούς τους περιορισμούς παραβιάζεται τότε για τα σημεία της διαδρομής εφαρμόζεται ο περιορισμός (6) του PDP2.

- $arks\_conroute$: Μια λίστα της μορφής $[(r_1, r_2), (r_2, r_3), \dots, (r_{m-1}, r_m)]$, η οποία περιέχει όλα τα τόξα που προκύπτουν από την διαδρομή $[0, r_1, \dots, r_m, 2n+1]$, εξαίροντας το σημείο έναρξης και τερματισμού.

```
model.cbLazy(quicksum(model._vars[i,j] for i,j in arks_conroute)<=len(arks_conroute)-1)
```

Τελικώς εάν δεν παραβιάζεται κανένας χρονικός περιορισμός ελέγχεται εάν κατά τη διάρκεια της διαδρομής η χωρητικότητα του οχήματος δεν υπερβαίνει την μέγιστη επιτρεπτή. Εάν σε κάποιο σημείο της διαδρομής η χωρητικότητα ξεπεράσει το όριο, τότε στο τμήμα της διαδρομής, από το πρώτο σημείο της μέχρι το σημείο παράβασης, εφαρμόζεται ο περιορισμός (5) του PDP2.

- $route\_segment$: Μια λίστα της μορφής $[r_1, r_2, \dots, r_c]$, η οποία περιέχει τα σημεία της του τμήματος της διαδρομής και την αλληλουχία τους. Η αρχική διαδρομή περιλαμβάνει m σημεία (χωρίς την έναρξη και τερματισμό), ενώ το τμήμα περιέχει c σημεία, όπου $c \leq m$.
- con_arcs : Μια λίστα της μορφής $[(r_1, r_2), (r_1, r_3), \dots, (r_{c-1}, r_c)]$, η οποία περιέχει όλα τα πιθανά τόξα μεταξύ των σημείων της διαδρομής.

- con_value : Ισούται με $max\left(1, \left\lceil \frac{\sum_{i \in route\_segment} q_i}{Q_{max}} \right\rceil\right)$

```
model.cbLazy(quicksum(model._vars[i,j] for i,j in con_arcs)<=nrs-con_value)
```

4.2.3 Δημιουργία προβλήματος και εύρεση της αρχικής λύσης

Τα προβλήματα πάνω στα οποία τρέχουν τα PDP1 και PDP2 στους κώδικες του παραρτήματος Α και κατά την αξιολόγηση επιδόσεων διαφέρουν. Στο παράθεμα οι κώδικες τρέχουν πάνω σε προβλήματα που δημιουργούνται τρόπο παρόμοιο με αυτόν των CVRP1 και CVRP2. Αρχικά ορίζεται η μέγιστη χωρητικότητα οχήματος $Qmax$, ο αριθμός των πελατών n καθώς και μία μέγιστη τιμή των συντεταγμένων του x και y, max_x και max_y. Στην συνέχεια δημιουργούνται τυχαία οι συντεταγμένες (x_i, y_i) του σημείου εκκίνησης και των πελατών, με τα x_i και y_i να παίρνουν τυχαίες τιμές μεταξύ $(0, max\_x)$ και $(0, max\_y)$ αντίστοιχα. Οι ζητήσεις q_i κάθε πελάτη i συλλογής παίρνουν τυχαίες τιμές μεταξύ μιας μέγιστης και ελάχιστης τιμής ζήτησης, ενώ οι πελάτες παράδοσης παίρνουν τις αντίστοιχες αρνητικές τιμές. Η μέγιστη και ελάχιστη τιμή της ζήτησης προκύπτουν από την $Qmax$ και δύο προκαθορισμένες παραμέτρους b_1, b_2 ως:

$$q_{max} = round\left(\frac{Q_{max}}{b_1}\right) \ και \ q_{min} = round\left(\frac{Q_{max}}{b_2}\right) \ όπου \ b_1 < b_2$$

Το κόστος μετάβασης μεταξύ των σημείων ισούται με την γεωμετρική τους απόσταση:

$$d_{ij} \text{ ή } d_{ijk} = c_{ij} \text{ ή } c_{ijk} = \sqrt{\left(x_i - x_j\right)^2 - \left(y_i - y_j\right)^2}$$

Στην συνέχεια ορίζεται μία μέγιστη διάρκεια δρομολογίου T, που καθορίζει πόσες χρονικές μονάδες από την στιγμή 0 όλα τα δρομολόγια πρέπει να τερματίσουν. Βάση της T και των προκαθορισμένων παραμέτρων $w_1 > w_2 \geq w_3 > w_4$ και w_5, w_6 υπολογίζονται οι παράμετροι $eput$, $lput$, $edut$, $ldut$, ο χρόνος φόρτωσης-εκφόρτωσης μονάδας προϊόντος ($load\_time$) και η διάρκεια χρονικού πλαισίου ($window$) ώς:

$$eput = round\left(\frac{T}{w_1}\right), lput = round\left(\frac{T}{w_2}\right), edut = round\left(\frac{T}{w_3}\right), ldut = round\left(\frac{T}{w_4}\right)$$

$$load\_time = \left(\frac{T}{w_5}\right), window = round\left(\frac{T}{w_6}\right)$$

Ο συντομότερος χρόνος εξυπηρέτησης e_i κάθε πελάτη i συλλογής παίρνει τυχαία τιμή από μία ομοιόμορφη κατανομή μεταξύ των τιμών $eput$ και $lput$, ενώ ο αργότερος χρόνος εξυπηρέτησης l_i ισούται με $e_i + window$. Παρόμοια διαδικασία ακολουθείται και με τους πελάτες παράδοσης για $edut$ και $ldut$. Οι χρόνοι εξυπηρέτησης d_i κάθε πελάτη i είναι ίσοι με $load\_time * abs(q_i)$. Για το υπολογισμό των χρόνων μετάβασης μεταξύ σημείων γίνεται ο ορισμός του tt, που εκφράζει πόσες μονάδες χρόνου απαιτούνται για τη διάνυση μίας μονάδας απόστασης, με την χρήση μίας προκαθορισμένης παραμέτρου z_1 ώς:

$$tt = \frac{T}{z_1 * \sqrt{max\_x^2 + max\_y^2}}$$

Οι χρόνοι μετάβασης είναι προκύπτουν ώς : $t_{ij} = tt * d_{ij}$ και $t_{ijk} = tt * d_{ijk}$.

Το PDP2 είναι πλήρως ορισμένο και ως αρχική εφικτή λύση μπορεί να γίνει η υπόθεση πως κάθε ζεύγος συλλογής-παράδοσης εξυπηρετείται από ένα δρομολόγιο. Το PDP1 για να οριστεί πλήρως χρειάζεται έναν αριθμό οχημάτων NV, ο οποίος εάν cpr ο εκτιμώμενος αριθμός ζευγών πελατών που εξυπηρετείται σε κάθε δρομολόγιο υπολογίζεται ώς:

$$NV = \left\lceil \frac{n}{cpr} \right\rceil$$

Το πρόβλημα με το PDP1 είναι πως σε αντίθεση με το PDP2 η εύρεση μίας εύκολης αρχικής εφικτής λύσης είναι δύσκολη, λόγω της έλλειψης εύκολων στην εφαρμογή ευρετικών αλγόριθμων. Λαμβάνοντας την δυσκολία αυτή υπόψιν, για να γίνει δίκαια σύγκριση της επίδοσης των PDP1 και PDP2 τα προβλήματα πάνω στα οποία θα τρέξουν οι κώδικες δημιουργούνται με ελαφρώς διαφορετικό τρόπο. Αρχικά δημιουργούνται τυχαίες εφικτές διαδρομές διάρκειας μικρότερης από T. Σε κάθε σημείο i της διαδρομής υπολογίζεται ο

χρόνος άφιξης a_i και τα e_i και l_i παίρνουν τιμές τέτοιες ώστε $e_i < a_i < l_i$ και $l_i - e_i > d_i$. Ως αρχική εφικτή λύση και στα δύο προβλήματα χρησιμοποιούνται οι ήδη γνωστές εφικτές διαδρομές.

4.2.4 Αποτελέσματα επιδόσεων PDP

Σε αυτή την ενότητα γίνεται παρουσίαση των επιδόσεων των CVRP1 και CVRP2 σε προβλήματα μικρού (5,10 ζεύγη πελάτων), μεσαίου (15,20 ζεύγη πελάτων) και μεγάλου (25 ζεύγη πελάτων) μεγέθους για δύο διαφορετικές ταχύτητες (z_1=4 και z_1=8) με μέγιστη χωρητικότητα οχημάτων Q_{max}=200 και με τη ζήτηση να κυμαίνεται μεταξύ 1~10 δεκάδες προϊόντων. Οι μέγιστες τιμές συντεταγμένων $max\_x = 500$ και $max\_y = 500$. Οι κώδικες δοκιμάζονται σε τρείς παραλλαγές του προβλήματος. Η πρώτη παραλλαγή συμβολίζεται ώς $PDPstw$ και έχει πολύ αυστηρά χρονικά περιθώρια εξυπηρέτησης των πελατών ($l_i - e_i \approx d_i$). Η δεύτερη παραλλαγή συμβολίζεται ώς $PDPtw$ και έχει χαλαρωμένα χρονικά περιθώρια εξυπηρέτησης των πελατών ($l_i - e_i \approx T/4 \sim T/2$). Η τρίτη παραλλαγή συμβολίζεται ώς $PDPntw$ και δεν περιέχει χρονικά περιθώρια εξυπηρέτησης των πελατών ($l_i - e_i = T$). Ό μέγιστος επιτρεπτός χρόνος εκτέλεσης κώδικα είναι 10 λεπτά (600 sec).

Πίνακας 4-6: Αποτελέσματα επιδόσεων PDP σε προβλήματα 5 ζευγών.

5 ζεύγη	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$PDPstw$				
PDP1 z_1=4	270	0,04	0%	2640,49
PDP2 z_1=4	100	16,9	0%	2640,49
PDP1 z_1=8	270	0,06	0%	2640,49
PDP2 z_1=8	100	16,8	0%	2640,49
$PDPtw$				
PDP1 z_1=4	270	0,3	0%	2296,9
PDP2 z_1=4	100	2,53	0%	2296,9
PDP1 z_1=8	270	0,27	0%	2296,9
PDP2 z_1=8	100	2,36	0%	2296,9
$PDPntw$				
PDP1 z_1=4	270	4,8	0%	2296,9
PDP2 z_1=4	100	2,23	0%	2296,9
PDP1 z_1=8	270	0,44	0%	1687,47
PDP2 z_1=8	100	0,03	0%	1687,47

Πίνακας 4-7: Αποτελέσματα επιδόσεων PDP σε προβλήματα 10 ζευγών.

10 ζεύγη	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$PDPstw$				
PDP1 z_1=4	1860	0,18	0%	4787,45
PDP2 z_1=4	400	600	49%	4903,25
PDP1 z_1=8	1860	0,27	0%	4309,88
PDP2 z_1=8	400	600	49%	4903,25
$PDPtw$				
PDP1 z_1=4	1860	79,33	0%	4369,56
PDP2 z_1=4	400	600	49%	4903,25
PDP1 z_1=8	1860	158,45	0%	3992,7
PDP2 z_1=8	400	600	49%	4903,25
$PDPntw$				
PDP1 z_1=4	1860	600	39%	3783,32
PDP2 z_1=4	400	600	39%	3961,21
PDP1 z_1=8	1860	600	26%	3169,57
PDP2 z_1=8	400	600	38%	3813,22

Πίνακας 4-8: Αποτελέσματα επιδόσεων PDP σε προβλήματα 15 ζευγών.

15 ζεύγη	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$PDPstw$				
PDP1 z_1=4	5970	2,37	0%	6728,98
PDP2 z_1=4	900	600	61,65%	6887,03
PDP1 z_1=8	5970	1,98	0,00%	5907,69
PDP2 z_1=8	900	600	61,65%	6887,03
$PDPtw$				
PDP1 z_1=4	5970	600	45,91%	6158,93
PDP2 z_1=4	900	600	60,84%	6887,03
PDP1 z_1=8	5970	600	39,58%	5222,53
PDP2 z_1=8	900	600	60,85%	6887,03
$PDPntw$				
PDP1 z_1=4	5970	600	57,41%	5723,43
PDP2 z_1=4	900	600	60,77%	6887,03
PDP1 z_1=8	5970	600	54,59%	5353,47
PDP2 z_1=8	900	600	56,92%	6256,21

Πίνακας 4-9: Αποτελέσματα επιδόσεων PDP σε προβλήματα 20 ζευγών.

20 ζεύγη	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$PDPstw$				
PDP1 z_1=4	13800	67,95	0%	9135,79
PDP2 z_1=4	1600	600	67,28%	9552,48
PDP1 z_1=8	13800	600	12,47%	8015,16
PDP2 z_1=8	1600	600	67,28%	9552,48
$PDPtw$				
PDP1 z_1=4	13800	600	54,44%	8871,88
PDP2 z_1=4	1600	600	67,04%	9552,48
PDP1 z_1=8	13800	600	52,60%	8081,71
PDP2 z_1=8	1600	600	67,05%	9552,48
$PDPntw$				
PDP1 z_1=4	13800	600	66,86%	8512,83
PDP2 z_1=4	1600	600	67,22%	9552,48
PDP1 z_1=8	13800	600	62,61%	7657,65
PDP2 z_1=8	1600	600	62,44%	8332,83

Πίνακας 4-10: Αποτελέσματα επιδόσεων PDP σε προβλήματα 20 ζευγών.

25 ζεύγη	Μεταβλητές	Χρόνος (sec)	Gap	Αντικειμενική
$PDPstw$				
PDP1 z_1=4	26550	353,13	0%	12069,51
PDP2 z_1=4	2500	600	72,56%	12566,32
PDP1 z_1=8	26550	600	25,24%	10656,08
PDP2 z_1=8	2500	600	72,56%	12566,32
$PDPtw$				
PDP1 z_1=4	26550	600	64,93%	11892,7
PDP2 z_1=4	2500	600	72,65%	12566,32
PDP1 z_1=8	26550	600	63,88%	10976,78
PDP2 z_1=8	2500	600	72,63%	12566,32
$PDPntw$				
PDP1 z_1=4	26550	600	73,43%	11725,25
PDP2 z_1=4	2500	600	72,49%	12566,32
PDP1 z_1=8	26550	600	73,30%	9995,69
PDP2 z_1=8	2500	600	70,27%	11637,66

Τα συμπεράσματα που προκύπτουν βάσει των επιδόσεων των PDP1 και PDP2 είναι τα εξής:

- Όσο πιο αυστηρά είναι τα χρονικά παράθυρα επίσκεψης κάθε πελάτη η απόδοση του PDP1 βελτιώνεται, ενώ αντιστρόφως όσο πιο χαλαρά είναι βελτιώνεται η απόδοση του PDP2.

- Η αύξηση της ταχύτητας αυξάνει σε κάποιον μικρό βαθμό την απόδοση και στα δύο μοντέλα. Η συνεισφορά της φαίνεται περισσότερο στην βελτίωση της τιμής της αντικειμενικής.

Από τις τιμές των δοκιμών φαίνεται πως το PDP1 υπερισχύει του PDP2 σχεδόν σε όλες τις περιπτώσεις, όμως αυτό δεν σημαίνει πως το μοντέλο του PDP1 είναι καλύτερο του PDP2. Όπως αναφέρθηκε, το PDP2 μπορεί να αρχίσει άμεσα την επίλυση σε κάθε πρόβλημα με την αρχική εφικτή λύση πως κάθε ζεύγος πελατών εξυπηρετείται από ένα δρομολόγιο. Από την άλλη το PDP1, χωρίς την βοήθεια κάποιου ευρετικού αλγόριθμου, θα δυσκολευτεί πολύ ακόμα και να βρει μία εφικτή λύση. Επιπλέον πρέπει να σημειωθεί πως η απόδοση του μοντέλου του PDP2 της *Ενότητας 2.4.3* μπορεί να βελτιωθεί δραστικά με την χρήση επιπλέον περιορισμών και έγκυρων ανισοτήτων, που προτείνονται από τους Ropke et al. [14].

4.3 Κατασκευή και αξιολόγηση του IRP

Στην *Ενότητα 2.5.3* παρουσιάζεται η μοντελοποίηση του κλασικού IRP με χρήση τεσσάρων μεταβλητών απόφασης, τεσσάρων δεικτών και callback περιορισμού. Τα δεδομένα εισόδου και η μορφή τους είναι:

- n : Ο αριθμός των πελατών.
- p : Ο αριθμός των χρονικών περιόδων του προβλήματος.
- k : Ο αριθμός των διαθέσιμων οχημάτων.
- N : Λίστα της μορφής $[1,2,\dots,n]$. Εκφράζει το σύνολο των πελατών.
- V : Λίστα της μορφής $[0,1,2,\dots,n]$. Εκφράζει το σύνολο όλων των κόμβων.
- K : Λίστα της μορφής $[1,2,\dots,k]$. Εκφράζει το σύνολο των οχημάτων.
- T : Λίστα της μορφής $[1,2,\dots,p]$. Εκφράζει το σύνολο των χρονικών περιόδων.
- $list\_of\_indexes\_1$: Λίστα της μορφής $[(i,j,v,t),\dots]$ για $i<j$ όπου $i \in V$, $j \in V$, $v \in K$ και $t \in T$. Περιέχει όλους τους δυνατούς συνδυασμούς τόξων, οχημάτων και περιόδων.
- $list\_of\_indexes\_2$: Λίστα της μορφής $[(i,v,t),\dots]$ όπου $i \in V$, $v \in K$ και $t \in T$. Περιέχει όλους τους δυνατούς συνδυασμούς κόμβων, οχημάτων και περιόδων.
- $list\_of\_indexes\_3$: Λίστα της μορφής $[(i,t),\dots]$ όπου $i \in V$ και $t \in T$. Περιέχει όλους τους δυνατούς συνδυασμούς κόμβων και περιόδων.

- Qk : Λεξικό της μορφής $\{v: Q_v, \dots\}$ όπου $v \in K$. Περιέχει πληροφορίες για την μέγιστη χωρητικότητα κάθε οχήματος.
- dti : Λεξικό με στοιχεία λίστες, της μορφής $\{i: [d_{1,i}, \dots, d_{p,i}], \dots\}$ όπου $i \in N$. Για κάθε πελάτη περιέχει λίστα με τις ζητήσεις του κάθε περίοδο.
- Ci : Λεξικό της μορφής $\{i: C_i, \dots\}$ όπου $i \in V$. Περιέχει πληροφορίες για την μέγιστη χωρητικότητα των αποθηκών των πελατών και προμηθευτή.
- $I0i$: Λεξικό της μορφής $\{i: I_{0,i}, \dots\}$ όπου $i \in V$. Περιέχει πληροφορίες για το αρχικό επίπεδο αποθέματος του προμηθευτή και των πελατών.
- rt : Λεξικό της μορφής $\{t: r_t, \dots\}$ όπου $t \in T$. Περιέχει πληροφορίες για την διαθέσιμη ποσότητα προϊόντος προς αποστολή κάθε περίοδο.
- c : Λεξικό της μορφής $\{(i,j): c_{ij}, \dots\}$ για $i < j$ όπου $i \in V$ και $j \in V$. Περιέχει πληροφορίες για το κόστος μετάβασης τόξου.
- hi : Λεξικό της μορφής $\{i: h_i, \dots\}$ όπου $i \in V$. Περιέχει πληροφορίες για το κόστος διατήρησης αποθέματος (προμηθευτή και πελατών) μίας μονάδας προϊόντος για μία περίοδο.

4.3.1 Ο κώδικας κατασκευής και επίλυσης IRP

Αρχικά δημιουργείται ένα κενό αντικείμενο Model.

```
m=Model('IRP')
```

Ακολουθεί ο ορισμός των μεταβλητών απόφασης x, y, q και I του προβλήματος.

```
x=m.addVars(list_of_indexes_1,vtype=GRB.INTEGER,ub=2,name='x')
y=m.addVars(list_of_indexes_2,vtype=GRB.BINARY,name='y')
q=m.addVars(list_of_indexes_2,vtype=GRB.INTEGER,name='q')
I=m.addVars(list_of_indexes_3,vtype=GRB.INTEGER,name='I')
```

Στη συνέχεια γίνεται ο ορισμός των σταθερών περιορισμών του προβλήματος και της αντικειμενικής συνάρτησης.

```
con0=m.addConstrs((I[i,0]==I0i[i] for i in V),name='con0')
con1=m.addConstrs((I[0,t-1]-I[0,t]+rt[t]-quicksum(q[i,vec,t] for i in N for vec in K)==0 for t in T),name='con1')
con2=m.addConstrs((I[i,t-1]-I[i,t]+quicksum(q[i,vec,t] for vec in K)-dti[i][t-1]==0 for i in N for t in T),name='con2')
#con3=m.addConstrs((I[i,t]>=0 for i in V for t in T),name='con3')
con4=m.addConstrs((I[i,t]<=Ci[i] for i in V for t in T),name='con4')
con5=m.addConstrs((quicksum(q[i,vec,t] for vec in K)<=Ci[i]-I[i,t-1] for i in N for t in T),name='con5')
con6=m.addConstrs((q[i,vec,t]<=Ci[i]*y[i,vec,t] for i in N for vec in K for t in T),name='con6')
con7=m.addConstrs((quicksum(q[i,vec,t] for i in N)<=Qk[vec]*y[0,vec,t] for vec in K for t in T),name='con7')
con8=m.addConstrs((quicksum(x[i,j,vec,t] for j in V if i<j)+quicksum(x[j,i,vec,t] for j in V if j<i)==
                   2*y[i,vec,t] for i in V for vec in K for t in T),name='con8')
con9=m.addConstrs(x[i,j,vec,t]<=1 for (i,j,vec,t) in list_of_indexes_1 if i!=0 if j!=0)
```

```
m.setObjective(quicksum(hi[v]*I[v,t] for (v,t) in list_of_indexes_3)+
               quicksum(c[i,j]*x[i,j,vec,t] for (i,j,vec,t) in list_of_indexes_1),GRB.MINIMIZE)
```

Οι callback περιορισμοί προστίθενται στο μοντέλο κατά την διαδικασία επίλυσης του προβλήματος μέσω της συνάρτησης *subtourlim*.

```
m._x = x
m._y= y
m.Params.lazyConstraints = 1
m.optimize(subtourelim)
```

Η συνάρτηση callback του IRP πρέπει εντοπίζει τις διαδρομές στη λύση του προβλήματος, αναγνωρίζει ποιες από αυτές είναι κλειστές (δεν επικοινωνούν με το σημείο αφετηρίας-τερματισμού) και εφαρμόζει τον περιορισμό callback (9) του IRP στα σημεία των διαδρομών αυτών.

```
def subtourelim(model, where):
    if where == GRB.Callback.MIPSOL:
        x_vals = model.cbGetSolution(model._x)
        y_vals = model.cbGetSolution(model._y)
```

- s : Μία λίστα της μορφής $[k_1, k_2, \dots, k_m, k_1]$, η οποία περιέχει τα σημεία μίας κλειστής διαδρομής (m σημείων) και την αλληλουχία τους.
- con_arcs : Μια λίστα της μορφής $[(i,j), \dots]$ για $i < j$ όπου $i \in s$ και $j \in s$.
- vec : Το όχημα που πραγματοποιεί την συγκεκριμένη διαδρομή.
- t : Η περίοδος που πραγματοποιείται η συγκεκριμένη διαδρομή.

```
model.cbLazy(quicksum(model._x[i,j,vec,t] for i,j in con_arks)
                        <=quicksum(model._y[i,vec,t] for i in s)-y[s[0],vec,t])
```

4.3.2 Δημιουργία τυχαίου προβλήματος και εύρεση αρχικής εφικτής λύσης

Τα προβλήματα πάνω στα οποία τρέχει ο κώδικας του IRP, δημιουργούνται τεχνητά. Αρχικά ορίζεται ο αριθμός των χρονικών περιόδων του προβλήματος p, ο αριθμός των πελατών n καθώς και μία μέγιστη τιμή των συντεταγμένων του x και y, max_x και max_y. Στην συνέχεια δημιουργούνται τυχαία οι συντεταγμένες (x_i, y_i) του σημείου εκκίνησης και των πελατών, με τα x_i και y_i να παίρνουν τυχαίες τιμές μεταξύ $(0, max\_x)$ και $(0, max\_y)$ αντίστοιχα. Γίνεται η παραδοχή ομοιόμορφου στόλου οχημάτων και ορίζεται η μέγιστη

χωρητικότητα οχήματος cap. Κάθε περίοδο οι ζητήσεις dt_i κάθε πελάτη i παίρνουν τυχαίες τιμές μεταξύ μιας μέγιστης ($dmax$) και ελάχιστης ($dmin$) τιμής ζήτησης, οι οποίες αποτελούν συνάρτηση της cap και δύο προκαθορισμένες παραμέτρους b_1, b_2 ως:

$$d_{max} = round\left(\frac{cap}{b_1}\right) \ \kappa\alpha\iota \ d_{min} = round\left(\frac{cap}{b_2}\right) \ \acute{o}\pi o\upsilon \ b_1 < b_2$$

Γνωρίζοντας τα d_{min} ,d_{max} ο αριθμός των διαθέσιμων οχημάτων k υπολογίζεται από την παρακάτω σχέση:

$$k = \left\lceil \frac{n * (d_{max} + d_{min})}{2 * cap} \right\rceil$$

Το συνολικό κόστος του IRP υπολογίζεται από το άθροισμα του συνολικού κόστους μετάβασης μεταξύ των τόξων και του συνολικού κόστους διατήρησης αποθέματος. Αρχικά ορίζεται το κόστος διάνυσης μίας μονάδας απόστασης $move_{cost}$ και βάση αυτού υπολογίζεται το κόστος διατήρησης αποθέματος αναφοράς hm_{cost} και το κόστος μετάβασης c_{ij} του τόξου (i, j) ώς:

$$hm_{cost} = \frac{2 * k * move_{cost} * \sqrt{\max\_x^2 + \max\_y^2}}{n * d_{max}}$$

$$c_{ij} = move_{cost} * \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}$$

Οι πελάτες χωρίζονται σε τρείς υποκατηγορίες και η πιθανότητα ένας πελάτης να ανήκει σε κάποια από αυτές ορίζεται από μία διακριτή κατανομή με πιθανότητες p_{t1}, p_{t2} και p_{t3} για κάθε υποκατηγορία αντίστοιχα. Οι τύποι των πελατών ανάλογα την υποκατηγορία στη οποία ανήκουν είναι:

- Τύπος 1 : Πελάτες με μεγάλο κόστος αποθήκευσης και μικρό αποθηκευτικό χώρο.
- Τύπος 2 : Πελάτες με μεσαίο κόστος αποθήκευσης και επαρκή αποθηκευτικό χώρο.
- Τύπος 3 : Πελάτες με μικρό κόστος αποθήκευσης και μεγάλο αποθηκευτικό χώρο.

Το κόστος διατήρησης αποθέματος h_i κάθε πελάτη, ανάλογα την υποκατηγορίας j στην οποία ανήκει, παίρνει μία τυχαία τιμή μεταξύ ενός $h_{min,j}$ και $h_{max,j}$ τα οποία με βάση κάποιες προκαθορισμένες παραμέτρους $\alpha_{1,j}$ και $\alpha_{2,j}$ υπολογίζονται ως:

$$h_{min,j} = \alpha_{1,j} * hm_{cost} \ , h_{max,j} = \alpha_{2,j} * hm_{cost} \ , \alpha_{1,3} < \alpha_{2,3} \leq \alpha_{1,2} < \alpha_{2,2} \leq \alpha_{1,1} < \alpha_{2,1}$$

Ανάλογα την j του κάθε πελάτη, το μέγιστο επιτρεπτό μέγεθος αποθέματος C_i με βάση τις προκαθορισμένες παραμέτρους $\alpha_{3,j}$ υπολογίζεται ώς :

$$C_i = round(\alpha_{3,j} * d_{max}) , \qquad \alpha_{3,1} < \alpha_{3,2} < \alpha_{3,3}$$

Όσον αφορά τον προμηθευτή γίνεται η παραδοχή πως έχει κόστος διατήρησης αποθέματος $h_{sup} = 0$ και μέγιστη χωρητικότητα αποθέματος $C_{sup} = n * p * d_{max}$. Η διαθέσιμη ποσότητα προϊόντων προς παράδοση κάθε περίοδο r_t παίρνει τυχαίες τιμές μεταξύ των $r_{min} = n * d_{max}$ και $r_{max} = (1{,}5 * n * d_{max})$.

Τελικώς ορίζονται τα αρχικά επίπεδα κάθε πελάτη I_{0i} τα οποία παίρνουν τυχαίες τιμές μεταξύ του $I_{0,min} = 0$ και $I_{0,max} = d_{min} + d_{max}$, ενώ ο προμηθευτής έχει αρχικό επίπεδο αποθέματος $I_{0,sup} = 0$.

Η εύρεση μία αρχικής εφικτής λύσης για το IRP γίνεται λύνοντας για κάθε περίοδο του προβλήματος το CVRP πρόβλημα για τις αντίστοιχες ζητήσεις, έχοντας λάβει υπόψιν τα αρχικά επίπεδα αποθέματος κάθε πελάτη.

4.3.3 Αποτελέσματα επιδόσεων IRP

Σε αυτή την ενότητα γίνεται παρουσίαση των επιδόσεων του IRP χρονικής διάρκειάς 3 ή 5 περιόδων για προβλήματα 10,20 και 30 πελατών. Οι μέγιστες τιμές συντεταγμένων $max\_x = max\_y = 1000$, η μέγιστη χωρητικότητα οχημάτων $cap = 500$, το κόστος διάνυσης μίας μονάδας απόστασης $move_{cost} = 2$ και οι τιμές των παραμέτρων $b_1 = 5$ και $b_2 = 50$. Ανάλογα τον τύπο του κάθε πελάτη οι τιμές των παραμέτρων α είναι:

Πίνακας 4-11: Τιμές των παραμέτρων α για κάθε τύπο πελάτη.

	$\alpha_{1,i}$	$\alpha_{2,i}$	$\alpha_{3,i}$
Τύπος 1	1,2	1,8	1+d_{min}
Τύπος 2	0,8	1,2	$p/2$
Τύπος 3	0,2	0,8	p

Κάθε συνδυασμός περιόδων και πελατών, επιπλέον θα αξιολογηθεί σε τρία προβλήματα με διαφορετικές πιθανότητες τύπων πελατών, οι οποίες είναι:

Πίνακας 4-12 : Πιθανότητες της διακριτής κατανομής τύπου πελατών κάθε προβλήματος.

	p_{t1}	p_{t2}	p_{t3}
Πρόβλημα 1	0,5	0,35	0,15
Πρόβλημα 2	0,2	0,6	0,2
Πρόβλημα 3	0,15	0,35	0,5

Πέρα από τις επιδόσεις του IRP, επίσης γίνεται σύγκριση μεταξύ της λύσης του IRP και της λύσης που προκύπτει από την επίλυση του CVRP κάθε περίοδο. Το ποσοστό βελτίωσης της λύσης υπολογίζεται ώς:

$$improvment\ \% = \frac{CVRP_{objective} - IRP_{objective}}{CVRP_{objective}}$$

Ό μέγιστος επιτρεπτός χρόνος εκτέλεσης κώδικα του CVRP ανά περίοδο είναι 20 δευτερόλεπτα, ενώ του IRP είναι 10 λεπτά (600 δευτερόλεπτα).

Πίνακας 4-13: Αποτελέσματα απόδοσης IRP για 10 πελάτες.

10 πελάτες	Χρόνος (sec)	Gap	IRP Obj	CVRP Obj	Improvement%
5 περίοδοι 2 οχήματα (836 μεταβλητές, 841 σταθεροί περιορισμοί)					
Πρόβλημα 1	7,19	0%	38487,32	42824,9	10,13
Πρόβλημα 2	5,06	0%	38339,24	42676,9	10,16
Πρόβλημα 3	10,05	0%	35090,18	39237	10,57
3 περίοδοι 2 οχήματα (506 μεταβλητές, 509 σταθεροί περιορισμοί)					
Πρόβλημα 1	0,35	0%	28604,79	32456,8	11,87
Πρόβλημα 2	0,37	0%	27888,11	31740,1	12,14
Πρόβλημα 3	0,36	0%	22930,44	28001,5	18,11

Πίνακας 4-14: Αποτελέσματα απόδοσης IRP για 10 πελάτες.

20 πελάτες	Χρόνος (sec)	Gap	IRP Obj	CVRP Obj	Improvement%
5 περίοδοι 3 οχήματα (3906 μεταβλητές, 3811 σταθεροί περιορισμοί)					
Πρόβλημα 1	600	10,16%	54264,79	57857,5	6,21
Πρόβλημα 2	600	9,91%	53819,38	57470,4	6,35
Πρόβλημα 3	600	11,28%	49237,25	54416	9,52
3 περίοδοι 3 οχήματα (2352 μεταβλητές, 2295 σταθεροί περιορισμοί)					
Πρόβλημα 1	236,95	0,00%	35849,51	40240,4	10,91
Πρόβλημα 2	260,71	0%	34945,26	39336,2	11,16
Πρόβλημα 3	514,33	0,00%	31019,42	36194,7	14,30

Πίνακας 4-15: Αποτελέσματα απόδοσης IRP για 10 πελάτες.

30 πελάτες	Χρόνος (sec)	Gap	IRP Obj	CVRP Obj	Improvement%
5 περίοδοι 4 οχήματα (10726 μεταβλητές, 10431 σταθεροί περιορισμοί)					
Πρόβλημα 1	600	15,04%	68889,47	71174,2	3,21
Πρόβλημα 2	600	13,55%	66118,39	69540,2	4,92
Πρόβλημα 3	600	17,40%	61084,18	64841	5,79
3 περίοδοι 4 οχήματα (6448 μεταβλητές, 6271 σταθεροί περιορισμοί)					
Πρόβλημα 1	600	10,18%	46088,14	47844,7	3,67
Πρόβλημα 2	600	10,57%	44339,17	46048,3	3,71
Πρόβλημα 3	600	12,05%	39791,93	42372,6	6,09

Τα συμπεράσματα που προκύπτουν βάσει των επιδόσεων του IRP είναι τα εξής:

- Πελάτες με μεγάλα όρια αποθέματος και μικρό κόστος διατήρησης δυσκολεύουν την επίλυση του προβλήματος, ενώ πελάτες με μικρά όρια αποθέματος και υψηλό κόστος διατήρησης την διευκολύνουν.
- Η εφαρμογή IRP σε προβλήματα με μεγάλα όρια αποθέματος και μικρά κόστη αποθήκευσης επιφέρει σημαντική μείωση του συνολικού κόστους σε σχέση με την στρατηγική εφαρμογής CVRP ανά περίοδο.

Κεφάλαιο 5. Συμπεράσματα - Προτάσεις

5.1 Σύνοψη αποτελεσμάτων και συμπεράσματα

Ο πρώτος στόχος της τρέχουσας διπλωματικής ήταν ο εντοπισμός και η δημιουργία ενός «πορτφολίου» μοντέλων και κωδίκων των βασικότερων προβλημάτων της εφοδιαστικής αλυσίδας, κάτι που λαμβάνοντας υπόψιν τους περιορισμούς χώρου και χρόνου πραγματοποιήθηκε επιτυχώς. Ο δεύτερος στόχος ήταν η ανάδειξη των δυνατοτήτων του solver της GUROBI και η εισαγωγή στην Python διεπαφή του. Χάρις στις απαιτήσεις των εξεταζόμενων προβλημάτων δόθηκε η ευκαιρία προβολής και πιο προχωρημένων δυνατοτήτων, όπως η χρήση περιορισμών callback.

Αν και η μέτρηση της επίδοσης των μοντέλων δεν ήταν στόχος τρέχουσας εργασίας, οι δοκιμές που πραγματοποιήθηκαν έδωσαν χρήσιμα συμπεράσματα για την συμπεριφορά των μοντέλων ανάλογα το μέγεθος και τον τύπο των προβλημάτων. Η δυσκολία εύρεσης της βέλτιστης λύσης σε πολλά προβλήματα οφείλεται, εν μέρει, στις περιορισμένες δυνατότητες του υπολογιστικού συστήματος και στους μικρούς χρόνους εκτέλεσης κώδικα αλλά φανερώνει επίσης και τις αδυναμίες της επίλυσης προβλημάτων βελτιστοποίησης μόνο με ακριβείς μεθόδους. Η χρήση ευρετικών (*heuristic*) ή μετευρετικών (*metaheuristic*) διαδικασιών, η τμηματοποίηση (clustering) των μεγάλων προβλημάτων και η χρήση *bilevel optimization* σε συνδυασμό με τη χρήση solver σε μοντέλα μαθηματικού προγραμματισμού διευκολύνει και επιταχύνει την επίλυση δύσκολων προβλημάτων.

5.2 Προτάσεις για μελλοντική εργασία

Η απόδοση των μοντέλων θα μπορούσε να βελτιωθεί σημαντικά με την εύρεση και χρήση αλγορίθμων υπολογισμού καλών αρχικών εφικτών λύσεων, κάτι στο οποίο δεν δόθηκε ιδιαίτερο βάρος. Στην εργασία επίσης έγινε περιγραφή διάφορων παραλλαγών του κάθε προβλήματος η μοντελοποίηση των οποίων θα μπορούσε να επιτευχθεί με κάποιες τροποποιήσεις στα μοντέλα και κώδικες που παρουσιάζονται.

Ένα σημαντικό πρόβλημα της εφοδιαστικής αλυσίδας, το οποίο δεν περιέχεται στην τρέχουσα διπλωματική, είναι το πρόβλημα δρομολόγησης παραγωγής ή PRP (Production Routing Problem) που αποτελεί γενίκευση του IRP. Το PRP έρχεται να συνδυάσει δύο ήδη γνωστά προβλήματα, το πρόβλημα μεγέθους παρτίδας (Lot-Sizing Problem ή LSP) και το

πρόβλημα δρομολόγησης (VRP), με σκοπό την εύρεση των βέλτιστων αποφάσεων λαμβάνοντας υπόψιν όλα τα στάδια της εφοδιαστικής αλυσίδας.

Τελικώς, σημαντική θα ήταν η πρακτική εφαρμογή των μοντέλων σε αληθινές εφαρμογές, επιτρέποντας έτσι την αξιολόγηση της χρησιμότητας τους και τον εντοπισμό των αδυναμιών τους.

Βιβλιογραφία

[1] Alan Rushton, Phil Croucher, Peter Baker, Introduction to logistics and distribution, *The handbook of logistics and distribution management,* Fifth Edition*,* Kogan Page Limited, UK, 3-15, 2014

[2] Toth, and Daniele Vigo. 2014. *Vehicle Routing: Problems, Methods, and Applications*, Second Edition. Society for Industrial and Applied Mathematics, USA.

[3] Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). *Integer Programming Formulation of Traveling Salesman Problems*. Journal of the ACM, 7(4), 326–329.

[4] Roberti, R., Toth, P. Models and algorithms for the *Asymmetric Traveling Salesman Problem: an experimental comparison*, EURO J Transp Logist 1, 113–133 (2012).

[5] Toth, P., Vigo, D. (2002). *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA.

[6] G. B. DANTZIG AND J. H. RAMSER, *The truck dispatching problem*, Management Science, 6 (1959), pp. 80–91.

[7] Laporte, G. and Y. Nobert, (1987), *Exact Algorithms for the Vehicle Routing Problem*, Annals of Discrete Mathematics, 31:147-184.

[8] Toth, and Daniele Vigo. 2014. *Vehicle Routing: Problems, Methods, and Applications*, Second Edition. Society for Industrial and Applied Mathematics, USA.

[9] B. L. GOLDEN, T. L. MAGNANTI, AND H. Q. NGUYEN, *Implementing vehicle routing algorithms*, Networks, 7 (1977), pp. 113–148

[10] Laporte, G., Mercure, H., & Nobert, Y. (1986). *An exact algorithm for the asymmetrical capacitated vehicle routing problem*. Networks, 16(1), 33–46.

[11] Kalantari, B., Hill, A. V., & Arora, S. R. (1985). *An algorithm for the traveling salesman problem with pickup and delivery customers*. European Journal of Operational Research, 22(3), 377–386.

[12] Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F (2002) *VRP with time windows*. In: Toth P, Vigo D (eds.) The Vehicle Routing Problem. SIAM, Philadelphia, PA, SIAM Monographs on Discrete Mathematics and Applications, vol. 9, 175–193

[13] Cordeau J-F (2006). *A branch-and-cut algorithm for the dial-a-ride problem*. Operations Research 54:573–586

[14] Ropke, Stefan & Cordeau, Jean-François & Laporte, Gilbert. (2007). *Models and branch-and-cut algorithms for pickup and delivery problems with time windows*. Networks. 49. 258-272.

[15] Bell, Dalberto, Fisher, Greenfield, Jaikumar, Kedia, Mack, Prutzman. (1983). *Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer*. Interfaces, 13(6), 4-23.

[16] Coelho, L. C., & Laporte, G. (2013). *The exact solution of several classes of inventory-routing problems*. Computers & Operations Research, 40(2), 558–565.

[17] L. V. Kantorovich, (1960) *Mathematical Methods of Organizing and Planning Production. Management Science* 6(4):366-422. http://dx.doi.org/10.1287/mnsc.6.4.366

[18] R. E. Bixby, *A brief history of linear and mixed-integer programming computation*, Documenta Mathematica, 2012, pp. 107-121

[19] Robert Bixby PhD, (2/12/2020), Mathematical Optimization: Past, Present, and Future (Part 1), [https://www.gurobi.com/resource/mathematical-optimization-past-present-and-future-part-1/, ανακτήθηκε 23/10/2020]

[20] Karmarkar, N, (1984). *A new polynomial-time algorithm for linear programming*, Combinatorica, 4, 373–395
[21] Robert Bixby PhD, (3/9/2020), *Mathematical Optimization: Past, Present, and Future (Part 2)*, [https://www.gurobi.com/resource/mathematical-optimization-past-present-and-future-part-2/, ανακτήθηκε 23/10/2020]
[22] About GUROBI, [https://www.gurobi.com/company/about-gurobi/ ανακτήθηκε 5/11/2020]
[23] Rimmi Anand, Divya Aggarwal & Vijay Kumar (2017) *A comparative analysis of optimization solvers*, Journal of Statistics and Management Systems, 20:4, 623-635
[24] (5/5/2010*), Gurobi 3.0 Again Wins on Public Performance Benchmarks vs. Competition*, [https://www.gurobi.com/news/gurobi-3-0-wins-performance-benchmarks/ , ανακτήθηκε 26/10/2020]
[25] Hans Mittelman, *BENCHMARKS FOR OPTIMIZATION SOFTWARE*, [http://plato.asu.edu/bench.html , ανακτήθηκε 26/10/2020]
[26] GUROBI OPTIMIZATION, [Gurobi Optimization], (15/7/2020), Ask the Experts: Geek Out On MIP with Gurobi's R&D Leaders [Video], [https://www.youtube.com/watch?v=i9c-j-YwkrM, ανακτήθηκε 23/10/2020]
[27] Tobias Achterberg, Roland Wunderling, (1/2013), *Mixed Integer Programming: Analyzing 12 Years of Progress*, ResearchGate
[28] Gurobi Optimizer, [https://www.gurobi.com/products/gurobi-optimizer/ανακτήθηκε 27/10/2020]

Παράρτημα Α - Κώδικες

Α1 Κώδικας απλού CVRP (CVRP 1)

1) Εισαγωγή Βιβλιοθηκών

In [1]:
```
import numpy as np
from gurobipy import *
import matplotlib.pyplot as plt
```

2) Εισαγόμενα μεγέθη

In [2]:
```
seed_number=1 # Seed του τυχαίου προβλήματος
n=10 # Αριθμός πελατών
Q=400 # Μέγιστη χωρητικότητα οχημάτων (ομοιογενής στόλος)
max_x=200 # Διάσταση x του διαγράμματος (απο 0 έως max_x)
max_y=100 # Διάσταση y του διαγράμματος (απο 0 έως max_y)
b1=4 # Παράμετρος ορισμού της ζήτησης
b2=10 # Παράμετρος ορισμού της ζήτησης
tol=0 # Παράμετρος για τον υπολογισμό των οχημάτων
```

In [3]:
```
stime=60 # Επιτρεπόμενος χρόνος επίλυσης
sstime=5 # Επιτρεπόμενος χρόνος του μοντέλου της αρχικής λύσης
```

3) Υπολογιζόμενα μεγέθη

In [4]:
```
rnd=np.random
rnd.seed(seed_number)
```

In [5]:
```
max_d=round(Q/b1)  # Μεγίστη δυνατή ζήτηση
min_d=round(Q/b2)  # Ελάχιστη δυνατή ζήτηση
evnv=math.ceil((n*(max_d+min_d))/(2*Q))
NV=evnv+tol
```

4) Δημιουργία τυχαίου προβλήματος

In [6]:
```
xc=rnd.rand(n+1)*max_x # Δημιουργία τυχαίων x
yc=rnd.rand(n+1)*max_y # Δημιουργία τυχαίων y
N=[i for i in range(1,n+1)] #Το σύνολο των πελατών
K=[i for i in range(1,NV+1)] # Το σύνολο των οχημάτων
V=[0]+N #Το σύνολο των κόμβων
A=[(i,j) for i in V for j in V if i!=j] #Όλα τα πιθανά τόξα i->j
c={(i,j): np.hypot(xc[i]-xc[j],yc[i]-yc[j]) for i,j in A} #Τα κόστη μετάβασης ίσα με την απόσταση μεταξύ των σημείων.
q={i: rnd.randint(min_d,max_d+1) for i in N} #Η τυχαία ζήτηση του κάθε πελάτη
```

5) Κατασκευή του μοντέλου

Δημιουργία κενού Model

In [7]:
```
m=Model("Capacitated Vehicle Routing Problem") # Εισαγωγή μοντέλου
```

```
Using license file C:\Users\thana\gurobi.lic
Academic license - for non-commercial use only - expires 2021-07-30
```

Ορισμός των μεταβλητών

In [8]:
```
x=m.addVars(A,vtype=GRB.BINARY,name='x') # Εισαγωγή μεταβλητής xij
u=m.addVars(N,vtype=GRB.CONTINUOUS,name='u') # Εισαγωγή μεταβλητής ui
```

Ορισμός των περιορισμών

In [9]:
```
con1=m.addConstrs((quicksum(x[i,j] for j in V if j!=i)==1 for i in N),name='con1') #Περιορισμός 1
con2=m.addConstrs((quicksum(x[i,j] for i in V if i!=j)==1 for j in N),name='con2') #Περιορισμός 2
con3=m.addConstr((quicksum(x[0,j] for j in N)==NV),name="con3") #Περιορισμός 3
con4=m.addConstrs((u[i]-u[j]+Q*x[i,j]<=Q-q[j] for i,j in A if i!=0 and j!=0),name='con4') #Περιορισμός 4
con5_1=m.addConstrs((q[i]<=u[i] for i in N),name='con5_1') #Περιορισμός 5
con5_2=m.addConstrs((u[i]<=Q for i in N),name='con5_2') #Περιορισμός 5
```

Ορισμός της αντικειμενικής συνάρτησης

In [10]:
```
m.setObjective(quicksum(x[i,j]*c[i,j] for i,j in A),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής
```

6) Υπολογισμός αρχικής εφικτής λύσης

Μοντέλο υπολογισμού της αρχική λύσης

In [11]:
```
ssm=Model("Starting Solution Model")
rtv=ssm.addVars(K,vtype=GRB.INTEGER,name="rtv")
AV=[(i,j) for i in K for j in N]
acus=ssm.addVars(AV,vtype=GRB.BINARY,name="acus")
sscon1=ssm.addConstrs((rtv[i]==quicksum(q[j]*acus[i,j] for j in N) for i in K),name="sscon1")
sscon2=ssm.addConstrs((quicksum(acus[i,j] for i in K)==1 for j in N),name="sscon2")
sscon3=ssm.addConstrs((rtv[i]<=Q for i in K),name="sscon3")
sscon4=ssm.addConstrs((rtv[i]>=1 for i in K),name="sscon4")
ssm.setObjective(NV*Q-quicksum(rtv[i] for i in K),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής
```

Επίλυση του μοντέλου της αρχικής λύσης

In [12]:
```
ssm.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
ssm.Params.TimeLimit=sstime #Όριο χρόνου
ssm.optimize()
```

Ορισμός της αρχικής λύσης στο CVRP

In [13]:
```
arks_in_ssol=[] # Λίστα με τα τόξα της αρχικής λύσης
arks_not_in_ssol=A[:] # Λίστα με τα τόξα που δεν συμετέχουν στην αρχική λύση
for veh in K:
    ssroute=[0]
    for cu in N:
        if acus[veh,cu].getAttr('x')>0.5:
            ssroute.append(cu)
    ssroute.append(0)
    if len(ssroute)>2:
        for i in range(len(ssroute)-1):
            arc=(ssroute[i],ssroute[i+1])
            arks_in_ssol.append(arc)
            arks_not_in_ssol.remove(arc)

for i,j in arks_in_ssol:
    x[i,j].setAttr("Start",1)
for i,j in arks_not_in_ssol:
    x[i,j].setAttr("Start",0)
```

7) Επίλυση του CVRP

In [14]:
```
m.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
m.Params.TimeLimit=stime
m.optimize()
```

8) Επιθεώρηση της λύσης

Προβολή των τιμών των μη μηδενικών μεταβλητών

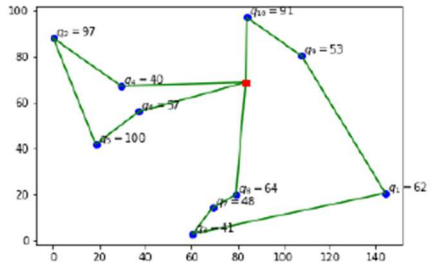
In [15]:
```
m.printAttr('x')
```

Τιμή της αντικειμενικής στην λύση

In [16]:
```
m.printAttr('ObjVal')
```

Γράφημα δικτύου και διαδρομών

In [17]:
```
active_arcs=[a for a in A if x[a].x>0.99] #Ευρεση ενεργών τόξων
for i,j in active_arcs:
    plt.plot([xc[i],xc[j]],[yc[i],yc[j]],c='g') #Σχεδιασμός της διαδρομής i->j
plt.plot(xc[0],yc[0],c='r',marker='s') # Τοποθέτηση του σημείου 0 (κέντρο διανομής)
plt.scatter(xc[1:],yc[1:],c='b') # Τοποθέτηση των πελατών
for i in N:
    plt.annotate('$q_{%d} = %d$'%(i,q[i]),(xc[i]+1,yc[i]+1)) #Εμφάνοση της ζήτησης σε κάθε πελάτη
```



Αλληλουχία και χωρητικότητα των διαδρομών

In [18]:
```
number_of_routes=0
for i in range(len(active_arcs)):
    if active_arcs[i][0]==0:
        number_of_routes=number_of_routes+1
print('Total nuber of routes : '+str(number_of_routes))
print(' ')

for i in range(number_of_routes):
    next_node=active_arcs[i][1]
    TC=q[next_node]
    route_text=str('Route '+str(i+1)+' : '+str(0)+', '+str(next_node)+', ')
    Route_Completed=False
    while Route_Completed==False:
        j=number_of_routes+next_node-1
        next_node=active_arcs[j][1]
        if next_node==0:
            Route_Completed=True
            route_text=route_text+str(0)
            print(route_text)
            print('Route '+str(i+1)+' has a total capacity of '+str(TC))
            print(' ')
        else:
            route_text=route_text+str(str(next_node)+', ')
            TC=TC+q[next_node]
```

```
Total nuber of routes : 2

Route 1 : 0, 4, 2, 5, 6, 0
Route 1 has a total capacity of 294

Route 2 : 0, 10, 9, 1, 3, 7, 8, 0
Route 2 has a total capacity of 359
```

A2 Κώδικας CVRP με callbacks (CVRP 2)

1) Εισαγωγή Βιβλιοθηκών

In [1]:
```
import numpy as np
from gurobipy import *
import matplotlib.pyplot as plt
```

2) Εισαγόμενα μεγέθη

In [2]:
```
seed_number=1 # Seed του τυχαίου προβλήματος
n=10 # Αριθμός πελατών
Q=400 # Μέγιστη χωρητικότητα οχημάτων (ομοιογενής στόλος)
max_x=200 # Διάσταση x του διαγράμματος (απο 0 έως max_x)
max_y=100 # Διάσταση y του διαγράμματος (απο 0 έως max_y)
b1=4 # Παράμετρος ορισμού της ζήτησης
b2=10 # Παράμετρος ορισμού της ζήτησης
tol=0 # Παράμετρος για τον υπολογισμό των οχημάτων
```

In [3]:
```
stime=60 # Επιτρεπόμενος χρόνος επίλυσης
sstime=5 # Επιτρεπόμενος χρόνος του μοντέλου της αρχικής λύσης
```

3) Υπολογιζόμενα μεγέθη

In [4]:
```
rnd=np.random
rnd.seed(seed_number)
```

In [5]:
```
max_d=round(Q/b1)  # Μέγιστη δυνατή ζήτηση
min_d=round(Q/b2)  # Ελάχιστη δυνατή ζήτηση
evnv=math.ceil((n*(max_d+min_d))/(2*Q))
NV=evnv+tol
```

4) Δημιουργία τυχαίου προβλήματος

In [6]:
```
xc=rnd.rand(n+1)*max_x # Δημιουργία τυχαίων x
yc=rnd.rand(n+1)*max_y # Δημιουργία τυχαίων y
N=[i for i in range(1,n+1)] #Το σύνολο των πελατών
K=[i for i in range(1,NV+1)] # Το σύνολο των οχημάτων
V=[0]+N #Το σύνολο των κόμβων
A=[(i,j) for i in V for j in V if i!=j] #Όλα τα πιθανά τόξα i->j
c={(i,j): np.hypot(xc[i]-xc[j],yc[i]-yc[j]) for i,j in A} #Τα κόστη μετάβασης ίσα με την απόσταση μεταξύ των σημείων.
q={i: rnd.randint(min_d,max_d+1) for i in N} #Η τυχαία ζήτηση του κάθε πελάτη
```

5) Κατασκευή του μοντέλου

Δημιουργία κενού Model

In [7]:
```
m=Model("Capacitated Vehicle Routing Problem") # Εισαγωγή μοντέλου
```

```
Using license file C:\Users\thana\gurobi.lic
Academic license - for non-commercial use only - expires 2021-07-30
```

Ορισμός της μεταβλητών

In [8]:
```
x=m.addVars(A,vtype=GRB.BINARY,name='x') # Εισαγωγή μεταβλητής xij
```

Ορισμός των περιορισμών

In [9]:
```
con1=m.addConstrs((quicksum(x[i,j] for j in V if j!=i)==1 for i in N),name='con1') #Περιορισμός 1
con2=m.addConstrs((quicksum(x[i,j] for i in V if i!=j)==1 for j in N),name='con2') #Περιορισμός 2
con3=m.addConstr((quicksum(x[0,j] for j in N)==len(K)),name="con3") #Περιορισμός 3
```

Ορισμός της αντικειμενικής συνάρτησης

In [10]:
```
m.setObjective(quicksum(x[i,j]*c[i,j] for i,j in A),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής
```

6) Ορισμός της συνάρτησης Callback

In [11]:
```
def subtourelim(model, where):
    if where == GRB.Callback.MIPSOL:
        # make a list of edges selected in the solution
        vals = model.cbGetSolution(model._vars)
        selected = [(i, j) for i, j in model._vars.keys() if vals[i,j]>0.5]
        arklist=selected[:]
        arklist.sort()
        while arklist:
            (sta,nex)=arklist[0]
            arklist.remove(arklist[0])
            route=[sta,nex]
            flag=1
            while flag>0:
                if nex==0: # Εύρεση διαδρομής που επικοινωνεί με το 0
                    rute=route[1:len(route)-1]
                    rs1=math.ceil(sum([q[i] for i in rute])/Q)
                    if rs1>1:
                        #Περιορισμός 4
                        model.cbLazy(quicksum(model._vars[i,j] for i,j in A if i in rute if j not in rute)>=rs1)
                    flag=-1

                elif nex==sta: # Εύρεση διαδρομής που δεν επικοινωνεί με το 0
                    rute=route[:len(route)-1]
                    rs2=math.ceil(sum([q[i] for i in rute])/Q)
                    #Περιορισμός 4
                    model.cbLazy(quicksum(model._vars[i,j] for i,j in A if i in rute if j not in rute)>=rs2)
                    flag=-1

                else:
                    next_ark=[(nex,j) for j in V if (nex,j) in arklist]
                    (pre,nex)=next_ark[0]
                    arklist.remove(next_ark[0])
                    route.append(nex)
```

7) Υπολογισμός αρχικής εφικτής λύσης

Μοντέλο υπολογισμού της αρχική λύσης

In [12]:
```
ssm=Model("Starting Solution Model")
rtv=ssm.addVars(K,vtype=GRB.INTEGER,name="rtv")
AV=[(i,j) for i in K for j in N]
acus=ssm.addVars(AV,vtype=GRB.BINARY,name="acus")
sscon1=ssm.addConstrs((rtv[i]==quicksum(q[j]*acus[i,j] for j in N) for i in K),name="sscon1")
sscon2=ssm.addConstrs((quicksum(acus[i,j] for i in K)==1 for j in N),name="sscon2")
sscon3=ssm.addConstrs((rtv[i]<=Q for i in K),name="sscon3")
sscon4=ssm.addConstrs((rtv[i]>=1 for i in K),name="sscon4")
ssm.setObjective(NV*Q-quicksum(rtv[i] for i in K),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής
```

Επίλυση του μοντέλου της αρχικής λύσης

In [13]:
```
ssm.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
ssm.Params.TimeLimit=sstime #Όριο χρόνου
ssm.optimize()
```

9) Επιθεώρηση της λύσης

Προβολή των τιμών των μη μηδενικών μεταβλητών

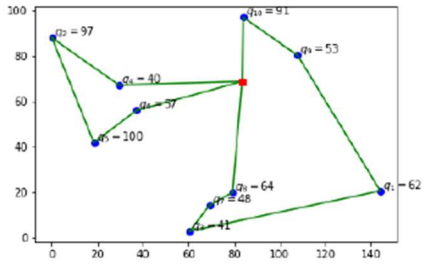
In [16]:
```
m.printAttr('x')
```

Τιμή της αντικειμενικής στην λύση

In [17]:
```
m.printAttr('ObjVal')
```

Γράφημα δικτύου και διαδρομών

In [18]:
```
active_arcs=[a for a in A if x[a].x>0.99] #Ευρεση ενεργών τόξων
for i,j in active_arcs:
    plt.plot([xc[i],xc[j]],[yc[i],yc[j]],c='g') #Σχεδιαδμός της διαδρομής i->j
plt.plot(xc[0],yc[0],c='r',marker='s') # Τοποθέτηση του σημείου 0 (κέντρο διανομής)
plt.scatter(xc[1:],yc[1:],c='b') # Τοποθέτηση των πελατών
for i in N:
    plt.annotate('$q_{%d} = %d$'%(i,q[i]),(xc[i]+1,yc[i]+1)) #Εμφάνοση της ζήτησης σε κάθε πελάτη
```



Αλληλουχία και χωρητικότητα των διαδρομών

In [19]:
```
number_of_routes=0
for i in range(len(active_arcs)):
    if active_arcs[i][0]==0:
        number_of_routes=number_of_routes+1
print('Total nuber of routes : '+str(number_of_routes))
print(' ')

for i in range(number_of_routes):
    next_node=active_arcs[i][1]
    TC=q[next_node]
    route_text=str('Route '+str(i+1)+' : '+str(0)+', '+str(next_node)+', ')
    Route_Completed=False
    while Route_Completed==False:
        j=number_of_routes+next_node-1
        next_node=active_arcs[j][1]
        if next_node==0:
            Route_Completed=True
            route_text=route_text+str(0)
            print(route_text)
            print('Route '+str(i+1)+' has a total capacity of '+str(TC))
            print(' ')
        else:
            route_text=route_text+str(str(next_node)+', ')
            TC=TC+q[next_node]
```

```
Total nuber of routes : 2

Route 1 : 0, 4, 2, 5, 6, 0
Route 1 has a total capacity of 294

Route 2 : 0, 8, 7, 3, 1, 9, 10, 0
Route 2 has a total capacity of 359
```

Α3 Κώδικας απλού PDP (PDP 1)

1) Εισαγωγή Βιβλιοθηκών

In [1]:
```
import numpy as np
from gurobipy import *
import matplotlib.pyplot as plt
```

2) Εισαγόμενα μεγέθη

In [2]:
```
seed_number=1 # Seed του τυχαίου προβλήματος
n=5 # Αριθμός απο ζεύγη πελατών
Qmax=200 # Μέγιστη χωρητικότητα οχημάτων (ομοιογενής στόλος)
max_x=500 # Διάσταση x του διαγράμματος (απο 0 έως max_x)
max_y=500 # Διάσταση y του διαγράμματος (απο 0 έως max_y)
b1=20 # Παράμετρος ορισμού της ζήτησης
b2=200 # Παράμετρος ορισμού της ζήτησης
tol=0 # Παράμετρος για τον υπολογισμό των οχημάτων
z1=3 # Παράμετρος ορισμού ταχύτητας οχημάτων
cpr=3 # Εκτιμούμενος αριθμός εξυπηρετούμενων ζευγών πελατών ανά δρομολόγιο
```

In [3]:
```
T=480 # Μέγιστη διάρκεια δρομολογίου
w1=1000 # Παράμετρος ορισμού χρονικών παραθύρων
w2=4 # Παράμετρος ορισμού χρονικών παραθύρων
w3=4 # Παράμετρος ορισμού χρονικών παραθύρων
w4=2 # Παράμετρος ορισμού χρονικών παραθύρων
w5=1000 # Παράμετρος ορισμού χρόνων φόρτωσης εκφόρτωσης
w6=2 # Παράμετρος ορισμού χρονικών παραθύρων
```

In [4]:
```
stime=60 # Επιτρεπόμενος χρόνος επίλυσης
```

3) Υπολογιζόμενα μεγέθη

In [5]:
```
rnd=np.random
rnd.seed(seed_number)
```

In [6]:
```
max_d=round(Qmax/b1)  # Μεγίστη δυνατή ζήτηση
min_d=round(Qmax/b2)  # Ελάχιστη δυνατή ζήτηση
tt=T/(z1*(max_x**2+max_y**2)**(1/2)) # Σε πόσες μονάδες χρόνου ένα όχημα διασχίζει μία μονάδα απόστασης
NV=math.ceil(n/cpr)
```

In [7]:
```
window=T/w6
eput=round(T/w1) # Ο συντομότερος χρόνος εξυπηρέτησης ενός πελάτη pickup (σε μονάδες χρόνου).
lput=round(T/w2) # Ο αργότερος χρόνος εξυπηρέτησης ενός πελάτη pickup (σε μονάδες χρόνου).
edut=round(T/w3)  # Ο συντομότερος χρόνος εξυπηρέτησης ενός πελάτη delivery (σε μονάδες χρόνου).
ldut=round(T/w4) # Ο αργότερος χρόνος εξυπηρέτησης ενός πελάτη delivery (σε μονάδες χρόνου).
load_time=(T/w5) # Ο χρόνος φόρτωσης/εκφόρτωσης μονάδας προϊόντος.
```

4) Δημιουργία τυχαίου προβλήματος

In [8]:
```
xs=rnd.rand()*max_x # Δημιουργία τυχαίου x του σημείου εκκίνησης
ys=rnd.rand()*max_x # Δημιουργία τυχαίου y του σημείου εκκίνησης
xf=xs # Δημιουργία τυχαίου x του σημείου τερματισμού
yf=ys # Δημιουργία τυχαίου y του σημείου τερματισμού
xp=[rnd.rand()*max_x for i in range(1,n+1)] #Δημιουργία τυχαίων x των σημείων φόρτωσης
yp=[rnd.rand()*max_y for i in range(1,n+1)] #Δημιουργία τυχαίων y των σημείων φόρτωσης
xd=[rnd.rand()*max_x for i in range(1,n+1)] #Δημιουργία τυχαίων x των σημείων παράδοσης
yd=[rnd.rand()*max_y for i in range(1,n+1)] #Δημιουργία τυχαίων y των σημείων παράδοσης
x_all=[xs]+xp+xd+[xf] # Λίστα με τις x συντεταγμένες όλων των σημείων
y_all=[ys]+yp+yd+[yf] # Λίστα με τις y συντεταγμένες όλων των σημείων
```

In [9]:
```
P=[i for i in range(1,n+1)] # Σύνολο πελατών φόρτωσης
D=[i for i in range(n+1,2*n+1)] # Σύνολο πελατών παράδοσης
V=[0]+P+D+[2*n+1] # Το σύνολο όλων των κόμβος (περιλαμβάνεται και ο κόμβος εκκίνησης και τερματισμού)
A=[(i,j) for i in V for j in V if i!=j if i!=2*n+1 if j!=0] # Το σύνολο των πιθανών διαδρομών
qp={i: rnd.randint(min_d,max_d+1) for i in P} # Δημιουργία random ποσοτήτων φόρτωσης
# Οι ποσότητες παράδοσης αντιστοιχίζονται στις ποσότητες φόρτωσης (qp[x]=-qd[x+n]).
qd={i: -qp[i-n] for i in D}
qes={0:0, 2*n+1:0} # Οι ζητήσεις στο σημείο άφιξης και τερματισμού είναι μηδενικές
q={**qp,**qd,**qes} # Ενοποίηση των ζητήσεων σε ένα ενοιαίο λεξικό
K=[i for i in range(1,NV+1)] #Σύνολο οχημάτων
Capacities=[Qmax for i in range(NV)] # Λίστα με τις χωρητικότητες των οχημάτων
Ck= dict(zip(K,Capacities)) # Λεξικό που αντιστοιχεί κάθε όχημα στο K στην χωρητικότητα του
# Ο πίνακας αποστάσεων των σημείων
dis_matrixij={(i,j): np.hypot(x_all[i]-x_all[j],y_all[i]-y_all[j]) for i,j in A}
# Λεξικό με το κόστος λειτουργίας κάθε οχήματος για κάθε διαδρομή.
cijk={(i,j,k): dis_matrixij[i,j] for i,j in A for k in K}
AK=[(i,j,k) for i,j in A for k in K] # Λίστα με πλειάδες των συνδυασμών των στοιχείων των συνόλων A και K
VK=[(i,k) for i in V for k in K]  # Λίστα με πλειάδες των συνδυασμών των στοιχείων των συνόλων V και K
```

In [10]:
```
tijk={(i,j,k): dis_matrixij[i,j]*tt for i,j in A for k in K} # Υποθέτω κάθε όχημα έχει την ίδια ταχύτητα
e_pickup_times=[rnd.randint(eput,lput) for i in range(n)]
e_delivery_times=[rnd.randint(edut,ldut) for i in range(n)]
l_pickup_times=[e_pickup_times[i]+window for i in range(n)]
l_delivery_times=[e_delivery_times[i]+window for i in range(n)]
etimes=[0]+e_pickup_times+e_delivery_times+[0] # Λίστα με τους νωρίτερους χρόνους εξυπηρέτησης κάθε σημείου.
ftimes=[T]+l_pickup_times+l_delivery_times+[T] # Λίστα με τους αργότερους χρόνους εξυπηρέτησης κάθε σημείου.
# Λίστα με τους χρόνους φόρτωσης εκφόρτωσης σε κάθε σημείο.
stimes=[0]+[qp[i]*load_time for i in P]+[qp[i]*load_time for i in P]+[0]
Ttimes=[T for i in range(NV)] # Λίστα με μέγιστη ώρα δρομολογίου κάθε οχήματος.
ei=dict(zip(V,etimes)) # Λεξικό με τους νωρίτερους χρόνους εξυπηρέτησης κάθε σημείου.
li=dict(zip(V,ftimes)) # Λεξικό με τους αργότερους χρόνους εξυπηρέτησης κάθε σημείου.
di=dict(zip(V,stimes)) # Λεξικό με τους χρόνους φόρτωσης εκφόρτωσης σε κάθε σημείο.
Tk=dict(zip(K,Ttimes)) # Λεξικό με μέγιστη ώρα δρομολογίου κάθε οχήματος.
```

5) Κατασκευή του μοντέλου

Δημιουργία κενού Model

In [11]:
```
m=Model("Pick_Up_and_Delivery_Model")
```

```
Using license file C:\Users\thana\gurobi.lic
Academic license - for non-commercial use only - expires 2021-07-30
```

Ορισμός των μεταβλητών

In [12]:
```
x=m.addVars(AK,vtype=GRB.BINARY,name='x') # Η διαδική μεταβλητή x
Q=m.addVars(VK,vtype=GRB.CONTINUOUS,name='Q') # Η συνεχής μεταβλητή Q
B=m.addVars(VK,vtype=GRB.CONTINUOUS,name='B') # Η συνεχής μεταβλητή B
```

Ορισμός των περιορισμών

In [13]:
```
# Περιορισμός 1
con1=m.addConstrs((quicksum(x[i,j,k] for j in V if j!=i if j!=0 for k in K)==1 for i in P+D),name="con1")
con2=m.addConstrs((quicksum(x[0,j,k] for j in V if j!=0)==1 for k in K),name="con2") # Περιορισμός 2
con3=m.addConstrs((quicksum(x[i,2*n+1,k] for i in V if i!=2*n+1)==1 for k in K),name="con3") # Περιορισμός 3
# Περιορισμός 4
con4=m.addConstrs(((quicksum(x[i,j,k] for i in V if i!=j if i!=2*n+1)
                    -quicksum(x[j,i,k] for i in V if i!=j if i!=0))==0 for j in P+D for k in K),name="con4")
# Περιορισμός 5
con5=m.addConstrs(((x[i,j,k]==1) >> (B[j,k]>=B[i,k]+di[i]+tijk[i,j,k]) for i,j in A for k in K),name="con5")
con6=m.addConstrs(((x[i,j,k]==1) >> (Q[j,k]==Q[i,k]+q[j]) for i,j in A for k in K),name="con6") # Περιορισμός 6
# Περιορισμός 7 (για τα σημεία φόρτωσης οπου q>0)
con7_1_1=m.addConstrs((Q[i,k]>=q[i] for i in P for k in K),name="con7_1_1")
# Περιορισμός 7 (για τα σημεία φόρτωσης οπου q>0)
con7_1_2=m.addConstrs((Q[i,k]<=Ck[k] for i in P for k in K),name="con7_1_2")
# Περιορισμός 7 (για τα σημεία εκφόρτωσης οπου q<0)
con7_2_1=m.addConstrs((Q[i,k]>=0 for i in [0]+D+[2*n+1] for k in K),name="con7_2_1")
# Περιορισμός 7 (για τα σημεία εκφόρτωσης οπου q<0)
con7_2_2=m.addConstrs((Q[i,k]<=Ck[k]+q[i] for i in [0]+D+[2*n+1] for k in K),name="con7_2_2")
```

In [14]:
```
# Περιορισμός 8
con8=m.addConstrs((quicksum(x[i,j,k] for j in V if j!=i if j!=0)
                   -quicksum(x[n+i,j,k] for j in V if j!=n+i if j!=0)==0 for i in P for k in K),name="con8")
con9=m.addConstrs((B[i,k]<=B[i+n,k] for i in P for k in K),name="con9") # Περιορισμός 9
con10_1=m.addConstrs((B[i,k]>=ei[i] for i in V for k in K),name="con10_1") # Περιορισμός 10
con10_2=m.addConstrs((B[i,k]<=li[i] for i in V for k in K),name="con10_2") # Περιορισμός 10
con11=m.addConstrs((B[2*n+1,k]-B[0,k]<=Tk[k] for k in K),name="con11") # Περιορισμός 11
```

Ορισμός της αντικειμενικής συνάρτησης

In [15]:
```
m.setObjective(quicksum(x[i,j,k]*cijk[i,j,k] for i,j in A for k in K),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής
```

6) Επίλυση του PDP

In [16]:
```
m.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
m.Params.TimeLimit=stime #Όριο χρόνου
m.optimize()
```

7) Επιθεώρηση της λύσης

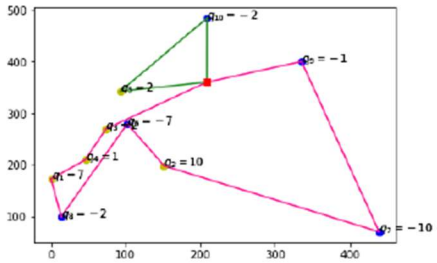
Αναλυτική αναφορά δρομολογίων

In [17]:
```
print('Το ελάχιστο κόστος είναι '+str(round(m.ObjVal,2))+'.') # Εκτύπωση του ελάχιστου κόστους με 2 δεκαδικά ψηφία.
print(' ')

UV=[] # Η λίστα αυτή θα γεμίσει με τα οχήματα που θα χρησιμοποιηθούν.
for k in K:
    if x[0,2*n+1,k].X>0.99: # Εάν x[αρχή,τερματισμός,k]=1 σημαίνει πως το όχημα αυτό δεν πραγματοποιεί κάποια διαδρομή.
        print('Το όχημα '+str(k)+' δεν χρησιμοποιείται.')
    else:
        UV.append(k)
print(' ')

for k in UV:
    print(' ')
    end_of_route=1
    list_of_routes=[(i,j) for i,j in A if x[i,j,k].X>0.99]
    for j in P:
        if x[0,j,k].X>0.99:
            nextp=j
            print('Το όχημα '+str(k)+' ξεκινά απο την έναρξη και φτάνει στον πελάτη '
                  +str(j)+' σε '+str(round(B[j,k].X,2))+' χρονικές μονάδες, όπου φορτώνει '+str(q[j])
                  +' μονάδες προϊόντος.')
    while end_of_route>0:
        for i,r in list_of_routes:
            if nextp==2*n+1:
                print('Το όχημα επιστρέφει στον τερματισμό. Η συνολική διάρκεια του δρομολογίου είναι ίση με '
                      +str(round(total_time,2))+' χρονικές μονάδες.')
                end_of_route=-1
                nextp=2*n+2
            if nextp==i:
                if q[r]>0:
                    print('Το όχημα '+str(k)+' στην συνέχεια επισκέπτεται τον πελάτη '+str(r)+' σε '
                          +str(round(B[r,k].X,2))+' χρονικές μονάδες, όπου φορτώνει '+str(q[r])+' μονάδες προϊόντος.')
                    nextp=r
                else:
                    if q[r]<0:
                        print('Το όχημα '+str(k)+ ' στην συνέχεια επισκέπτεται τον πελάτη '+str(r)
                              +' σε '+str(round(B[r,k].X,2))+' χρονικές μονάδες, όπου εκφορτώνει '+str(q[r]*(-1))
                              +' μονάδες προϊόντος.')
                        nextp=r
                    else:
                        total_time=B[i,k].X+di[i]+tijk[i,r,k]
                        nextp=r
```

```
Το ελάχιστο κόστος είναι 1875.51.

Το όχημα 1 ξεκινά απο την έναρξη και φτάνει στον πελάτη 3 σε 99.17 χρονικές μονάδες, όπου φορτώνει 2 μονάδες προϊόντος.
Το όχημα 1 στην συνέχεια επισκέπτεται τον πελάτη 4 σε 115.0 χρονικές μονάδες, όπου φορτώνει 1 μονάδες προϊόντος.
Το όχημα 1 στην συνέχεια επισκέπτεται τον πελάτη 1 σε 129.67 χρονικές μονάδες, όπου φορτώνει 7 μονάδες προϊόντος.
Το όχημα 1 στην συνέχεια επισκέπτεται τον πελάτη 8 σε 150.0 χρονικές μονάδες, όπου εκφορτώνει 2 μονάδες προϊόντος.
Το όχημα 1 στην συνέχεια επισκέπτεται τον πελάτη 6 σε 226.0 χρονικές μονάδες, όπου εκφορτώνει 7 μονάδες προϊόντος.
Το όχημα 1 στην συνέχεια επισκέπτεται τον πελάτη 2 σε 250.77 χρονικές μονάδες, όπου φορτώνει 10 μονάδες προϊόντος.
Το όχημα 1 στην συνέχεια επισκέπτεται τον πελάτη 7 σε 326.87 χρονικές μονάδες, όπου εκφορτώνει 10 μονάδες προϊόντος.
Το όχημα 1 στην συνέχεια επισκέπτεται τον πελάτη 9 σε 431.0 χρονικές μονάδες, όπου εκφορτώνει 1 μονάδες προϊόντος.
Το όχημα επιστρέφει στον τερματισμό. Η συνολική διάρκεια του δρομολογίου είναι ίση με 461.56 χρονικές μονάδες.

Το όχημα 2 ξεκινά απο την έναρξη και φτάνει στον πελάτη 5 σε 91.41 χρονικές μονάδες, όπου φορτώνει 2 μονάδες προϊόντος.
Το όχημα 2 στην συνέχεια επισκέπτεται τον πελάτη 10 σε 304.0 χρονικές μονάδες, όπου εκφορτώνει 2 μονάδες προϊόντος.
Το όχημα επιστρέφει στον τερματισμό. Η συνολική διάρκεια του δρομολογίου είναι ίση με 333.01 χρονικές μονάδες.
```

Γραφήμα δικτύου - διαδρομών

In [18]:
```
all_xs=[xs]+xp+xd+[xf] # Λίστα με όλα τα x
all_ys=[ys]+yp+yd+[yf] # Λίστα με όλα τα y

 # Κωδικοί χρωμάτων για μέχρι 5 οχήματα (με περισότερα δεν θα έχει νόημα το παρακάτω γράφημα)
colourk={1:"deeppink",2:"forestgreen",3:"orange",4:"deepskyblue",5:"gray"}

for k in K:
    active_arcs=[(i,j) for i,j in A if x[i,j,k].X>0.99] #Ευρεση ενεργών τόξων
    for i,j in active_arcs:
        plt.plot([all_xs[i],all_xs[j]],[all_ys[i],all_ys[j]],c=colourk[k%5]) #Σχεδιαδμός της διαδρομής i->j
    plt.plot(xs,ys,c='r',marker='s') # Τοποθέτηση του σημείου 0 (εκκίνηση-τερματισμός).
    plt.scatter(xp,yp,c='y') # Τοποθέτηση σημείων φόρτωσης
    plt.scatter(xd,yd,c='b') # Τοποθέτηση σημείων παράδοσης
    for i in P:
        plt.annotate('$q_{%d} = %d$'%(i,q[i]),(x_all[i],y_all[i])) #Εμφάνοση της ζήτησης σε κάθε πελάτη
    for i in D:
        plt.annotate('$q_{%d} = %d$'%(i,q[i]),(x_all[i],y_all[i])) #Εμφάνοση της ζήτησης σε κάθε πελάτη
```



Α4 Κώδικας PDP με callbacks (PDP 2)

1) Εισαγωγή Βιβλιοθηκών

In [1]:
```
import numpy as np
from gurobipy import *
import matplotlib.pyplot as plt
```

2) Εισαγόμενα μεγέθη

In [2]:
```
seed_number=1 # Seed του τυχαίου προβλήματος
n=5 # Αριθμός απο ζεύγη πελατών
Qmax=200 # Μέγιστη χωρητικότητα οχημάτων (ομοιογενής στόλος)
max_x=500 # Διάσταση x του διαγράμματος (απο 0 έως max_x)
max_y=500 # Διάσταση y του διαγράμματος (απο 0 έως max_y)
b1=20 # Παράμετρος ορισμού της ζήτησης
b2=200 # Παράμετρος ορισμού της ζήτησης
z1=3 # Παράμετρος ορισμού ταχύτητας οχημάτων
```

In [3]:
```
T=480 # Μέγιστη διάρκεια δρομολογίου
w1=1000 # Παράμετρος ορισμού χρονικών παραθύρων
w2=4 # Παράμετρος ορισμού χρονικών παραθύρων
w3=4 # Παράμετρος ορισμού χρονικών παραθύρων
w4=2 # Παράμετρος ορισμού χρονικών παραθύρων
w5=1000 # Παράμετρος ορισμού χρόνων φόρτωσης εκφόρτωσης
w6=2 # Παράμετρος ορισμού χρονικών παραθύρων
```

In [4]:
```
stime=60 # Επιτρεπόμενος χρόνος επίλυσης
```

3) Υπολογιζόμενα μεγέθη

In [5]:
```
rnd=np.random
rnd.seed(seed_number)
```

In [6]:
```
max_d=round(Qmax/b1)  # Μεγιστη δυνατή ζήτηση
min_d=round(Qmax/b2)  # Ελάχιστη δυνατή ζήτηση
tt=T/(z1*(max_x**2+max_y**2)**(1/2)) # Σε πόσες μονάδες χρόνου ένα όχημα διασχίζει μία μονάδα απόστασης
```

In [7]:
```
window=T/w6
eput=round(T/w1) # Ο συντομότερος χρόνος εξυπηρέτησης ενός πελάτη pickup (σε μονάδες χρόνου).
lput=round(T/w2) # Ο αργότερος χρόνος εξυπηρέτησης ενός πελάτη pickup (σε μονάδες χρόνου).
edut=round(T/w3)  # Ο συντομότερος χρόνος εξυπηρέτησης ενός πελάτη delivery (σε μονάδες χρόνου).
ldut=round(T/w4) # Ο αργότερος χρόνος εξυπηρέτησης ενός πελάτη delivery (σε μονάδες χρόνου).
load_time=(T/w5) # Ο χρόνος φόρτωσης/εκφόρτωσης μονάδας προϊόντος.
```

4) Δημιουργία τυχαίου προβλήματος

In [8]:
```
xs=rnd.rand()*max_x # Δημιουργία τυχαίου x του σημείου εκκίνησης
ys=rnd.rand()*max_x # Δημιουργία τυχαίου y του σημείου εκκίνησης
xf=xs # Δημιουργία τυχαίου x του σημείου τερματισμού
yf=ys # Δημιουργία τυχαίου y του σημείου τερματισμού
xp=[rnd.rand()*max_x for i in range(1,n+1)] #Δημιουργία τυχαίων x των σημείων φόρτωσης
yp=[rnd.rand()*max_y for i in range(1,n+1)] #Δημιουργία τυχαίων y των σημείων φόρτωσης
xd=[rnd.rand()*max_x for i in range(1,n+1)] #Δημιουργία τυχαίων x των σημείων παράδοσης
yd=[rnd.rand()*max_y for i in range(1,n+1)] #Δημιουργία τυχαίων y των σημείων παράδοσης
x_all=[xs]+xp+xd+[xf] # Λίστα με τις x συντεταγμένες όλων των σημείων
y_all=[ys]+yp+yd+[yf] # Λίστα με τις y συντεταγμένες όλων των σημείων
```

In [9]:
```
P=[i for i in range(1,n+1)] # Σύνολο πελατών φόρτωσης
D=[i for i in range(n+1,2*n+1)] # Σύνολο πελατών παράδοσης
PD=P+D # Η ένωση των συνόλων P και D
V=[0]+P+D+[2*n+1] # Το σύνολο όλων των κόμβος (περιλαμβάνεται και ο κόμβος εκκίνησης και τερματισμού)
A1=[(0,j) for j in P] # Το υποσύνολο του A που απότελείτε απο τους κόμβους που ξεκινάνε απο το σημείο εκκίνησης
A2=[(i,j) for i in PD for j in PD if i!=j] # # Το υποσύνολο του A που απότελείτε απο τους κόμβους μεταξύ των πελατών
A3=[(i,2*n+1) for i in D] # Το υποσύνολο του A που απότελείτε απο τους κόμβους που καταλήγουν στο σημείο τερματισμού
A=A1+A2+A3 # Το σύνολο των πιθανών διαδρομών
qp={i: rnd.randint(min_d,max_d+1) for i in P} # Δημιουργία random ποσοτήτων φόρτωσης
# Οι ποσότητες παράδοσης αντιστοιχίζονται στις ποσότητες φόρτωσης (qp[x]=-qd[x+n]).
qd={i: -qp[i-n] for i in D}
qes={0:0, 2*n+1:0} # Οι ζητήσεις στο σημείο άφιξης και τερματισμού είναι μηδενικές
q={**qp,**qd,**qes} # Ενοποίηση των ζητήσεων σε ένα ενοιαίο λεξικό
# Ο πίνακας αποστάσεων των σημείων
dis_matrixij={(i,j): np.hypot(x_all[i]-x_all[j],y_all[i]-y_all[j]) for i,j in A}
cij=dis_matrixij # Λεξικό με τα κόστη μετάβασης.
```

In [10]:
```
tij={(i,j): dis_matrixij[i,j]*tt for i,j in A} # Υποθέτω κάθε όχημα έχει την ίδια ταχύτητα
e_pickup_times=[rnd.randint(eput,lput) for i in range(n)]
e_delivery_times=[rnd.randint(edut,ldut) for i in range(n)]
l_pickup_times=[e_pickup_times[i]+window for i in range(n)]
l_delivery_times=[e_delivery_times[i]+window for i in range(n)]
etimes=[0]+e_pickup_times+e_delivery_times+[0] # Λίστα με τους νωρίτερους χρόνους εξυπηρέτησης κάθε σημείου.
ftimes=[T]+l_pickup_times+l_delivery_times+[T] # Λίστα με τους αργότερους χρόνους εξυπηρέτησης κάθε σημείου.
# Λίστα με τους χρόνους φόρτωσης εκφόρτωσης σε κάθε σημείο.
stimes=[0]+[qp[i]*load_time for i in P]+[qp[i]*load_time for i in P]+[0]
ei=dict(zip(V,etimes)) # Λεξικό με τους νωρίτερους χρόνους εξυπηρέτησης κάθε σημείου.
li=dict(zip(V,ftimes)) # Λεξικό με τους αργότερους χρόνους εξυπηρέτησης κάθε σημείου.
di=dict(zip(V,stimes)) # Λεξικό με τους χρόνους φόρτωσης εκφόρτωσης σε κάθε σημείο.
```

5) Κατασκευή του μοντέλου

Δημιουργία κενού Model

In [11]:
```
m=Model("Pick_Up_and_Delivery_Model_Callbacks")
```

```
Using license file C:\Users\thana\gurobi.lic
Academic license - for non-commercial use only - expires 2021-07-30
```

Ορισμός της μεταβλητής

In [12]:
```
x=m.addVars(A,vtype=GRB.BINARY,name='x') # Η διαδική μεταβλητή x
```

Ορισμός των περιορισμών

In [13]:
```
con1=m.addConstrs((quicksum(x[i,j] for i in V if (i,j) in A)==1 for j in P+D),name="con1")
con2=m.addConstrs((quicksum(x[i,j] for j in V if (i,j) in A)==1 for i in P+D),name="con2")
```

Ορισμός της αντικειμενικής συνάρτησης

In [14]:
```
m.setObjective(quicksum(x[i,j]*cij[i,j] for i,j in A),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής
```

6) Ορισμός της συνάρτησης callback

In [15]:
```
def subtourelim(model, where):
    if where == GRB.Callback.MIPSOL:
        # make a list of edges selected in the solution
        vals = model.cbGetSolution(model._vars)
        selected = [(i, j) for i, j in model._vars.keys() if vals[i,j]>0.5]
        arklist=selected[:]
        arklist.sort()
        while arklist:
            (sta,nex)=arklist[0]
            arks_in_route=[arklist[0]]
            arklist.remove(arklist[0])
            route=[sta,nex]
            flag=1
            while flag>0:
                if nex==sta or nex==2*n+1: # Εύρεση διαδρομής
                    route_type=routetype(route)
                    if route_type==1: # Στην περίπτωση αποκομένης διαδρομής
                        # Το route περιέχει ένα στοιχείο 2 φορές. Δεν περιέχει το 0 και 2n+1
                        # Ο περιορισμός αφορά την χωρητικότητα + subtour elimination
                        con_arcs=[(i,j) for i in route[1:] for j in route[1:] if i!=j]
                        rs=math.ceil(abs(sum([q[i] for i in route[1:]]))/Qmax)
                        con_value=max(1,rs) # Η τιμή που θα μπεί στον περιορισμό της χωρητικότητας
                        model.cbLazy(quicksum(model._vars[i,j] for i,j in con_arcs)<=len(route)-1-con_value)
                    else: # Στην περίπτωση κανονικής διαδρομής
                        # Το route περιέχει το 0 και 2n+1
                        # Το ceiling της συνολικής ζήτησης / τη χωρητικότητα των οχημάτων
                        if len(route)-2>2:
                            for i in range(2,len(route)-1):
                                route_segment=route[1:i+1]
                                nrs=len(route_segment)
                                arcs_in_route_segment=[(route_segment[i],route_segment[i+1]) for i in range(nrs-1)]
                                rs=math.ceil(abs(sum([q[i] for i in route_segment]))/Qmax)
                                con_value=max(1,rs) # Η τιμή που θα μπεί στον περιορισμό της χωρητικότητας
                                if len(route_segment)>0 and rs>1:
                                    con_arcs=[(i,j) for i in route_segment for j in route_segment if i!=j]
                                    model.cbLazy(quicksum(model._vars[i,j] for i,j in con_arcs)<=nrs-con_value)
                        # Στην περίπτωση που υπάρχουν unpaired pick up and demands
                        if route_type==2:
                                con_arcs_1=[(i,j) for i in route for j in route if i!=j if i!=2*n+1 if j!=2*n+1]
                                con_arcs_2=[arc for arc in con_arcs_1 if arc in A]
                                model.cbLazy(quicksum(model._vars[i,j] for i,j in con_arcs_2)<=len(route)-1-2)
                        # Στην περίπτωση που κάποιος χρονικός περιορισμός παραβιάζεται
                        if route_type==3:
                            con_route=route[1:len(route)-1]
                            arks_conroute=[(route[i],route[i+1]) for i in range(len(con_route)-1)]
                            #model.cbLazy(quicksum(model._vars[i,j] for i,j in arks_in_route)<=len(arks_in_route)-1)
                            model.cbLazy(quicksum(model._vars[i,j] for i,j in arks_conroute)<=len(arks_conroute)-1)
                        flag=-1

                else:
                    next_ark=[(nex,j) for j in V if (nex,j) in arklist]
                    (pre,nex)=next_ark[0]
                    arks_in_route.append(next_ark[0])
                    arklist.remove(next_ark[0])
                    route.append(nex)
```

In [16]:
```
def routetype(rout):
    if rout[0]==0: # Διαδρομή μεταξύ σημείο έναρξης και τερματισμού
        unpaired=-1
        rout_delivery_nodes=[i for i in rout if i>n if i!=2*n+1]
        for i in rout_delivery_nodes:
            if i-n not in rout:       # Εάν στην διαδρομή περιλαμβάνεται αλλά όχι delivery το pick up
                unpaired=1
        if unpaired>0: # Εάν βρεθεί πως η διαδρομή είναι unpaired
            routtype=2
        else:
            in_time=isintime(rout)
            if in_time<0: # Εάν βρεθεί πως η διαδρομή παραβιάζει κάποιους απο τους χρονικούς περιορισμούς
                routtype=3
            else:
                routtype=4

    else:
        routtype=1 # Κλειστή διαδρομή που δεν επικοινωνεί με starting point και ending point

    return routtype
```

In [17]:
```
def isintime(rut):

    intime=1 # Εάν δεν παραβιαστεί κανένας χρονικός περιορισμός κρατά θετική τιμή

    # Εάν υπάρχουν παραπάνω pick up απο ότι delivery
    #if len(rut)%2!=0:
        #intime=-1

    # Τα pick up γίνονται πρίν το delivery
    pick_in_route=[i for i in rut if i>0 if i<n+1]
    del_in_route=[i for i in rut if i>n if i<2*n+1]
    if len(pick_in_route)==len(del_in_route):
        for i in pick_in_route:
            if rut.index(i)>rut.index(i+n):
                intime=-1

    # Τήρηση χρονικών πλαισίων
    out_node=ei[0]
    for i in range(1,len(rut)):
        in_node=max([ei[rut[i]],out_node+tij[rut[i-1],rut[i]]])
        out_node=in_node+di[rut[i]]
        if out_node>li[rut[i]]:
            intime=-1
            break


    return intime
```

7) Εύρεση αρχικής εφικτής λύσης

In [18]:
```
arks_not_in_solu=A[:]
for i in P:
    x[0,i].setAttr("Start",1)
    x[i,i+n].setAttr("Start",1)
    x[i+n,2*n+1].setAttr("Start",1)
    arks_not_in_solu.remove((0,i))
    arks_not_in_solu.remove((i,i+n))
    arks_not_in_solu.remove((i+n,2*n+1))

for i,j in arks_not_in_solu:
    x[i,j].setAttr("Start",0)
    pass

m.update()
```

8) Επίλυση του PDP

In [19]:
```
m.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
m.Params.TimeLimit=stime #Όριο χρόνου
m._vars = x
m.Params.lazyConstraints = 1
m.optimize(subtourelim)
```

9) Επιθεώρηση της λύσης

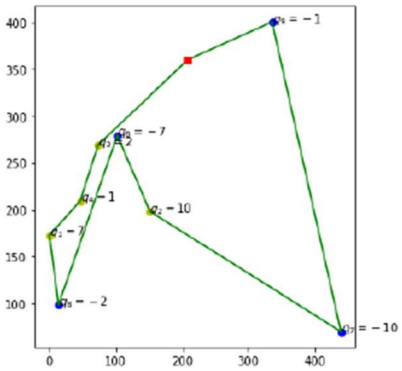
Αναλυτική αναφορά δρομολογίων

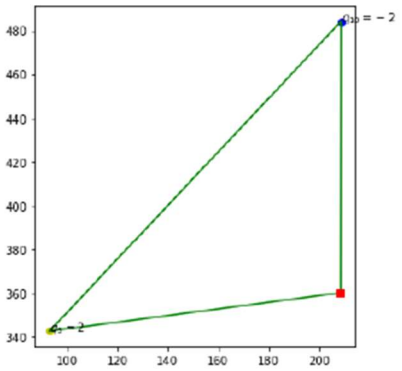
In [20]:
```
arcs__in_sol=[(i,j) for i,j in A if x[i,j].X>0.99] #Ευρεση ενεργών τόξων
print('Το ελάχιστο κόστος είναι '+str(round(m.ObjVal,2))+'.') # Εκτύπωση του ελάχιστου κόστους με 2 δεκαδικά ψηφία.
print(' ')
print('Report Διαδρομών')
print('------------------')
arcs__in_sol.sort()
all_routes=[]
while arcs__in_sol:
    route=[arcs__in_sol[0][0],arcs__in_sol[0][1]]
    next_node=arcs__in_sol[0][1]
    arcs__in_sol.remove(arcs__in_sol[0])
    flag=1
    while flag>0:
        next_ark=[(next_node,j) for j in V if (next_node,j) in arcs__in_sol]
        next_node=next_ark[0][1]
        route.append(next_node)
        arcs__in_sol.remove(next_ark[0])
        if next_node==2*n+1:
            all_routes.append(route)
            flag=-1

endtimes=[]
for i in range(len(all_routes)):
    diadromi=all_routes[i]
    print(' ')
    print('Δρομολόγιο '+str(i+1))
    for j in range(len(diadromi)):
        if j==0:
            out_node=max([ei[0],ei[diadromi[1]]-tij[0,diadromi[1]]])
            print('Το όχημα ξεκινά άδειο απο το σημείο εκκίνησης τη χρονική στιγμή '+str(round(out_node,2)))
        elif j>0 and j<len(diadromi)-1:
            point=diadromi[j]
            ppoint=diadromi[j-1]
            npoint=diadromi[j+1]
            in_node=max([ei[point],out_node+tij[ppoint,point]])
            out_node=in_node+di[point]
            if point<n+1:
                print('> Το όχημα εισέρχεται στον κόμβο '+str(point)+' τη χρονική στιγμή '+str(round(in_node,2))+
                      ' , φορτώνει '+str(q[point])+' μονάδες προιόντος και αποχωρέι τη χρονική στιγμή '+
                      str(round(out_node,2))+' κατευθηνόμενο προς τον κόμβο '+str(diadromi[j+1]))
            else:
                print('> Το όχημα εισέρχεται στον κόμβο '+str(point)+' τη χρονική στιγμή '+str(round(in_node,2))+
                      ' , ξεφορτώνει '+str(abs(q[point]))+' μονάδες προιόντος και αποχωρέι τη χρονική στιγμή '+
                      str(round(out_node,2))+' κατευθηνόμενο προς τον κόμβο '+str(npoint))
        else:
            route_time=out_node+tij[diadromi[j-1],diadromi[j]]
            endtimes.append(route_time)
            print('> Το όχημα γυρίζει άδειο στο σημείο τερματισμού τη χρονική στιγμή '+str(round(route_time,2)))
```

```
Το ελάχιστο κόστος είναι 1875.51.

Report Διαδρομών
------------------

Δρομολόγιο 1
Το όχημα ξεκινά άδειο απο το σημείο εκκίνησης τη χρονική στιγμή 0
> Το όχημα εισέρχεται στον κόμβο 3 τη χρονική στιγμή 36.83 , φορτώνει 2 μονάδες προιόντος και αποχωρέι τη χρονική στιγμή
37.79 κατευθηνόμενο προς τον κόμβο 4
> Το όχημα εισέρχεται στον κόμβο 4 τη χρονική στιγμή 115 , φορτώνει 1 μονάδες προιόντος και αποχωρέι τη χρονική στιγμή 1
15.48 κατευθηνόμενο προς τον κόμβο 1
> Το όχημα εισέρχεται στον κόμβο 1 τη χρονική στιγμή 128.83 , φορτώνει 7 μονάδες προιόντος και αποχωρέι τη χρονική στιγμ
ή 132.19 κατευθηνόμενο προς τον κόμβο 8
> Το όχημα εισέρχεται στον κόμβο 8 τη χρονική στιγμή 150 , ξεφορτώνει 2 μονάδες προιόντος και αποχωρέι τη χρονική στιγμή
150.96 κατευθηνόμενο προς τον κόμβο 6
> Το όχημα εισέρχεται στον κόμβο 6 τη χρονική στιγμή 226 , ξεφορτώνει 7 μονάδες προιόντος και αποχωρέι τη χρονική στιγμή
229.36 κατευθηνόμενο προς τον κόμβο 2
> Το όχημα εισέρχεται στον κόμβο 2 τη χρονική στιγμή 250.77 , φορτώνει 10 μονάδες προιόντος και αποχωρέι τη χρονική στιγ
μή 255.57 κατευθηνόμενο προς τον κόμβο 7
> Το όχημα εισέρχεται στον κόμβο 7 τη χρονική στιγμή 326.87 , ξεφορτώνει 10 μονάδες προιόντος και αποχωρέι τη χρονική στ
ιγμή 331.67 κατευθηνόμενο προς τον κόμβο 9
> Το όχημα εισέρχεται στον κόμβο 9 τη χρονική στιγμή 409.99 , ξεφορτώνει 1 μονάδες προιόντος και αποχωρέι τη χρονική στι
γμή 410.47 κατευθηνόμενο προς τον κόμβο 11
> Το όχημα γυρίζει άδειο στο σημείο τερματισμού τη χρονική στιγμή 440.56

Δρομολόγιο 2
Το όχημα ξεκινά άδειο απο το σημείο εκκίνησης τη χρονική στιγμή 20.59
> Το όχημα εισέρχεται στον κόμβο 5 τη χρονική στιγμή 47 , φορτώνει 2 μονάδες προιόντος και αποχωρέι τη χρονική στιγμή 4
7.96 κατευθηνόμενο προς τον κόμβο 10
> Το όχημα εισέρχεται στον κόμβο 10 τη χρονική στιγμή 123 , ξεφορτώνει 2 μονάδες προιόντος και αποχωρέι τη χρονική στιγμ
ή 123.96 κατευθηνόμενο προς τον κόμβο 11
> Το όχημα γυρίζει άδειο στο σημείο τερματισμού τη χρονική στιγμή 152.01
```

Γρφήματα διαδρομών

In [21]:
```
all_xs=[xs]+xp+xd+[xf] # Λίστα με όλα τα x
all_ys=[ys]+yp+yd+[yf] # Λίστα με όλα τα y

k=len(all_routes) # Αριθμός δρομολογίων

fig, axs = plt.subplots(k, 1, figsize=(5,6*k))

for kk in range(k):
    current_route=all_routes[kk]
    arcs_in_current_route=[]
    for i in range(len(current_route)-1):
        arcs_in_current_route.append((current_route[i],current_route[i+1]))

    for i,j in arcs_in_current_route:
        axs[kk].plot([all_xs[i],all_xs[j]],[all_ys[i],all_ys[j]],c='g') #Σχεδιαδμός της διαδρομής i->j
    axs[kk].plot(xs,ys,c='r',marker='s') # Τοποθέτηση του σημείου 0 (εκκίνηση-τερματισμός).
    axs[kk].plot(xf,yf,c='r',marker='s') # Τοποθέτηση του σημείου 0 (εκκίνηση-τερματισμός).


    P_in_route=[i for i in current_route if i<=n if i>0]
    D_in_route=[i+n for i in P_in_route]
    xp_in_route=[all_xs[i] for i in P_in_route]
    yp_in_route=[all_ys[i] for i in P_in_route]
    xd_in_route=[all_xs[i] for i in D_in_route]
    yd_in_route=[all_ys[i] for i in D_in_route]

    axs[kk].scatter(xp_in_route,yp_in_route,c='y') # Τοποθέτηση σημείων φόρτωσης
    axs[kk].scatter(xd_in_route,yd_in_route,c='b') # Τοποθέτηση σημείων παράδοσης
    for i in P_in_route:
        axs[kk].annotate('$q_{%d} = %d$'%(i,q[i]),(x_all[i],y_all[i])) #Εμφάνοση της ζήτησης σε κάθε πελάτη
    for i in D_in_route:
        axs[kk].annotate('$q_{%d} = %d$'%(i,q[i]),(x_all[i],y_all[i])) #Εμφάνοση της ζήτησης σε κάθε πελάτη
```





Α5 Κώδικας IRP

1) Εισαγωγή Βιβλιοθηκών

In [1]:
```
import numpy as np
from gurobipy import *
import matplotlib.pyplot as plt
```

2) Εισαγόμενα μεγέθη

In [2]:
```
seed_number=1 # Seed του τυχαίου προβλήματος
n=15 # Αριθμός πελατών
p=3 # Αριθμός χρονικών περιόδων
cap=500 # Μέγιστη χωρητικότητα οχημάτων (ομοιογενής στόλος)
max_x=1000 # Διάσταση x του διαγράμματος (απο 0 έως max_x)
max_y=1000 # Διάσταση y του διαγράμματος (απο 0 έως max_y)
b1=5 # Παράμετρος ορισμού της ζήτησης
b2=50 # Παράμετρος ορισμού της ζήτησης
move_cost=2 # Κόστος διάνυσης μίας μονάδας απόστασης
I0sup=0 # Αρχικό επίπεδο αποθήκης προμηθευτή
suplier_holding_cost=0 # Κόστος h του προμηθευτή
```

Τύποι πελατών

- 1 : Πελάτης με υψηλό κόστος αποθήκευσης και χαμηλό αποθηκευτικό χώρο
- 2 : Πελάτης με ισσοροπημένο κόστος αποθήκευσης και μέτριο αποθηκευτικό χώρο
- 3 : Πελάτης με χαμηλό κόστος αποθήκευσης και υψηλό αποθηκευτικό χώρο

In [3]:
```
pt1=0.2 # Διακριτή πιθανότητα πελάτη τύπου 1
pt2=0.6 # Διακριτή πιθανότητα πελάτη τύπου 2
pt3=0.2 # Διακριτή πιθανότητα πελάτη τύπου 3
a11=1.2 # Παράμετρος για τον υπολογισμο h πελατών τύπου 1
a12=1.8 # Παράμετρος για τον υπολογισμο h πελατών τύπου 1
a21=0.8 # Παράμετρος για τον υπολογισμο h πελατών τύπου 2
a22=1.2 # Παράμετρος για τον υπολογισμο h πελατών τύπου 2
a31=0.2 # Παράμετρος για τον υπολογισμο h πελατών τύπου 3
a32=0.8 # Παράμετρος για τον υπολογισμο h πελατών τύπου 3
```

In [4]:
```
stime=60 # Επιτρεπόμενος χρόνος επίλυσης IRP
vrp_time=10 # Επιτρεπόμενος χρόνος επίλυσης CVRP
svrp_time=2 # Επιτρεπόμενος χρόνος επίλυσης αρχικής λύσης CVRP
```

3) Υπολογιζόμενα μεγέθη

In [5]:
```
rnd=np.random
rnd.seed(seed_number)
```

In [6]:
```
dmax=round(cap/b1)  # Μέγιστη δυνατή ζήτηση
dmin=round(cap/b2)  # Ελάχιστη δυνατή ζήτηση
```

In [7]:
```
diag=(max_x**2+max_y**2)**(1/2) # Μήκος διαγωνίου
k=math.ceil((n*(dmax+dmin))/(2*cap)) # Αριθμός οχημάτων
h_m_cost=round((2*move_cost*k*diag)/(dmax*n),3) #Ισσοροπημένο κόστος διαχείρησης αποθέματος (αποθήκευσης)
csup=dmax*p*n # Χωρητικότητα αποθήκης προμηθευτή
rtmin=n*dmax # Ελάχιστη ποσότητα προιόντων διαθέσιμα για αποστολή ανα περίοδο
rtmax=round(1.5*n*dmax) # Μέγιστη ποσότητα προιόντων διαθέσιμα για αποστολή ανα περίοδο
I0min=0 #Ελάχιστο αρχικό επίπεδο αποθέματος
I0max=dmax+dmin #Μέγιστο αρχικό επίπεδο αποθέματος
```

In [8]:
```
a13=(1+dmin/dmax) # Παράμετρος για τον υπολογισμο χωρητικότητας αποθήκης πελατών τύπου 1
a23=p/2 # Παράμετρος για τον υπολογισμο χωρητικότητας αποθήκης πελατών τύπου 2
a33=p # Παράμετρος για τον υπολογισμο χωρητικότητας αποθήκης πελατών τύπου 3
```

4) Δημιουργία τυχαίου προβλήματος

In [9]:
```
T=[i+1 for i in range(p)] # Το σύνολο των χρονικών περιόδων
```

In [10]:
```
xc=rnd.rand(n+1)*max_x # Δημιουργία τυχαίων x
yc=rnd.rand(n+1)*max_y # Δημιουργία τυχαίων y
N=[i for i in range(1,n+1)] #Το σύνολο των πελατών
V=[0]+N #Το σύνολο των κόμβων
A=[(i,j) for i in V for j in V if i!=j] #Όλες οι πιθανές διαδρομές i->j
c={(i,j): np.hypot(xc[i]-xc[j],yc[i]-yc[j])*move_cost for i,j in A} #Τα κόστη μετάβασης
```

In [11]:
```
K=[i+1 for i in range(k)] #Σύνολο οχημάτων
Qk=dict(zip(K,[cap for i in K])) # Λεξικό με τις χωρητικότητες κάθε οχήματος
```

In [12]:
```
types=[1,2,3]
probabilities=[pt1,pt2,pt3]
customer_types=rnd.choice(types,size=n,p=probabilities)
```

In [13]:
```
holding_costs=[suplier_holding_cost]
for i in range(len(N)):
    if customer_types[i]==1: # Εάν ο πελάτης είναι τύπου 1 το κόστος αποθήκευσης +20% με +80% του ισοροπημένου
        hc=round(rnd.uniform(a11*h_m_cost,a12*h_m_cost),3)
        holding_costs.append(hc)
    elif customer_types[i]==2: # Εάν ο πελάτης είναι τύπου 2 το κόστος αποθήκευσης -20% με +20% του ισοροπημένου
        hc=round(rnd.uniform(a21*h_m_cost,a22*h_m_cost),3)
        holding_costs.append(hc)
    else: # Εάν ο πελάτης είναι τύπου 3 το κόστος αποθήκευσης -80% με -20% του ισοροπημένου
        hc=round(rnd.uniform(a31*h_m_cost,a32*h_m_cost),3)
        holding_costs.append(hc)
hi=dict(zip(V,holding_costs)) # Λεξικό με τα holding costs κάθε πελάτη
```

In [14]:
```
customer_rcaps=[csup]
for i in range(len(N)):
    if customer_types[i]==1: # Εάν ο πελάτης είναι τύπου 1 έχει μέγεθος αποθήκης dmax + dmin
        rcap=round(dmax*a13)
        customer_rcaps.append(rcap)
    elif customer_types[i]==2: # Εάν ο πελάτης είναι τύπου 2 έχει μέγεθος αποθήκης dmax * p/2
        rcap=round(dmax*a23)
        customer_rcaps.append(rcap)
    else: # Εάν ο πελάτης είναι τύπου 3 έχει μέγεθος αποθήκης dmax dmax * p
        rcap=round(dmax*a33)
        customer_rcaps.append(rcap)
Ci=dict(zip(V,customer_rcaps)) # Λεξικό με τα όρια αποθήκης κάθε πελάτη
```

In [15]:
```
demands_of_each_customer=[] # Η λίστα με τις ζητήσεις όλων των πελατών
for i in N:
    # Δημιουργία τυχαίων ζητήσεων ενός πελάτη για κάθε περίοδο
    demands_of_customer_i=[rnd.randint(dmin,dmax+1) for j in T]
    demands_of_each_customer.append(demands_of_customer_i)
dti=dict(zip(N,demands_of_each_customer)) # Λεξικό με τις ζητήσεις κάθε πελάτη για τον χρονικό ορίζοντα
```

In [16]:
```
# Δημιουργία αρχικών επιπέδων αποθήκης για κάθε πελάτη
customer_starting_inv=[I0sup]+[rnd.randint(I0min,I0max+1) for i in N]
I0i=dict(zip(V,customer_starting_inv)) # Λεξικό με τα αρχικά επίπεδα αποθήκης κάθε πελάτη
```

In [17]:
```
# Δημιουργία τυχαίων διαθέσιμων ποσοτήτων προς αποστολή ανα περίοδο
available_quantity=[rnd.randint(rtmin,rtmax+1) for i in T]
rt=dict(zip(T,available_quantity))
```

5) Κατασκευή του μοντέλου

Δημιουργία κενού Model

In [18]:
```
m=Model('IRP')
```

```
Using license file C:\Users\thana\gurobi.lic
Academic license - for non-commercial use only - expires 2021-07-30
```

Ορισμός των μεταβλητών

In [19]:
```
# Δείκτες θα είναι της μορφής i,j,k,t
list_of_indexes_1=[]
for vec in K:
    for t in T:
        for (i,j) in A:
            if i<j:
                list_of_indexes_1.append((i,j,vec,t))
x=m.addVars(list_of_indexes_1,vtype=GRB.INTEGER,ub=2,name='x') # Η μεταβλητή x

# Οι δείκτες θα είναι της μορφής v,k,t
list_of_indexes_2=[]
for v in V:
    for vec in K:
        for t in T:
            list_of_indexes_2.append((v,vec,t))
y=m.addVars(list_of_indexes_2,vtype=GRB.BINARY,name='y') # Η μεταβλητή y
q=m.addVars(list_of_indexes_2,vtype=GRB.INTEGER,name='q') # Η μεταβλητή q

# Οι δείκτες θα είναι της μορφής v,t
list_of_indexes_3=[]
for v in V:
    for t in [0]+T:
        list_of_indexes_3.append((v,t))
I=m.addVars(list_of_indexes_3,vtype=GRB.INTEGER,name='I') # Η μεταβλητή I
```

Ορισμός των περιορισμών

In [20]:
```
con0=m.addConstrs((I[i,0]==I0i[i] for i in V),name='con0') # Εκχώρηση αρχικών επίπεδω αποθήκης
con1=m.addConstrs((I[0,t-1]-I[0,t]+rt[t]-quicksum(q[i,vec,t] for i in N for vec in K)==0 for t in T),name='con1') # Περ 1
con2=m.addConstrs((I[i,t-1]-I[i,t]+quicksum(q[i,vec,t] for vec in K)-dti[i][t-1]==0 for i in N for t in T),name='con2') #2
#con3=m.addConstrs((I[i,t]>=0 for i in V for t in T),name='con3') # Περ 3 (λόγω ορισμού του I όχι απαραίτητος)
con4=m.addConstrs((I[i,t]<=Ci[i] for i in V for t in T),name='con4') # Περ 4
con5=m.addConstrs((quicksum(q[i,vec,t] for vec in K)<=Ci[i]-I[i,t-1] for i in N for t in T),name='con5') # Περ 5
con6=m.addConstrs((q[i,vec,t]<=Ci[i]*y[i,vec,t] for i in N for vec in K for t in T),name='con6') # Περ 6
con7=m.addConstrs((quicksum(q[i,vec,t] for i in N)<=Qk[vec]*y[0,vec,t] for vec in K for t in T),name='con7') # Περ 7
con8=m.addConstrs((quicksum(x[i,j,vec,t] for j in V if i<j)+quicksum(x[j,i,vec,t] for j in V if j<i)==
                   2*y[i,vec,t] for i in V for vec in K for t in T),name='con8') # Περ 8
con9=m.addConstrs(x[i,j,vec,t]<=1 for (i,j,vec,t) in list_of_indexes_1 if i!=0 if j!=0) # Περ
```

Ορισμός της αντικειμενικής συνάρτησης

In [21]:
```
m.setObjective(quicksum(hi[v]*I[v,t] for (v,t) in list_of_indexes_3)+
               quicksum(c[i,j]*x[i,j,vec,t] for (i,j,vec,t) in list_of_indexes_1),GRB.MINIMIZE)
```

6) Ορισμός της συνάρτησης callback

In [22]:
```
def subtourelim(model, where):
    if where == GRB.Callback.MIPSOL:
        x_vals = model.cbGetSolution(model._x) # Τιμές της x
        y_vals = model.cbGetSolution(model._y) # Τιμές της y
        for vec in K:
            for t in T:
                index_list=[(i,j,kk,tt) for (i,j,kk,tt) in list_of_indexes_1 if kk==vec if tt==t] # Βρίσκω τους δείκτες γι
                active_arcss=[] # Λίστα με τα ενεργά τόξα στην λύση                              # συγκεκριμένο vec και t
                x_2=[] # Λίστα με τα x που πέρνουνε τιμή =2 (αρχίζουν πάντα απο το 0)

                for index in index_list:
                    if x_vals[index]>0.5:
                        active_arcss.append(index)
                    if x_vals[index]>1.5:
                        x_2.append(index)

                if len(active_arcss)>0: # Εάν για το συγκεκριμένο vec την συγκεκριμένη t υπάρχουν διαδρομές
                    list_of_S=findroutes(active_arcss,x_2) # Επιστρέφει μία λίστα με λίστες που περιέχουν τα S
                    for s in list_of_S:
                        con_arks=[]
                        s.sort()
                        for pnt in range(len(s)):
                            c_pnt=s[pnt]
                            if pnt<len(s)-1:
                                for opnts in s[pnt+1:]:
                                    con_arks.append((c_pnt,opnts))

                        model.cbLazy(quicksum(model._x[i,j,vec,t] for i,j in con_arks)
                                     <=quicksum(model._y[i,vec,t] for i in s)-y[s[0],vec,t]) # Περ 10
```

In [23]:
```
def findroutes(arcs, list_of_2s):
    unassigned=arcs[:]
    routes=[] # Η λίστα με τις διαδρομές
    while unassigned: # Σταματάει όταν unassigned=[]
        start=unassigned[0]
        next_point=start[1]
        route=[next_point] # Η λίστα που περιέχει τα σημεία της διαδρομής
        unassigned.remove(start)

        flag=-1 # Όταν πάρει θετική τιμή θα έχει βρεθεί διαδρομή
        if start in list_of_2s: # Περίπτωση εξυπηρέτησης μόνο ενός πελάτη απο το depot
            flag=1
            #routes.append(route)
        while flag<0:

            arc_with_new_next_point=[idx for idx in unassigned if idx[0]==next_point] # Έυρεση του επόμενου σημείου εάν ί
            if len(arc_with_new_next_point)==0:
                arc_with_new_next_point=[idx for idx in unassigned if idx[1]==next_point] # Έυρεση του επόμενου σημείου εά
                if len(arc_with_new_next_point)==0:
                    next_point=0 # Εάν δεν βρεθεί τίποτα το επόμενο σημείο είναι το 0
                else:
                    next_point=arc_with_new_next_point[0][0]
                    unassigned.remove(arc_with_new_next_point[0])
            else:
                next_point=arc_with_new_next_point[0][1]
                unassigned.remove(arc_with_new_next_point[0])

            if next_point==0: # Εύρεση διαδρομής
                #routes.append(route)
                flag=1
            elif next_point==start[0]: # Εύρεση κλειστής διαδρομής
                route.append(next_point)
                routes.append(route)
                flag=1
            else:
                route.append(next_point)

    return routes
```

7) Εύρεση αρχικής εφικτής λύσης

In [24]:
```
def find_vrp_routes_from_arcs(arklist):
    vrp_routes=[]
    arklist.sort()
    for i in range(k):
        startin_arc=arklist[i]
        croute=[startin_arc[0],startin_arc[1]]
        next_vrp_node=startin_arc[1]
        rfound=-1
        while rfound<0:
            next_vrp_arc=[a for a in arklist if a[0]==next_vrp_node][0]
            if next_vrp_arc[1]==startin_arc[0]:
                croute.append(next_vrp_arc[1])
                rfound=1
            else:
                croute.append(next_vrp_arc[1])
                next_vrp_node=next_vrp_arc[1]
        vrp_routes.append(croute)
    return vrp_routes
```

In [25]:
```
def subtourelim1(model, where):

    t=tt
    N1=[i for i in N if vrp_demands[i][t]>0]
    V1=[0]+N1
    A1=[(i,j) for i in V1 for j in V1 if i!=j]

    if where == GRB.Callback.MIPSOL:
        # make a list of edges selected in the solution
        vals = model.cbGetSolution(model._vars)
        selected = [(i, j) for i, j in model._vars.keys() if vals[i,j]>0.5]
        arklist=selected[:]
        arklist.sort()
        while arklist:
            (sta,nex)=arklist[0]
            arklist.remove(arklist[0])
            route=[sta,nex]
            flag=1
            while flag>0:
                if nex==0: # Εύρεση διαδρομής που επικοινωνεί με το 0
                    rute=route[1:len(route)-1]
                    rs1=math.ceil(sum([vrp_demands[i][t] for i in rute])/cap)
                    if rs1>1:
                        #Περιορισμός 4
                        model.cbLazy(quicksum(model._vars[i,j] for i,j in A1 if i in rute if j not in rute)>=rs1)
                    flag=-1

                elif nex==sta: # Εύρεση διαδρομής που δεν επικοινωνεί με το 0
                    rute=route[:len(route)-1]
                    rs2=math.ceil(sum([vrp_demands[i][t] for i in rute])/cap)
                    #Περιορισμός 4
                    model.cbLazy(quicksum(model._vars[i,j] for i,j in A1 if i in rute if j not in rute)>=rs2)
                    flag=-1

                else:
                    next_ark=[(nex,j) for j in V1 if (nex,j) in arklist]
                    (pre,nex)=next_ark[0]
                    arklist.remove(next_ark[0])
                    route.append(nex)
```

In [26]:
```
def find_vrp_demands(irp_demands,I0s):
    vrp_customer_demands=[]
    for i in N:
        vrp_customer=irp_demands[i]
        starting_inv=I0s[i]
        vrpcd=[]
        for j in range(len(vrp_customer)):
            if starting_inv>vrp_customer[j]:
                starting_inv=starting_inv-vrp_customer[j]
                vrpcd.append(0)
            else:
                vrpcd.append(vrp_customer[j]-starting_inv)
                starting_inv=0
        vrp_customer_demands.append(vrpcd)
    vrpdti=dict(zip(N,vrp_customer_demands))
    return vrpdti
```

In [27]:
```
def vrp_period_active_arcs(t):
    N1=[i for i in N if vrp_demands[i][t]>0]
    V1=[0]+N1
    A1=[(i,j) for i in V1 for j in V1 if i!=j]

    vrp=Model("Capacitated Vehicle Routing Problem") # Εισαγωγή μοντέλου
    x=vrp.addVars(A1,vtype=GRB.BINARY,name='x') # Εισαγωγή μεταβλητής xij
    con1=vrp.addConstrs((quicksum(x[i,j] for j in V1 if j!=i)==1 for i in N1),name='con1') #Περιορισμός 1
    con2=vrp.addConstrs((quicksum(x[i,j] for i in V1 if i!=j)==1 for j in N1),name='con2') #Περιορισμός 2
    con3=vrp.addConstr((quicksum(x[0,j] for j in N1)==len(K)),name="con3")
    vrp.setObjective(quicksum(x[i,j]*c[i,j] for i,j in A1),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής

    ssm=Model("Starting Solution Model")
    rtv=ssm.addVars(K,vtype=GRB.INTEGER,name="rtv")
    AV1=[(i,j) for i in K for j in N1]
    acus=ssm.addVars(AV1,vtype=GRB.BINARY,name="acus")
    sscon1=ssm.addConstrs((rtv[i]==quicksum(vrp_demands[j][t]*acus[i,j] for j in N1) for i in K),name="sscon1")
    sscon2=ssm.addConstrs((quicksum(acus[i,j] for i in K)==1 for j in N1),name="sscon2")
    sscon3=ssm.addConstrs((rtv[i]<=cap for i in K),name="sscon3")
    sscon4=ssm.addConstrs((rtv[i]>=1 for i in K),name="sscon4")
    ssm.setObjective(k*cap-quicksum(rtv[i] for i in K),sense=GRB.MINIMIZE) #Ορισμός της αντικειμενικής
    ssm.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
    ssm.Params.TimeLimit=svrp_time #Οριο χρόνου
    ssm.optimize()
    arks_in_ssol=[]
    arks_not_in_ssol=A1[:]
    for veh in K:
        ssroute=[0]
        for cu in N1:
            if acus[veh,cu].getAttr('x')>0.5:
                ssroute.append(cu)
        ssroute.append(0)
        if len(ssroute)>2:
            for i in range(len(ssroute)-1):
                arc=(ssroute[i],ssroute[i+1])
                arks_in_ssol.append(arc)
                arks_not_in_ssol.remove(arc)

    for i,j in arks_in_ssol:
        x[i,j].setAttr("Start",1) # VarHintVal Start
    for i,j in arks_not_in_ssol:
        x[i,j].setAttr("Start",0)

    vrp.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
    vrp.Params.TimeLimit=vrp_time #Οριο χρόνου
    vrp._vars = x
    vrp.Params.lazyConstraints = 1
    vrp.optimize(subtourelim1)

    names_of_vars=vrp.VarName # Λίστα με τα ονόματα των μεταβλητών
    values_in_solution=vrp.X # Οι τιμές των μεταβλητών στην λύση
    solution_values_dict=dict(zip(names_of_vars,values_in_solution))
    vrp_active_arcs=[a for a in A1 if x[a].x>0.99] #Ευρεση ενεργών τόξων
    objectiv=vrp.getAttr("ObjVal")

    del vrp
    del ssm

    return vrp_active_arcs,objectiv
```

In [28]:
```
vrp_demands=find_vrp_demands(dti,I0i)
vrp_total_cost=0
vrp_arcs_per_period=[]
all_vrp_routes=[]
for tt in range(p):
    (vrp_t_arks,vrp_period_cost)=vrp_period_active_arcs(tt)
    vrp_total_cost=vrp_total_cost+vrp_period_cost
    vrp_arcs_per_period.append(vrp_t_arks)
    vrp_period_routes=find_vrp_routes_from_arcs(vrp_t_arks)
    all_vrp_routes.append(vrp_period_routes)
    vrp_route_num=0
    for r in vrp_period_routes:
        vrp_route_num=vrp_route_num+1
```

In [29]:
```
def find_irp_arcs(rooot):
    list_of_irp_arcs=[]

    if len(rooot)<4:
        ghr=len(rooot)-2
    else:
        ghr=len(rooot)-1

    for i in range(ghr):
        if rooot[i]<rooot[i+1]:
            list_of_irp_arcs.append((rooot[i],rooot[i+1]))
        else:
            list_of_irp_arcs.append((rooot[i+1],rooot[i]))

    return list_of_irp_arcs
```

In [30]:
```
for t in T:
    for vec in K:
        ssroute=all_vrp_routes[t-1][vec-1] # [0,1,2,..,0]
        irp_arcs_in_route=find_irp_arcs(ssroute) # Τα τόξα της διαδρομή ενός οχήματος μία περίοδο
        irp_arcs_not_in_route=[(ii,jj,kk,tt) for (ii,jj,kk,tt) in list_of_indexes_1
                               if kk==vec if tt==t if (ii,jj) not in irp_arcs_in_route]

        ### Αρχικές τιμές του x
        for ii,jj in irp_arcs_in_route:
            if len(irp_arcs_in_route)==1:
                x[ii,jj,vec,t].setAttr("Start",2)
            else:
                x[ii,jj,vec,t].setAttr("Start",1)

        for aa in irp_arcs_not_in_route:
            x[aa].setAttr("Start",0)

        ### Αρχικές τιμές του y και q
        y_index_in=[(0,vec,t)]+[(ii,vec,t) for ii in ssroute if ii!=0]
        y_index_not_in=[(ii,kk,tt) for (ii,kk,tt) in list_of_indexes_2
                        if kk==vec if tt==t if (ii,kk,tt) not in y_index_in]

        for aa in y_index_in:
            y[aa].setAttr("Start",1)
            if aa[0]>0:
                q[aa].setAttr("Start",vrp_demands[aa[0]][t-1])
            else:
                q[aa].setAttr("Start",0)

        for aa in y_index_not_in:
                y[aa].setAttr("Start",0)
                q[aa].setAttr("Start",0)
```

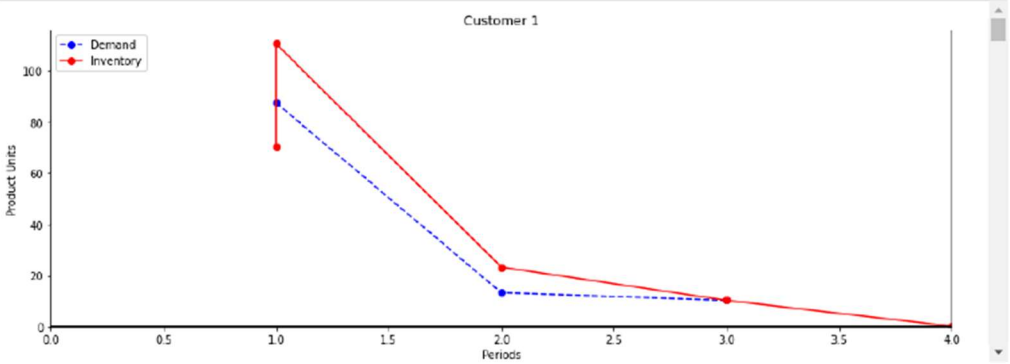
8) Επίλυση του IRP

In [31]:
```
m.Params.LogToConsole=0 # Απενεργοποίηση του output της κονσόλας
m.Params.TimeLimit=stime #Όριο χρόνου
m._x = x
m._y= y
m.Params.lazyConstraints = 1
m.optimize(subtourelim)
```

9) Επιθεώρηση της λύσης

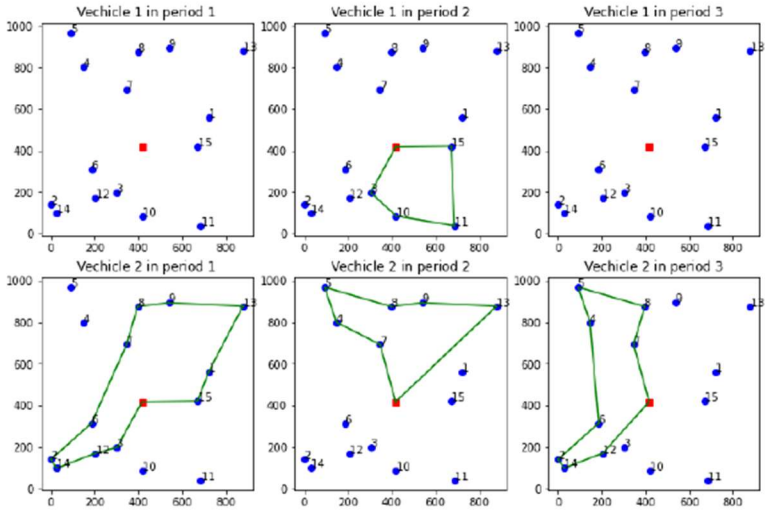
Διαγράμματα αποθεμάτων

In [32]:
```
fig, axs = plt.subplots(n, 1, figsize=(15,6*n))

for customer in N:

    time=T
    demand=[dti[customer][t-1] for t in T]
    customer_inventory=[I[customer,t].getAttr('X') for t in [0]+T]

    customer_supply=[]
    for t in T:
        supplied=0
        for k in K:
            if q[customer,k,t].getAttr('X')>0.5:
                supplied=1
                customer_supply.append(q[customer,k,t].getAttr('X'))
        if supplied==0:
            customer_supply.append(0)

    plan=[customer_inventory[0]]
    plan_time=[1]
    for t in T:
        plan.append(customer_inventory[t-1]+customer_supply[t-1])
        plan_time.append(t)
        plan.append(customer_inventory[t])
        plan_time.append(t+1)



    axs[customer-1].plot(time,demand,'bo--',label='Demand')
    axs[customer-1].plot(plan_time,plan,'ro-',label='Inventory')
    axs[customer-1].set_xlabel('Periods')
    axs[customer-1].set_ylabel('Product Units')
    axs[customer-1].axis([0, p+1, 0, max(plan)+0.05*max(plan)])
    axs[customer-1].legend(loc="upper left")
    axs[customer-1].set_title('Customer '+str(customer))
```



Γρφήματα δικτύου και διαδρομών ανά περίοδο

In [33]:
```
figs, axss = plt.subplots(k, p, figsize=(2*p*k,4*k))

for vec in K:
    for t in T:

        axss[vec-1][t-1].set_title('Vechicle '+str(vec)+' in period '+str(t))
        axss[vec-1][t-1].plot(xc[0],yc[0],c='r',marker='s') # Τοποθέτηση του σημείου 0 (κέντρο διανομής)
        axss[vec-1][t-1].scatter(xc[1:],yc[1:],c='b') # Τοποθέτηση των πελατών
        for i in N:
            axss[vec-1][t-1].annotate('%d '%(i),(xc[i]+1,yc[i]+1)) #Εμφάνοση της ζήτησης σε κάθε πελάτη

        active_arks=[]
        set_of_interest=[(i,j,kk,tt) for (i,j,kk,tt) in list_of_indexes_1 if kk==vec if tt==t]
        for indx in set_of_interest:
            if x[indx].getAttr('X')>0.5:
                axss[vec-1][t-1].plot([xc[indx[0]],xc[indx[1]]],[yc[indx[0]],yc[indx[1]]],c='g')
```



Παράρτημα Β -Δεδομένα και λύσεις προβλημάτων

Β1 Δεδομένα και λύσεις του CVRP

Δεδομένα προβλημάτων VRP 10 πελατών

Πελάτης	X	Y	Ζήτηση $b1$=4 b2=10	Ζήτηση $b1$=10 b2=40
0	109.76	52.89	0	0
1	143.04	56.80	91	29
2	120.55	92.56	69	39
3	108.98	7.10	59	29
4	84.73	8.71	59	29
5	129.18	2.02	54	24
6	87.52	83.26	79	17
7	178.35	77.82	72	10
8	192.73	87.00	41	11
9	76.69	97.86	49	19
10	158.35	79.92	97	35

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0, 2, 10, 8, 7, 1, 0
- 0, 5, 3, 4, 9, 6, 0

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0 , 6, 9, 2, 10, 8, 7, 1, 5, 3, 4, 0

Δεδομένα προβλημάτων VRP 20 πελατών

Πελάτης	X	Y	Ζήτηση $b1$=4 b2=10	Ζήτηση $b1$=10 b2=40
0	188.93	31.54	0	0
1	104.37	36.37	74	12
2	82.93	57.02	88	26
3	52.91	43.86	69	39
4	154.85	98.84	43	13
5	91.23	10.20	74	12
6	113.69	20.89	82	20
7	3.76	16.13	53	23
8	123.53	65.31	88	26
9	122.42	25.33	79	17
10	123.39	46.63	61	31
11	188.75	24.44	49	19
12	136.36	15.90	40	10
13	71.90	11.04	50	20
14	87.41	65.63	90	28
15	139.53	13.82	83	21
16	12.05	19.66	98	36
17	133.35	36.87	63	33
18	134.13	82.10	99	37
19	42.08	9.71	42	12
20	25.79	83.79	97	35

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0, 3, 16, 7, 19, 13, 5, 0
- 0, 11, 15, 12, 6, 9, 0
- 0, 14, 20, 2, 1, 0

- 0, 17, 10, 8, 18, 4, 0

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0, 4, 18, 8, 14, 2, 3, 20, 7, 16, 19, 13, 5, 1, 6, 6, 9, 12, 15, 0
- 0, 10, 17, 11, 0

Δεδομένα προβλημάτων VRP 30 πελατών

Πελάτης	X	Y	Ζήτηση $b1$=4 b2=10	Ζήτηση $b1$=10 b2=40
0	63.60	84.64	0	0
1	82.85	69.95	78	16
2	12.83	29.74	75	13
3	138.49	81.38	62	32
4	113.32	39.65	45	40
5	53.08	88.11	63	39
6	104.65	58.13	83	15
7	18.79	88.17	72	33
8	115.19	69.25	51	21
9	185.86	72.53	80	10
10	63.71	50.13	60	21
11	133.48	95.61	50	18
12	26.36	64.40	94	30
13	143.27	42.39	83	20
14	57.88	60.64	77	32
15	36.64	1.92	68	21
16	117.30	30.16	87	15
17	4.02	66.02	80	39
18	165.79	29.01	42	38
19	0.94	61.80	67	25
20	135.56	42.88	59	18
21	54.00	13.55	65	12
22	147.04	29.83	63	37
23	192.44	57.00	93	29
24	49.75	59.09	91	35
25	115.23	57.43	86	33
26	118.41	65.32	60	31
27	114.45	65.21	93	29
28	44.62	43.14	69	24
29	190.55	89.65	43	30
30	89.43	36.76	75	31

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0, 2, 15, 21, 28, 24, 0
- 0, 3, 29, 9, 23, 18, 22, 0
- 0, 8, 27, 26, 25, 6, 0
- 0, 12, 19, 17, 7, 5, 0
- 0, 14, 10, 1, 0
- 0, 30, 4, 16, 20, 13, 11, 0

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0, 1, 8, 27, 26, 25, 6, 30, 21, 15, 2, 19, 17, 12, 7, 5, 0
- 0, 14, 24, 28, 10, 4, 16, 20, 13, 22, 18, 23, 9, 29, 3, 11, 0

Δεδομένα προβλημάτων VRP 50 πελατών

Πελάτης	X $b1$=4, b2=10	Y $b1$=4, b2=10	Ζήτηση $b1$=4, b2=10	X $b1$=10, b2=40	Y $b1$=10, b2=40	Ζήτηση $b1$=10, b2=40	X $b1$=20, b2=100	Y $b1$=20, b2=100	Ζήτηση $b1$=20, b2=100
0	6.64	71.47	0	113.82	53.66	0	83.40	67.88	0
1	196.51	46.95	49	81.44	89.67	36	144.06	21.16	9
2	74.66	45.60	84	13.83	99.03	27	0.02	26.55	17
3	84.02	90.64	53	139.49	21.69	20	60.47	49.16	5
4	10.12	13.72	97	90.71	66.31	19	29.35	5.34	14
5	73.10	22.92	46	144.41	26.33	22	18.47	57.41	4
6	3.33	88.16	78	173.28	2.07	23	37.25	14.67	11
7	46.15	90.44	79	195.10	75.84	35	69.11	58.93	4
8	152.98	64.58	48	171.16	32.00	16	79.35	69.98	18
9	188.82	32.47	53	2.34	38.35	16	107.76	10.23	17
10	150.00	51.97	96	72.00	58.83	17	83.84	41.41	15
11	67.88	0.01	47	146.00	83.10	18	137.04	69.44	10
12	97.91	31.19	56	34.33	62.90	23	40.89	41.42	17
13	67.80	42.55	96	104.21	87.27	34	175.62	5.00	19
14	35.90	88.53	62	10.87	27.35	17	5.48	53.59	13
15	34.20	67.99	96	40.00	79.80	26	134.09	66.38	6
16	92.69	45.61	55	3.70	18.56	34	83.46	51.49	11
17	174.91	48.34	65	158.74	95.28	32	111.74	94.46	9
18	188.82	78.87	48	44.78	68.75	34	28.08	58.66	8
19	121.65	22.94	75	69.07	21.55	25	39.62	90.34	9
20	119.33	88.03	46	185.62	94.74	35	160.15	13.75	12
21	156.73	31.37	57	140.88	73.09	18	193.65	13.93	17
22	100.01	95.75	47	6.37	25.39	13	62.68	80.74	19
23	10.07	47.18	60	32.94	21.33	16	138.46	39.77	17
24	139.82	71.16	65	124.30	51.82	27	175.28	16.54	12
25	198.48	15.37	42	115.45	2.57	17	178.92	92.75	18
26	53.45	73.04	100	47.58	20.75	30	17.01	34.78	17
27	135.82	64.63	56	186.84	42.47	35	7.81	75.08	20
28	172.86	21.49	96	122.79	37.42	12	33.97	72.60	14
29	150.17	18.65	90	107.13	46.36	38	175.63	88.33	17
30	192.90	80.76	85	117.98	27.76	26	19.67	62.37	7
31	110.85	74.71	63	146.02	58.68	34	84.22	75.09	6
32	42.48	67.48	64	62.39	86.39	28	191.58	34.89	18
33	44.49	27.69	44	79.64	11.75	23	106.63	26.99	18
34	43.75	17.49	76	41.97	51.74	33	138.38	89.59	4
35	113.91	70.45	84	37.24	13.21	34	63.10	42.81	15
36	90.42	46.32	89	188.87	71.69	14	137.30	96.48	7
37	194.05	84.04	63	147.91	39.61	14	166.93	66.34	5
38	136.11	20.49	70	98.09	56.54	22	3.66	62.17	6
39	17.06	16.50	94	45.48	18.33	27	150.03	11.47	20
40	11.28	12.48	65	50.87	14.48	33	197.77	94.95	4
41	97.57	72.21	60	11.61	48.81	40	149.63	44.99	14
42	176.20	3.05	73	86.88	35.56	32	56.09	57.84	12
43	195.28	74.70	77	62.36	94.04	35	157.86	40.81	17
44	123.53	9.26	97	139.27	76.53	30	20.65	23.70	7
45	108.50	21.75	63	75.55	74.87	11	89.58	90.34	4
46	170.92	74.93	67	35.92	90.37	15	181.72	57.37	12
47	148.77	73.17	69	4.94	8.34	35	58.72	0.29	17
48	95.72	4.56	73	13.45	55.22	33	57.56	61.71	19
49	135.42	20.92	91	135.88	58.45	37	26.01	32.66	13
50	121.41	28.69	93	90.74	96.19	39	3.87	52.71	4

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0, 3, 22, 20, 31, 35, 26, 0
- 0, 6, 14, 7, 32, 15, 0
- 0, 8, 43, 30, 37, 18, 46, 0
- 0, 13, 2, 12, 16, 36, 0
- 0, 17, 9, 25, 1, 47, 24, 27,0
- 0, 23, 4, 40, 39, 34, 0
- 0,41,10,21,28,42,0
- 0,49,38,29,44,5,0
- 0,50,19,45,48,11,33,0

Τα δρομολόγια στην καλύτερη λύση για $b1$=10, $b2$=40 είναι:

- 0, 19, 49, 26, 39, 35, 23, 47, 16, 22, 14, 9, 41, 48, 12, 45, 0
- 0, 24, 28, 30, 25, 33, 42, 29, 0
- 0, 38, 4, 10, 34, 18, 15, 2, 46, 43, 32, 1, 50, 13, 0
- 0, 49, 31, 21, 44, 11, 17, 20, 7, 36, 27, 8, 6, 3, 5, 37, 0

Τα δρομολόγια στην καλύτερη λύση για $b1$=20, $b2$=1000 είναι:

- 0, 7, 48, 42, 3, 35, 12, 6, 47, 4, 44, 49, 26, 2, 50, 14, 38, 27, 30, 5, 18, 28, 19, 22, 8, 0
- 0, 31, 45, 17, 36, 34, 11, 15, 23, 41, 43, 37, 29, 25, 40, 46, 32, 21, 13, 24, 20, 39, 1, 9, 33, 10, 16, 0

Δεδομένα προβλημάτων VRP 100 πελατών

Πελάτης	X $b1$=4, b2=10	Y $b1$=4, b2=10	Ζήτηση $b1$=4, b2=10	X $b1$=10, b2=40	Y $b1$=10, b2=40	Ζήτηση $b1$=10, b2=40	X $b1$=20, b2=100	Y $b1$=20, b2=100	Ζήτηση $b1$=20, b2=100
0	156.09	8.50	0	69.45	90.88	0	2.51	0.04	0
1	197.01	9.45	66	29.63	81.55	38	14.39	97.68	9
2	150.71	63.36	44	196.37	15.94	31	193.46	37.66	9
3	0.81	94.93	81	95.67	62.89	14	113.62	97.38	11
4	53.90	93.67	50	99.48	39.84	36	40.66	60.47	18
5	82.10	44.74	52	127.89	6.27	14	50.47	82.88	9
6	85.64	72.77	80	73.72	42.40	19	148.77	57.47	13
7	59.57	74.22	73	27.38	25.87	20	39.09	62.81	18
8	80.23	30.70	77	164.42	84.90	22	116.27	28.56	5
9	24.13	11.98	79	37.97	3.33	18	194.00	58.68	17
10	196.14	44.39	72	102.26	95.90	11	169.37	75.00	15
11	81.22	39.18	98	44.86	35.54	15	47.97	85.83	18
12	113.84	53.18	73	19.57	35.67	17	98.75	75.51	13
13	68.72	84.54	66	172.44	1.63	10	123.99	69.81	7
14	157.77	53.63	75	194.58	18.52	36	165.80	86.45	7
15	82.27	68.03	65	192.17	40.13	11	31.36	32.27	7
16	71.85	60.92	90	181.31	92.93	36	3.72	67.08	5
17	79.90	9.85	64	154.81	9.96	13	14.00	45.09	7
18	60.37	9.20	58	66.63	94.53	35	97.27	38.21	20
19	155.04	5.60	60	16.22	86.95	28	121.27	41.08	9

20	185.24	8.65	70	81.45	45.42	34	113.77	40.15	4
21	65.06	23.72	68	46.45	32.67	28	63.47	31.74	9
22	190.57	83.95	48	26.50	23.27	30	197.72	62.19	18
23	2.79	52.24	58	10.69	61.45	40	115.95	43.02	11
24	106.69	51.31	65	145.12	3.31	38	76.03	97.38	10
25	60.92	64.98	64	2.29	1.56	18	110.19	67.78	14
26	176.57	54.46	79	154.12	42.88	28	149.07	19.86	4
27	50.12	3.25	82	29.39	6.81	35	133.85	42.67	18
28	135.48	58.02	99	15.90	25.19	34	52.98	34.33	8
29	162.08	77.11	68	17.92	22.12	17	13.27	79.76	11
30	86.43	37.62	64	134.41	25.32	20	74.02	88.00	11
31	150.43	49.10	58	49.07	13.11	37	125.94	90.38	14
32	165.92	98.16	43	84.11	1.20	38	42.03	66.27	16
33	75.81	24.47	84	111.47	11.55	34	150.55	27.02	11
34	19.31	37.43	59	172.11	61.85	28	13.31	25.24	11
35	51.23	4.57	85	145.41	97.43	13	52.06	85.49	4
36	118.39	31.01	94	54.07	99.03	22	160.95	52.77	16
37	95.30	83.15	40	26.30	40.91	29	38.69	80.22	10
38	97.59	80.70	50	11.07	16.30	23	127.89	57.25	4
39	91.70	64.00	43	60.32	63.88	32	104.93	73.31	9
40	104.92	36.81	46	52.42	49.03	10	184.96	51.90	13
41	88.40	31.28	67	91.23	98.94	20	52.66	77.09	16
42	170.53	80.18	75	136.66	6.53	13	13.19	56.89	17
43	86.69	7.04	97	139.13	78.32	16	147.01	46.57	10
44	165.37	68.36	46	56.70	28.84	37	154.44	34.27	6
45	101.87	38.07	59	75.99	24.14	13	181.56	6.82	17
46	17.28	63.39	74	36.23	66.25	35	186.39	37.79	15
47	132.01	92.69	90	157.71	24.61	16	2.79	7.96	19
48	41.32	85.39	90	11.37	66.59	29	46.87	98.28	11
49	169.46	49.77	78	139.40	51.73	12	123.36	18.16	20
50	136.27	42.78	60	155.74	42.41	28	189.80	81.19	15
51	35.67	82.26	83	155.48	55.47	28	190.04	87.50	19
52	13.97	73.82	94	51.88	28.71	16	111.33	68.84	10
53	1.94	44.71	40	74.76	70.66	30	183.12	56.95	15
54	178.41	96.89	76	117.52	41.49	21	128.31	16.10	9
55	26.69	36.32	85	54.56	36.05	32	78.00	46.69	5
56	155.84	39.75	81	74.17	82.87	10	97.20	34.52	19
57	185.03	73.87	42	39.41	92.50	14	120.86	22.50	6
58	143.04	44.91	73	91.97	4.60	23	109.91	59.25	17
59	98.16	93.73	52	8.92	23.26	19	185.24	31.23	14
60	93.90	25.72	55	159.96	34.85	12	183.75	91.63	17
61	176.54	38.18	94	15.39	81.50	11	78.98	90.96	11
62	97.68	69.71	73	103.77	98.55	22	192.65	25.71	12
63	82.91	89.71	63	61.36	96.90	25	34.79	11.09	16
64	34.82	12.11	52	115.51	90.49	32	25.27	19.30	20
65	95.06	24.15	45	191.89	29.66	11	27.02	49.96	15
66	156.73	22.83	78	129.11	99.20	33	101.13	72.86	10
67	111.29	48.95	41	7.07	24.94	22	4.30	20.82	4
68	31.97	89.15	50	86.08	10.59	15	189.59	24.80	20
69	28.77	35.81	46	102.00	95.10	16	165.42	85.17	5
70	129.89	38.52	46	107.24	23.34	11	3.00	41.58	11
71	107.84	91.67	78	136.28	68.98	20	35.24	61.67	18
72	65.14	11.38	78	55.52	5.84	16	66.41	23.37	8
73	29.40	63.11	51	25.77	73.07	39	26.20	10.20	4
74	31.97	13.28	90	78.54	88.17	16	161.90	51.59	5
75	25.88	37.42	61	191.28	27.24	16	68.95	47.71	20
76	115.78	32.44	90	37.43	37.91	16	188.02	15.27	6
77	18.45	68.01	90	180.80	37.43	21	116.40	62.18	8
78	182.11	79.55	91	108.76	74.88	28	175.77	54.40	7
79	16.59	50.39	42	91.38	23.78	31	168.95	65.41	20
80	160.08	29.62	52	176.41	17.19	40	181.08	14.45	5
81	175.43	88.60	69	91.72	44.93	28	91.98	75.15	15
82	18.69	35.19	49	144.83	30.45	28	109.27	22.20	15
83	85.26	73.84	91	79.81	83.92	29	159.72	51.94	16
84	94.64	55.54	48	180.81	23.77	12	57.14	78.53	8
85	116.04	20.15	45	138.01	50.24	22	98.05	2.23	17
86	143.25	54.85	92	139.92	94.26	39	119.82	32.44	4
87	5.41	51.96	93	65.54	63.40	19	3.11	87.29	12

88	146.28	34.88	83	151.36	86.73	29	118.70	84.47	11
89	153.39	2.46	41	127.21	94.02	18	86.74	53.84	11
90	1.95	14.88	83	48.00	75.08	15	161.47	86.66	5
91	61.66	13.19	55	32.11	69.96	30	63.05	94.98	11
92	46.57	70.79	52	159.28	96.80	31	178.58	82.64	6
93	100.69	70.91	87	191.83	99.44	21	115.57	85.41	8
94	190.74	61.17	52	91.63	45.18	11	36.80	9.87	9
95	111.56	67.19	61	118.20	7.09	21	157.59	65.13	11
96	19.49	44.75	96	171.54	29.28	25	122.41	70.35	16
97	124.34	69.61	58	91.44	15.24	40	10.78	61.02	17
98	172.60	37.83	65	190.37	41.75	22	84.04	79.96	4
99	31.53	10.00	53	115.15	13.13	25	135.81	3.46	11
100	198.49	96.03	70	164.15	60.41	22	183.72	77.02	19

Τα δρομολόγια στην καλύτερη λύση για $b1$=10, $b2$=40 είναι:

- 0, 9, 99, 64, 74, 4, 13, 0
- 0, 15, 16, 25, 7, 84, 0
- 0, 20, 1, 61, 26, 49, 0
- 0, 30, 11, 5, 83, 6, 0
- 0, 35, 91, 72, 18, 27, 89, 0
- 0, 36, 76, 40, 45, 85, 0
- 0, 43, 24, 12, 67, 19, 0
- 0, 44, 29, 42, 81, 54, 32, 0
- 0, 50, 37, 97, 28, 86, 2, 0
- 0, 53, 23, 87, 79, 96, 75, 0
- 0, 58, 31, 14, 56, 66, 0
- 0, 63, 3, 68, 51, 48, 0
- 0, 65, 60, 41, 8, 33, 17, 0
- 0, 69, 55, 34, 82, 90, 21, 0
- 0, 70, 38, 59, 71, 47, 98, 0
- 0, 73, 46, 77, 52, 92, 0
- 0, 80, 95, 93, 62, 39, 88, 0
- 0, 100, 22, 78, 57, 94, 10, 0

Τα δρομολόγια στην καλύτερη λύση για $b1$=4, $b2$=10 είναι:

- 0, 18, 0
- 0, 27, 25, 9, 68, 79, 70, 33, 99, 95, 5, 82, 47, 97, 32, 58, 74, 0
- 0, 28, 67, 59, 38, 29, 22, 7, 12, 37, 76, 11, 21, 52, 44, 55, 40, 39, 0
- 0, 51, 34, 100, 77, 65, 42, 24, 30, 54, 85, 49, 71, 43, 88, 8, 62, 10, 41, 0
- 0, 63, 36, 57, 1, 19, 61, 48, 23, 73, 91, 46, 90, 87, 53, 56, 0

- 0, 69, 92, 93, 16, 75, 2, 14, 13, 17, 80, 84, 98, 15, 96, 60, 50, 26, 0
- 0, 83, 35, 86, 66, 89, 64, 78, 3, 20, 6, 45, 72, 31, 81, 94, 4, 0

Τα δρομολόγια στην καλύτερη λύση για $b1$=20, $b2$=100 είναι:

- 0, 15, 28, 75, 55, 89, 18, 56, 82, 85, 99, 54, 49, 57, 86, 8, 95, 6, 43, 74, 36, 83, 46, 2, 59, 62, 68, 76, 80, 45, 26, 44, 33, 34, 67, 0
- 0, 64, 65, 71, 4, 32, 7, 5, 84, 3, 93, 88, 31, 38, 78, 40, 9, 22, 53, 79, 10, 69, 90, 14, 60, 51, 50, 100, 92, 27, 94, 0
- 0, 73, 63, 72, 21, 20, 23, 19, 13, 96, 77, 58, 25, 52, 39, 66, 12, 81, 98, 30, 61, 24, 91, 48, 11, 35, 41, 37, 1, 87, 29, 16, 97, 42, 70, 17, 47, 0

B2 Δεδομένα και λύσεις του PDP

Στα προβλήματα η μέγιστη διάρκεια δρομολογίου T είναι ίση με 480 χρονικές μονάδες.

Σημείο αφετηρίας και τερματισμού (tw = time window)

Αριθμός	X	Y	Ζήτηση	tw1	tw2
0	260	347	0	0	480
61	260	347	0	0	480

Πελάτες συλλογής

Πελάτης	X	Y	Ζήτηση	tw1 $PDPtw$	tw2 $PDPtw$	tw1 $PDPstw$	tw2 $PDPstw$
1	400	200	50	20	120	30	60
2	345	120	60	60	200	70	110
3	200	10	100	120	330	150	240
4	470	420	90	20	120	30	90
5	88	23	10	200	400	230	240
6	300	488	80	10	200	20	70
7	200	425	100	50	210	80	140
8	200	400	20	120	290	130	150
9	300	200	100	190	370	230	290
10	7	17	40	20	270	70	90
11	113	451	50	20	160	30	60
12	212	488	50	60	200	70	100
13	290	390	70	90	340	110	160
14	380	190	30	10	120	30	50
15	297	5	80	50	270	80	130
16	37	255	90	30	100	40	90
17	183	91	10	90	210	120	130
18	201	229	80	30	200	140	190
19	117	162	30	20	150	30	60
20	1	199	40	60	260	70	100
21	27	410	100	10	100	40	90
22	280	270	50	220	330	240	270
23	72	10	40	300	390	360	390
24	400	50	10	20	120	50	70
25	480	70	90	40	130	70	120

Πελάτες παράδοσης

Πελάτης	X	Y	Ζήτηση	tw1 $PDPtw$	tw2 $PDPtw$	tw1 $PDPstw$	tw2 $PDPstw$
26	328	90	-50	80	140	100	140
27	100	50	-60	210	320	250	290
28	20	335	-100	300	440	330	390
29	37	137	-90	150	290	160	220
30	5	400	-10	250	380	300	310
31	400	402	-80	370	470	400	450
32	220	300	-100	110	270	160	220
33	222	324	-20	220	390	300	320
34	317	301	-100	300	440	330	390
35	490	320	-40	80	220	180	210
36	320	214	-50	200	300	260	290
37	277	305	-50	160	250	160	200
38	201	200	-70	200	400	210	250
39	70	219	-30	120	350	170	190
40	100	420	-80	210	410	220	270
41	200	495	-90	210	390	230	280
42	340	410	-10	290	420	300	310
43	471	482	-80	320	470	330	380
44	470	34	-30	90	210	170	200
45	433	145	-40	180	360	210	240
46	400	476	-100	110	210	150	210
47	5	85	-50	290	410	320	350
48	150	300	-40	400	470	430	460
49	25	445	-10	180	259	210	230
50	140	310	-90	210	320	250	300

Για πρόβλημα n ζευγών πελατών συλλογής παράδοσης χρησιμοποιούνται οι πρώτοι n πελάτες από τους παραπάνω πίνακες, τροποποιώντας την αρίθμηση των πελατών παράδοσης έτσι ώστε ο πελάτης συλλογής με αρίθμηση i να αντιστοιχεί στον πελάτη παράδοσης με αρίθμηση $i+n$. Επίσης απαιτείται τροποποίηση του σημείου τερματισμού σε $2n+1$.

Τα δρομολόγια της αρχικής εφικτής λύσης για τα προβλήματα με 5 ζεύγη πελατών είναι:

- 0, 1, 2, 6, 3, 7, 8, 11
- 0, 4, 9, 5, 10, 11

Τα δρομολόγια στην καλύτερη λύση του $PDPstw$ για 5 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 6, 3, 7, 8, 11
- 0, 4, 9, 5, 10, 11

Για z_1=8

- 0, 1, 2, 6, 3, 5, 7, 10, 8, 11
- 0, 4, 9, 11

Τα δρομολόγια στην καλύτερη λύση του $PDPtw$ για 5 ζεύγη πελατών είναι:
Για z_1=4 και z_1=8

- 0, 1, 2, 6, 3, 5, 7, 8, 10, 11
- 0, 4, 9, 11

Τα δρομολόγια στην καλύτερη λύση του $PDPntw$ για 5 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 6, 3, 5, 7, 8,10, 11
- 0, 4, 9, 11

Για z_1=8

- 0, 4, 1, 2, 6, 7, 3, 5, 9, 8, 10, 11

Τα δρομολόγια της αρχικής εφικτής λύσης για τα προβλήματα με 10 ζεύγη πελατών είναι:

- 0, 1, 2, 11, 3, 12, 13, 21
- 0, 4, 14, 5, 15, 21
- 0, 6, 7, 8, 17, 9, 18, 19, 16, 21
- 0, 10, 20, 21

Τα δρομολόγια στην καλύτερη λύση του $PDPstw$ για 10 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 11, 3, 12, 13, 21
- 0, 4, 14, 5, 15, 21
- 0, 6, 7, 8, 17, 18, 16, 21
- 0, 10, 20, 9, 19, 21

Για z_1=8

- 0, 1, 2, 11, 3, 5, 12, 15, 13, 21
- 0, 6, 4, 14, 9, 19, 16, 21
- 0, 7, 8, 17, 18, 21
- 0, 10, 20, 21

Τα δρομολόγια στην καλύτερη λύση του $PDPtw$ για 10 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 11, 3, 5, 12, 13, 15, 21
- 0, 4, 6, 14, 16, 21
- 0, 7, 8, 18, 17, 21
- 0, 10, 20, 9, 19, 21

Για z_1=8

- 0, 1, 2, 11, 3, 5, 12, 13, 15, 21
- 0, 6, 16, 21
- 0, 7, 8, 18, 17, 21
- 0, 10, 4, 20, 9, 14, 19, 21

Τα δρομολόγια στην καλύτερη λύση του $PDPntw$ για 10 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 11, 3, 5, 12, 13, 15, 8, 18, 21
- 0, 6, 4, 16, 7, 14, 17, 21
- 0, 10, 9, 20, 19, 21

Για z_1=8

- 0, 1, 2, 11, 3, 5, 12, 13, 15, 7, 8, 18, 17, 21
- 0, 6, 4, 16, 14, 10, 9, 20, 19, 21

Τα δρομολόγια της αρχικής εφικτής λύσης για τα προβλήματα με 15 ζεύγη πελατών είναι:

- 0,1,2,16,3,17,18,31
- 0,4,19,5,20,31
- 0,6,7,8,22,9,23,24,21,31
- 0,10,25,31
- 0,11,12,13,27,28,26,31
- 0,14,15,29,30,31

Τα δρομολόγια στην καλύτερη λύση του $PDPstw$ για 15 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 16, 3, 17, 18, 31
- 0, 4, 19, 5, 20, 31
- 0, 6, 12, 13, 27, 28, 9, 24, 21, 31
- 0, 10, 25, 31
- 0, 11, 7, 8, 22, 26, 23, 31
- 0, 14, 15, 29, 30, 31

Για z_1=8

- 0, 1, 2, 16, 3, 5, 17, 20, 18, 31
- 0, 6, 21, 31
- 0, 7, 8, 22, 23, 31

- 0, 10, 15, 25, 30, 31
- 0, 11, 12, 13, 27, 28, 26, 9, 24, 31
- 0, 14, 4, 29, 19, 31

Τα δρομολόγια στην καλύτερη λύση του $PDPtw$ για 15 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 16, 3, 5, 17, 18, 20, 31
- 0, 7, 8, 23, 22, 31
- 0, 10, 25, 15, 30, 31
- 0, 11, 12, 13, 27, 28, 26, 9, 24, 31
- 0, 14, 4, 6, 19, 29, 21, 31

Για z_1=8

- 0, 4, 1, 14, 16, 15, 19, 29, 30, 31
- 0, 10, 25, 2, 3, 5, 17, 18, 20, 31
- 0, 11, 12, 6, 27, 26, 9, 24, 21, 31
- 0, 13, 7, 8, 23, 22, 28, 31

Τα δρομολόγια στην καλύτερη λύση του $PDPntw$ για 15 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 16, 3, 5, 17, 18, 20, 31
- 0, 6, 4, 21, 19, 31
- 0, 10, 9, 25, 24, 31
- 0, 11, 12, 13, 27, 26, 28, 31
- 0, 14, 15, 29, 30, 7, 8, 23, 22, 31

Για z_1=8

- 0, 4, 6, 19, 10, 21, 25, 31
- 0, 7, 8, 23, 22, 1, 2, 16, 3, 5, 17, 18, 20, 31
- 0, 11, 12, 13, 27, 28, 9, 26, 24, 31
- 0, 14, 15, 29, 30, 31

Τα δρομολόγια της αρχικής εφικτής λύσης για τα προβλήματα με 20 ζεύγη πελατών είναι:

- 0,1,2,21,3,22,23,41
- 0,4,24,5,25,41
- 0,6,7,8,27,9,28,29,26,41

- 0,10,30,41
- 0,11,12,13,32,33,31,41
- 0,14,15,34,35,41
- 0,16,17,18,36,37,38,41
- 0,19,20,39,40,41

Τα δρομολόγια στην καλύτερη λύση του $PDPstw$ για 20 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 21, 3, 22, 23, 41
- 0, 4, 24, 5, 25, 41
- 0, 6, 7, 8, 27, 28, 26, 41
- 0, 10, 17, 30, 37, 41
- 0, 11, 12, 13, 32, 33, 31, 41
- 0, 14, 15, 34, 35, 41
- 0, 16, 18, 36, 38, 41
- 0, 19, 20, 39, 40, 9, 29, 41

Για z_1=8

- 0, 1, 2, 21, 3, 5, 22, 25, 23, 41
- 0, 4, 13, 24, 33, 9, 29, 41
- 0, 6, 26, 41
- 0, 11, 20, 17, 18, 40, 31, 37, 38, v
- 0, 12, 7, 8, 32, 27, 28, 41
- 0, 14, 16, 34, 36, 41
- 0, 19, 10, 15, 39, 30, 35, 41

Τα δρομολόγια στην καλύτερη λύση του $PDPtw$ για 20 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 14, 21, 34, 41
- 0, 2, 3, 5, 22, 23, 25, 41
- 0, 4, 13, 33, 24, 41
- 0, 6, 7, 8, 28, 27, 26, 41
- 0, 10, 17, 30, 37, 41
- 0, 12, 11, 32, 31, 9, 29, 41
- 0, 16, 18, 36, 38, 41

- 0, 20, 19, 39, 40, 15, 35, 41

Για z_1=8

- 0, 1, 2, 21, 3, 22, 23, 41
- 0, 4, 14, 34, 15, 24, 5, 25, 35, 41
- 0, 6, 26, 41
- 0, 8, 7, 28, 27, 41
- 0, 11, 12, 13, 32, 31, 33, 41
- 0, 16, 17, 18, 36, 38, 37, 41
- 0, 19, 10, 20, 39, 40, 30, 9, 29, 41

Τα δρομολόγια στην καλύτερη λύση του $PDPntw$ για 20 ζεύγη πελατών είναι:

Για z_1=4

- 0, 1, 2, 21, 3, 22, 5, 23, 25, 41
- 0, 4, 13, 33, 24, 41
- 0, 7, 8, 28, 27, 19, 20, 39, 40, 41
- 0, 10, 9, 30, 29, 41
- 0, 12, 11, 31, 32, 41
- 0, 14, 15, 34, 35, 6, 26, 41
- 0, 16, 17, 18, 36, 38, 37, 41

Για z_1=8

- 0, 1, 2, 21, 3, 5, 22, 17, 18, 38, 37, 23, 25, 41
- 0, 7, 4, 27, 19, 24, 10, 39, 30, 41
- 0, 11, 12, 13, 31, 33, 32, 41
- 0, 14, 15, 34, 16, 35, 36, 8, 28, 41
- 0, 20, 40, 9, 29, 6, 26, 41

Τα δρομολόγια της αρχικής εφικτής λύσης για τα προβλήματα με 25 ζεύγη πελατών είναι:

- 0,1,2,26,3,27,28,51
- 0,4,29,5,30,51
- 0,6,7,8,32,9,33,34,31,51
- 0,10,35,51

- 0,11,12,13,37,38,36,51
- 0,14,15,39,40,51
- 0,16,17,18,41,42,43,51
- 0,19,20,44,45,51
- 0,21,46,22,47,23,48,51
- 0,24,25,49,50,51

Τα δρομολόγια στην καλύτερη λύση του $PDPstw$ για 25 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 26, 3, 27, 28, 51
- 0, 4, 29, 5, 30, 51
- 0, 6, 12, 13, 37, 38, 9, 34, 31, 51
- 0, 7, 8, 32, 33, 51
- 0, 10, 35, 22, 47, 23, 48, 51
- 0, 11, 21, 46, 36, 51
- 0, 14, 15, 39, 40, 51
- 0, 16, 18, 41, 43, 51
- 0, 19, 20, 17, 44, 45, 42, 51
- 0, 24, 25, 49, 50, 51

Για z_1=8

- 0, 1, 2, 26, 3, 5, 27, 30, 28, 51
- 0, 4, 25, 29, 50, 0
- 0, 6, 12, 13, 37, 38, 9, 34, 31, 51
- 0, 7, 8, 32, 33, 51
- 0, 11, 20, 17, 45, 36, 42, 51
- 0, 14, 24, 15, 39, 49, 40, 51
- 0, 16, 41, 51
- 0, 19, 10, 44, 35, 22, 47, 23, 48, 51
- 0, 21, 18, 46, 43, 51

Τα δρομολόγια στην καλύτερη λύση του $PDPtw$ για 25 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 26, 3, 27, 28, 51
- 0, 4, 13, 38, 29, 51

- 0, 6, 7, 8, 32, 33, 31, 51
- 0, 10, 35, 22, 23, 47, 48, 51
- 0, 12, 11, 36, 37, 51
- 0, 14, 24, 25, 49, 50, 39, 51
- 0, 15, 5, 30, 40, 51
- 0, 16, 17, 18, 41, 43, 42, 51
- 0, 20, 19, 44, 45, 9, 34, 51
- 0, 21, 46, 51

Για z_1=8

- 0, 1, 2, 26, 3, 27, 28, 51
- 0, 4, 14, 39, 29, 20, 22, 45, 47, 23, 48, 51
- 0, 11, 12, 13, 37, 38, 9, 36, 34, 51
- 0, 15, 5, 30, 40, 51
- 0, 16, 17, 18, 41, 43, 42, 51
- 0, 19, 10, 44, 35, 7, 8, 32, 33, 51
- 0, 21, 46, 6, 31, 51
- 0, 25, 24, 49, 50, 51

Τα δρομολόγια στην καλύτερη λύση του $PDPntw$ για 25 ζεύγη πελατών είναι:
Για z_1=4

- 0, 1, 2, 26, 3, 27, 28, 51
- 0, 4, 6, 31, 5, 29, 30, 51
- 0, 10, 35, 22, 47, 23, 48, 51
- 0, 11, 12, 13, 37, 38, 36, 51
- 0, 14, 15, 39, 40, 7, 8, 33, 32, 51
- 0, 16, 17, 18, 41, 42, 43, 51
- 0, 19, 20, 44, 45, 9, 34, 51
- 0, 21, 46, 51
- 0, 25, 24, 50, 49, 51

Για z_1=8

- 0, 1, 2, 26, 3, 5, 27, 28, 30, 51
- 0, 4, 29, 10, 35, 51
- 0, 7, 8, 33, 32, 19, 20, 17, 44, 45, 42, 51
- 0, 11, 12, 13, 37, 38, 9, 36, 34, 51
- 0, 14, 15, 39, 40, 21, 6, 46, 31, 51
- 0, 22, 23, 47, 18, 48, 43, 51
- 0, 24, 25, 16, 49, 41, 50, 51

Β3 Δεδομένα και λύσεις του IRP

Δεδομένα του IRP προβλήματος 1 για 10 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	96.89	201.85	0	0	3000	0
1	58.57	569.31	2	12.798	150	37
2	962.40	195.10	2	9.977	150	95
3	616.56	583.70	1	17.769	110	48
4	86.63	476.31	1	17.77	110	110
5	561.27	517.81	3	4.307	300	61
6	616.52	823.10	1	19.495	110	62
7	963.84	732.23	1	17.58	110	96
8	574.30	69.06	2	13.494	150	47
9	371.16	672.13	2	11.053	150	29
10	452.15	643.48	2	9.623	150	89

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	1039	1323	1244
1	35	81	94
2	37	85	37
3	29	41	60
4	99	37	28
5	63	42	30
6	97	13	30
7	28	80	48
8	100	63	72
9	36	57	70
10	17	43	99

Στη λύση του IRP προβλήματος 1 για 10 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 8(53), 5(2), 6(48), 9(7), 0(0)

Για την περίοδο 2

- 0(0), 4(26), 1(79), 9(57), 5(44), 3(22), 7(12), 2(64), 8(63), 0(0)

Για την περίοδο 3

- 0(0), 4(28), 1(94), 9(70), 10(70), 6(30), 7(48), 3(60), 5(28), 8(72), 0(0)

Δεδομένα του IRP προβλήματος 1 για 10 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	96.89	201.85	0	0	5000	0
1	58.57	569.31	2	12.798	250	15
2	962.40	195.10	2	9.977	250	102
3	616.56	583.70	1	17.769	110	11
4	86.63	476.31	1	17.77	110	25
5	561.27	517.81	3	4.307	500	88
6	616.52	823.10	1	19.495	110	25
7	963.84	732.23	1	17.58	110	26
8	574.30	69.06	2	13.494	250	32
9	371.16	672.13	2	11.053	250	69
10	452.15	643.48	2	9.623	250	107

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	1368	1119	1416	1081	1323
1	35	81	94	37	85
2	37	29	41	60	99
3	37	28	63	42	30
4	97	13	30	28	80
5	48	100	63	72	36
6	57	70	17	43	99
7	47	58	71	72	57
8	39	99	49	77	26
9	13	67	56	73	21
10	51	93	99	43	29

Στη λύση του IRP προβλήματος 1 για 10 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 4(72), 1(20), 5(50), 3(26), 6(32), 7(79), 2(5), 8(7), 0(0)

Για την περίοδο 2

- 0(0), 4(13), 1(81), 9(11), 10(37), 6(70), 3(28), 5(112), 8(148), 0(0)

Για την περίοδο 3

- 0(0), 4(30), 1(94), 9(56), 10(100), 6(17), 7(71), 3(63), 5(69), 0(0)

Για την περίοδο 4

- 0(0), 4(28), 1(37), 9(73), 10(68), 3(42), 6(43), 7(72), 2(60), 8(77), 0(0)

Για την περίοδο 5

- 0(0), 4(80), 1(85), 9(21), 10(3), 3(30), 6(99), 7(57), 2(99), 8(26), 0(0)

Δεδομένα του IRP προβλήματος 1 για 20 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	452.15	126.32	0	0	6000	0
1	201.85	508.83	2	7.916	150	53
2	569.31	431.79	2	9.236	150	25
3	195.10	915.94	1	12.594	110	27
4	583.70	709.02	1	14.248	110	40
5	476.31	890.66	3	2.16	300	97
6	517.81	588.89	1	14.634	110	74
7	823.10	636.83	1	14.138	110	45
8	732.23	342.21	2	9.407	150	58
9	69.06	823.79	2	7.002	150	86
10	672.13	305.15	2	8.806	150	70
11	643.48	307.54	1	14.217	110	51
12	828.01	201.32	1	11.578	110	103
13	204.47	265.16	3	3.586	300	53
14	617.49	650.92	2	9.683	150	68
15	617.70	154.61	1	13.637	110	52
16	301.07	388.70	2	9.348	150	89
17	871.74	647.61	2	8.204	150	19
18	589.65	848.30	3	3.331	300	29
19	981.77	63.84	2	8.052	150	90
20	442.23	414.44	1	11.584	110	97

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	2426	2195	2663
1	35	98	35
2	36	42	79
3	42	91	77
4	48	22	85
5	81	78	75
6	89	83	40
7	48	16	55
8	84	78	79
9	44	76	42
10	97	30	64
11	59	56	41
12	65	30	30
13	17	46	77
14	65	22	24
15	60	79	12
16	55	81	99
17	62	68	96
18	64	17	14
19	55	44	50
20	26	54	67

Στη λύση του IRP προβλήματος 1 για 20 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 2(16), 6(15), 3(15), 5(89), 18(66), 4(8), 7(19), 17(111), 8(26), 10(27), 11(8), 15(8), 0(0)

Για την περίοδο 2

- 0(0), 11(56), 10(30), 8(78), 12(22), 19(59), 15(79), 0(0)
- 0(0), 2(37), 6(83), 14(19), 4(22), 3(91), 9(34), 1(80), 16(47), 13(87), 0(0)

Για την περίοδο 3

- 0(0), 2(79), 7(55), 17(96), 8(79), 10(64), 11(41), 15(12), 0(0)
- 0(0), 16(99), 1(35), 9(42), 3(77), 5(48), 4(85), 14(24), 6(40), 20(50), 0(0)

Δεδομένα του IRP προβλήματος 1 για 20 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	452.15	126.32	0	0	10000	0
1	201.85	508.83	2	7.916	250	49
2	569.31	431.79	2	9.236	250	10
3	195.10	915.94	1	12.594	110	0
4	583.70	709.02	1	14.248	110	62
5	476.31	890.66	3	2.16	500	17
6	517.81	588.89	1	14.634	110	38
7	823.10	636.83	1	14.138	110	35
8	732.23	342.21	2	9.407	250	105
9	69.06	823.79	2	7.002	250	77

10	672.13	305.15	2	8.806	250	27
11	643.48	307.54	1	14.217	110	50
12	828.01	201.32	1	11.578	110	82
13	204.47	265.16	3	3.586	500	44
14	617.49	650.92	2	9.683	250	19
15	617.70	154.61	1	13.637	110	3
16	301.07	388.70	2	9.348	250	73
17	871.74	647.61	2	8.204	250	33
18	589.65	848.30	3	3.331	500	80
19	981.77	63.84	2	8.052	250	17
20	442.23	414.44	1	11.584	110	24

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	2131	2288	2977	2953	2878
1	35	98	35	36	42
2	79	42	91	77	48
3	22	85	81	78	75
4	89	83	40	48	16
5	55	84	78	79	44
6	76	42	97	30	64
7	59	56	41	65	30
8	30	17	46	77	65
9	22	24	60	79	12
10	55	81	99	62	68
11	96	64	17	14	55
12	44	50	26	54	67
13	63	35	37	50	84
14	55	68	96	80	61
15	63	78	62	99	29
16	39	100	52	77	33
17	69	95	33	66	28
18	53	77	100	20	37
19	34	27	71	72	17
20	28	85	42	49	57

Στη λύση του IRP προβλήματος 1 για 20 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :

Για την περίοδο 1

- 0(0), 13(19), 20(4), 2(69), 6(38), 3(22), 5(47), 4(27), 14(36), 7(24), 17(36), 19(44), 10(28), 11(46), 15(60), 0(0)

Για την περίοδο 2

- 0(0), 2(42), 18(50), 5(101), 3(85), 1(84), 16(66), 13(72), 0(0)
- 0(0), 11(64), 10(81), 15(78), 0(0)
- 0(0), 12(12), 17(95), 7(56), 14(68), 4(83), 6(42), 20(85), 0(0)

Για την περίοδο 3

- 0(0), 2(91), 6(97), 14(96), 7(41), 17(33), 19(71), 12(26), 0(0)
- 0(0), 16(52), 1(35), 9(29), 3(81), 5(101), 18(120), 4(40), 20(42), 0(0)
- 0(0), 11(17), 10(99), 15(62), 0(0)

Για την περίοδο 4

- 0(0), 13(50), 16(77), 1(36), 9(91), 3(78), 5(41), 4(48), 6(30), 20(49), 0(0)
- 0(0), 15(99), 12(54), 19(89), 0(0)
- 0(0), 2(77), 14(82), 7(95), 17(94), 8(65), 10(62), 11(14), 0(0)

Για την περίοδο 5

- 0(0), 2(48), 11(55), 10(68), 8(65), 12(67), 15(29), 0(0)
- 0(0), 13(84), 16(33), 1(42), 3(75), 5(33), 18(37), 4(16), 14(59), 6(64), 20(57), 0(0)

Δεδομένα του IRP προβλήματος 1 για 30 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	672.13	414.44	0	0	9000	0
1	643.48	332.24	2	8.201	150	17
2	828.01	721.32	2	8.688	150	96
3	204.47	473.85	1	11.295	110	59
4	617.49	798.64	1	11.368	110	82
5	617.70	90.87	3	4.443	300	79
6	301.07	874.56	1	10.604	110	101
7	871.74	776.98	1	11.868	110	20
8	589.65	771.57	2	7.435	150	20
9	981.77	63.10	2	6.407	150	7
10	442.23	594.44	2	8.99	150	78
11	126.32	792.49	1	11.098	110	72
12	508.83	274.13	1	9.582	110	47
13	431.79	371.02	3	5.219	300	33
14	915.94	853.11	2	6.562	150	101
15	709.02	678.73	1	10.094	110	62
16	890.66	754.38	2	8.16	150	76
17	588.89	417.26	2	8.199	150	40
18	636.83	320.90	3	2.179	300	21
19	342.21	372.30	2	7.221	150	19
20	823.79	275.35	1	12.682	110	42
21	305.15	679.56	1	12.717	110	16
22	307.54	581.63	1	12.652	110	21
23	201.32	970.20	2	8.622	150	60
24	265.16	415.51	1	10.393	110	79
25	650.92	330.49	2	6.513	150	41
26	154.61	966.71	1	12.155	110	109
27	388.70	825.90	2	8.247	150	86
28	647.61	305.40	2	6.48	150	61
29	848.30	823.28	2	6.838	150	20
30	63.84	556.91	1	10.781	110	63

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	3862	3392	3824
1	44	50	26
2	54	67	63
3	35	37	50
4	84	55	68
5	96	80	61
6	63	78	62
7	99	29	39
8	100	52	77
9	33	69	95
10	33	66	28
11	53	77	100
12	20	37	34
13	27	71	72
14	17	28	85
15	42	49	57
16	59	20	10
17	72	27	48
18	45	87	37
19	60	92	54

20	29	13	83
21	43	90	27
22	34	13	42
23	91	67	23
24	91	53	78
25	95	65	60
26	63	85	88
27	76	80	13
28	30	14	44
29	75	49	36
30	84	41	78

Στη λύση του IRP προβλήματος 1 για 30 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 7(79), 29(55), 4(2), 8(80), 23(31), 21(27), 22(13), 30(21), 24(12), 19(41), 13(16), 17(32), 0(0)
- 0(0), 9(95), 5(97), 12(10), 18(24), 1(27), 25(54), 0(0)

Για την περίοδο 2

- 0(0), 10(21), 6(40), 23(67), 26(39), 11(58), 30(41), 3(13), 24(53), 19(92), 13(49), 17(27), 0(0)
- 0(0), 1(50), 18(87), 25(65), 0(0)
- 0(0), 15(29), 2(25), 16(3), 7(29), 29(49), 4(55), 8(52), 27(70), 21(90), 22(13), 0(0)

Για την περίοδο 3

- 0(0), 10(28), 22(42), 21(27), 8(77), 4(68), 29(36), 14(29), 7(39), 16(10), 2(63), 15(57), 0(0)
- 0(0), 17(48), 13(72), 19(54), 12(34), 5(61), 9(95), 20(83), 0(0)
- 0(0), 24(78), 3(50), 30(78), 11(100), 26(88), 23(23), 6(62), 27(13), 0(0)
- 0(0), 1(26), 18(37), 28(27), 25(60), 0(0)

Δεδομένα του IRP προβλήματος 1 για 30 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	672.13	414.44	0	0	15000	0
1	643.48	332.24	2	8.201	250	65
2	828.01	721.32	2	8.688	250	105
3	204.47	473.85	1	11.295	110	77
4	617.49	798.64	1	11.368	110	85
5	617.70	90.87	3	4.443	500	36
6	301.07	874.56	1	10.604	110	93
7	871.74	776.98	1	11.868	110	42
8	589.65	771.57	2	7.435	250	25
9	981.77	63.10	2	6.407	250	91
10	442.23	594.44	2	8.99	250	53
11	126.32	792.49	1	11.098	110	90
12	508.83	274.13	1	9.582	110	96
13	431.79	371.02	3	5.219	500	28
14	915.94	853.11	2	6.562	250	14
15	709.02	678.73	1	10.094	110	42
16	890.66	754.38	2	8.16	250	102
17	588.89	417.26	2	8.199	250	14
18	636.83	320.90	3	2.179	500	91
19	342.21	372.30	2	7.221	250	103
20	823.79	275.35	1	12.682	110	48
21	305.15	679.56	1	12.717	110	102

22	307.54	581.63	1	12.652	110	6
23	201.32	970.20	2	8.622	250	67
24	265.16	415.51	1	10.393	110	55
25	650.92	330.49	2	6.513	250	84
26	154.61	966.71	1	12.155	110	31
27	388.70	825.90	2	8.247	250	77
28	647.61	305.40	2	6.48	250	26
29	848.30	823.28	2	6.838	250	14
30	63.84	556.91	1	10.781	110	18

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	4153	3999	3279	4465	4302
1	44	50	26	54	67
2	63	35	37	50	84
3	55	68	96	80	61
4	63	78	62	99	29
5	39	100	52	77	33
6	69	95	33	66	28
7	53	77	100	20	37
8	34	27	71	72	17
9	28	85	42	49	57
10	59	20	10	72	27
11	48	45	87	37	60
12	92	54	29	13	83
13	43	90	27	34	13
14	42	91	67	23	91
15	53	78	95	65	60
16	63	85	88	76	80
17	13	30	14	44	75
18	49	36	84	41	78
19	27	69	92	89	30
20	30	17	88	82	57
21	43	72	86	50	31
22	29	52	26	31	70
23	89	51	96	71	30
24	73	18	66	94	54
25	79	50	13	61	70
26	65	84	84	54	90
27	12	19	99	33	64
28	29	61	100	18	40
29	94	52	57	41	22
30	61	48	33	27	24

Στη λύση του IRP προβλήματος 1 για 30 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 15(11), 7(11), 14(28), 29(80), 8(9), 23(22), 26(34), 30(43), 24(18), 22(23), 10(6), 13(15), 5(3), 28(3), 0(0)

Για την περίοδο 2

- 0(0), 1(29), 25(45), 28(61), 12(50), 5(104), 9(22), 0(0)
- 0(0), 10(20), 22(52), 21(13), 6(71), 23(51), 26(84), 11(3), 30(48), 3(46), 24(18), 13(90), 0(0)
- 0(0), 15(78), 16(46), 7(77), 14(91), 29(52), 4(56), 8(27), 17(29), 0(0)

Για την περίοδο 3

- 0(0), 4(62), 8(71), 27(53), 6(33), 23(96), 26(84), 11(87), 10(10), 0(0)
- 0(0), 2(30), 16(88), 7(100), 14(90), 29(57), 15(95), 0(0)

- 0(0), 12(29), 13(27), 19(85), 24(66), 3(96), 30(33), 21(86), 22(26), 17(14), 0(0)
- 0(0), 20(87), 9(148), 5(48), 28(100), 18(78), 1(26), 25(13), 0(0)

Για την περίοδο 4
- 0(0), 1(54), 18(95), 28(18), 25(61), 0(0)
- 0(0), 10(72), 21(50), 22(31), 3(13), 24(40), 13(34), 17(44), 0(0)
- 0(0), 20(82), 5(110), 12(13), 19(89), 24(40), 3(13), 30(27), 11(37), 26(54), 23(71), 6(66), 27(33), 0(0)
- 0(0), 8(72), 4(99), 29(41), 7(20), 16(76), 2(50), 15(65), 0(0)

Για την περίοδο 5
- 0(0), 20(57), 0(0)
- 0(0), 2(84), 16(80), 7(37), 14(91), 29(22), 15(60), 0(0)
- 0(0), 17(75), 10(27), 21(31), 22(70), 13(13), 12(83), 28(40), 18(24), 1(67), 25(70), 0(0)
- 0(0), 4(29), 8(17), 27(64), 6(28), 23(30), 26(90), 11(60), 30(24), 3(61), 24(54), 19(30), 0(0)

Δεδομένα του IRP προβλήματος 2 για 10 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	96.89	201.85	0	0	3000	0
1	58.57	569.31	2	12.798	150	37
2	962.40	195.10	2	9.977	150	95
3	616.56	583.70	1	17.769	110	48
4	86.63	476.31	2	11.847	150	110
5	561.27	517.81	3	4.307	300	61
6	616.52	823.10	1	19.495	110	62
7	963.84	732.23	1	17.58	110	96
8	574.30	69.06	2	13.494	150	47
9	371.16	672.13	2	11.053	150	29
10	452.15	643.48	2	9.623	150	89

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	1039	1323	1244
1	35	81	94
2	37	85	37
3	29	41	60
4	99	37	28
5	63	42	30
6	97	13	30
7	28	80	48
8	100	63	72
9	36	57	70
10	17	43	99

Στη λύση του IRP προβλήματος 2 για 10 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 8(53), 5(2), 6(48), 9(7), 0(0)

Για την περίοδο 2

- 0(0), 4(26), 1(79), 9(57), 5(44), 3(22), 7(12), 2(64), 8(63), 0(0)

Για την περίοδο 3

- 0(0), 4(28), 1(94), 9(70), 10(70), 6(30), 7(48), 3(60), 5(28), 8(72), 0(0)

Δεδομένα του IRP προβλήματος 2 για 10 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	96.89	201.85	0	0	5000	0
1	58.57	569.31	2	12.798	250	15
2	962.40	195.10	2	9.977	250	102
3	616.56	583.70	1	17.769	110	11
4	86.63	476.31	2	11.847	250	25
5	561.27	517.81	3	4.307	500	88
6	616.52	823.10	1	19.495	110	25
7	963.84	732.23	1	17.58	110	26
8	574.30	69.06	2	13.494	250	32
9	371.16	672.13	2	11.053	250	69
10	452.15	643.48	2	9.623	250	107

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	1368	1119	1416	1081	1323

1	35	81	94	37	85
2	37	29	41	60	99
3	37	28	63	42	30
4	97	13	30	28	80
5	48	100	63	72	36
6	57	70	17	43	99
7	47	58	71	72	57
8	39	99	49	77	26
9	13	67	56	73	21
10	51	93	99	43	29

Στη λύση του IRP προβλήματος 2 για 10 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :

Για την περίοδο 1

- 0(0), 4(72), 1(20), 5(50), 3(26), 6(32), 7(79), 2(5), 8(7), 0(0)

Για την περίοδο 2

- 0(0), 4(13), 1(81), 9(11), 10(37), 6(70), 3(28), 5(112), 8(148), 0(0)

Για την περίοδο 3

- 0(0), 4(30), 1(94), 9(56), 10(100), 6(17), 7(71), 3(63), 5(69), 0(0)

Για την περίοδο 4

- 0(0), 4(28), 1(37), 9(73), 10(68), 3(42), 6(43), 7(72), 2(60), 8(77), 0(0)

Για την περίοδο 5

- 0(0), 4(80), 1(85), 9(21), 10(3), 3(30), 6(99), 7(57), 2(99), 8(26), 0(0)

Δεδομένα του IRP προβλήματος 2 για 20 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	452.15	126.32	0	0	6000	0
1	201.85	508.83	2	7.916	150	53
2	569.31	431.79	2	9.236	150	25
3	195.10	915.94	1	12.594	110	27
4	583.70	709.02	2	9.499	150	40
5	476.31	890.66	3	2.16	300	97
6	517.81	588.89	1	14.634	110	74
7	823.10	636.83	1	14.138	110	45
8	732.23	342.21	2	9.407	150	58
9	69.06	823.79	2	7.002	150	86
10	672.13	305.15	2	8.806	150	70
11	643.48	307.54	1	14.217	110	51
12	828.01	201.32	1	11.578	110	103
13	204.47	265.16	3	3.586	300	53
14	617.49	650.92	2	9.683	150	68
15	617.70	154.61	1	13.637	110	52
16	301.07	388.70	2	9.348	150	89
17	871.74	647.61	2	8.204	150	19
18	589.65	848.30	3	3.331	300	29
19	981.77	63.84	2	8.052	150	90
20	442.23	414.44	2	7.723	150	97

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	2426	2195	2663
1	35	98	35
2	36	42	79

3	42	91	77
4	48	22	85
5	81	78	75
6	89	83	40
7	48	16	55
8	84	78	79
9	44	76	42
10	97	30	64
11	59	56	41
12	65	30	30
13	17	46	77
14	65	22	24
15	60	79	12
16	55	81	99
17	62	68	96
18	64	17	14
19	55	44	50
20	26	54	67

Στη λύση του IRP προβλήματος 2 για 20 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 2(16), 6(15), 3(15), 5(89), 18(66), 4(8), 7(19), 17(111), 8(26), 10(27), 11(8), 15(8), 0(0)

Για την περίοδο 2

- 0(0), 11(56), 10(30), 8(78), 12(22), 19(59), 15(79), 0(0)
- 0(0), 2(37), 6(83), 14(19), 4(22), 3(91), 9(34), 1(80), 16(47), 13(87), 0(0)

Για την περίοδο 3

- 0(0), 2(79), 7(55), 17(96), 8(79), 10(64), 11(41), 15(12), 0(0)
- 0(0), 16(99), 1(35), 9(42), 3(77), 5(48), 4(85), 14(24), 6(40), 20(50), 0(0)

Δεδομένα του IRP προβλήματος 2 για 20 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	452.15	126.32	0	0	10000	0
1	201.85	508.83	2	7.916	250	49
2	569.31	431.79	2	9.236	250	10
3	195.10	915.94	1	12.594	110	0
4	583.70	709.02	2	9.499	250	62
5	476.31	890.66	3	2.16	500	17
6	517.81	588.89	1	14.634	110	38
7	823.10	636.83	1	14.138	110	35
8	732.23	342.21	2	9.407	250	105
9	69.06	823.79	2	7.002	250	77
10	672.13	305.15	2	8.806	250	27
11	643.48	307.54	1	14.217	110	50
12	828.01	201.32	1	11.578	110	82
13	204.47	265.16	3	3.586	500	44
14	617.49	650.92	2	9.683	250	19
15	617.70	154.61	1	13.637	110	3
16	301.07	388.70	2	9.348	250	73
17	871.74	647.61	2	8.204	250	33
18	589.65	848.30	3	3.331	500	80
19	981.77	63.84	2	8.052	250	17
20	442.23	414.44	2	7.723	250	24

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	2131	2288	2977	2953	2878
1	35	98	35	36	42
2	79	42	91	77	48
3	22	85	81	78	75
4	89	83	40	48	16
5	55	84	78	79	44
6	76	42	97	30	64
7	59	56	41	65	30
8	30	17	46	77	65
9	22	24	60	79	12
10	55	81	99	62	68
11	96	64	17	14	55
12	44	50	26	54	67
13	63	35	37	50	84
14	55	68	96	80	61
15	63	78	62	99	29
16	39	100	52	77	33
17	69	95	33	66	28
18	53	77	100	20	37
19	34	27	71	72	17
20	28	85	42	49	57

Στη λύση του IRP προβλήματος 2 για 20 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :

Για την περίοδο 1

- 0(0), 13(91), 0(0)
- 0(0), 15(60), 11(46), 10(28), 12(12), 19(44), 17(36), 7(24), 14(36), 4(27), 5(54), 3(22), 6(38), 2(69), 20(4), 0(0)

Για την περίοδο 2

- 0(0), 6(42), 4(83), 18(50), 5(90), 3(85), 1(84), 16(66), 0(0)
- 0(0), 11(64), 10(81), 17(95), 7(56), 14(68), 2(42), 20(86), 0(0)
- 0(0), 15(78), 0(0)

Για την περίοδο 3

- 0(0), 2(91), 6(97), 4(40), 14(96), 7(41), 17(33), 19(71), 12(26), 0(0)
- 0(0), 16(52), 1(35), 9(29), 3(81), 5(142), 18(120), 20(41), 0(0)
- 0(0), 11(17), 10(99), 15(62), 0(0)

Για την περίοδο 4

- 0(0), 13(91), 16(77), 1(36), 9(91), 3(78), 4(48), 6(30), 20(49), 0(0)
- 0(0), 15(99), 12(54), 19(89), 0(0)
- 0(0), 2(90), 14(80), 7(95), 17(94), 8(65), 10(62), 11(14), 0(0)

Για την περίοδο 5

- 0(0), 11(55), 10(68), 8(65), 12(67), 15(29), 0(0)
- 0(0), 13(43), 16(33), 1(42), 3(75), 5(37), 18(37), 4(16), 14(61), 6(64), 2(35), 20(57), 0(0)

Δεδομένα του IRP προβλήματος 2 για 30 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	672.13	414.44	0	0	9000	0
1	643.48	332.24	2	8.201	150	17
2	828.01	721.32	2	8.688	150	96
3	204.47	473.85	1	11.295	110	59
4	617.49	798.64	2	7.578	150	82
5	617.70	90.87	3	4.443	300	79
6	301.07	874.56	1	10.604	110	101
7	871.74	776.98	1	11.868	110	20
8	589.65	771.57	2	7.435	150	20
9	981.77	63.10	2	6.407	150	7
10	442.23	594.44	2	8.99	150	78
11	126.32	792.49	1	11.098	110	72
12	508.83	274.13	1	9.582	110	47
13	431.79	371.02	3	5.219	300	33
14	915.94	853.11	2	6.562	150	101
15	709.02	678.73	1	10.094	110	62
16	890.66	754.38	2	8.16	150	76
17	588.89	417.26	2	8.199	150	40
18	636.83	320.90	3	2.179	300	21
19	342.21	372.30	2	7.221	150	19
20	823.79	275.35	2	8.455	150	42
21	305.15	679.56	2	8.478	150	16
22	307.54	581.63	2	8.435	150	21
23	201.32	970.20	2	8.622	150	60
24	265.16	415.51	1	10.393	110	79
25	650.92	330.49	2	6.513	150	41
26	154.61	966.71	2	8.104	150	109
27	388.70	825.90	2	8.247	150	86
28	647.61	305.40	3	2.178	300	61
29	848.30	823.28	2	6.838	150	20
30	63.84	556.91	1	10.781	110	63

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	3862	3392	3824
1	44	50	26
2	54	67	63
3	35	37	50
4	84	55	68
5	96	80	61
6	63	78	62
7	99	29	39
8	100	52	77
9	33	69	95
10	33	66	28
11	53	77	100
12	20	37	34
13	27	71	72
14	17	28	85
15	42	49	57
16	59	20	10
17	72	27	48
18	45	87	37
19	60	92	54
20	29	13	83
21	43	90	27
22	34	13	42
23	91	67	23
24	91	53	78
25	95	65	60
26	63	85	88
27	76	80	13

28	30	14	44
29	75	49	36
30	84	41	78

Στη λύση του IRP προβλήματος 2 για 30 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 9(95), 5(97), 25(54), 0(0)
- 0(0), 7(79), 29(55), 4(2), 8(80), 23(31), 21(27), 22(26), 30(21), 24(12), 19(41), 12(10), 18(24), 1(27), 17(32), 0(0)

Για την περίοδο 2

- 0(0), 17(27), 13(65), 19(92), 24(53), 3(13), 30(41), 11(58), 26(39), 23(67), 6(40), 27(5), 8(52), 4(55), 29(49), 7(29), 16(3), 2(25), 15(29), 0(0)
- 0(0), 27(5), 21(90), 10(21), 0(0)
- 0(0), 1(50), 18(87), 25(65), 0(0)

Για την περίοδο 3

- 0(0), 10(28), 22(42), 21(27), 8(77), 4(68), 29(36), 14(29), 7(39), 16(10), 2(63), 15(57), 0(0)
- 0(0), 17(48), 13(72), 19(54), 12(34), 5(61), 9(95), 20(83), 0(0)
- 0(0), 24(78), 3(50), 30(78), 11(100), 26(88), 23(23), 6(62), 27(13), 0(0)
- 0(0), 1(26), 18(37), 28(27), 25(60), 0(0)

Δεδομένα του IRP προβλήματος 2 για 30 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	672.13	414.44	0	0	15000	0
1	643.48	332.24	2	8.201	250	65
2	828.01	721.32	2	8.688	250	105
3	204.47	473.85	1	11.295	110	77
4	617.49	798.64	2	7.578	250	85
5	617.70	90.87	3	4.443	500	36
6	301.07	874.56	1	10.604	110	93
7	871.74	776.98	1	11.868	110	42
8	589.65	771.57	2	7.435	250	25
9	981.77	63.10	2	6.407	250	91
10	442.23	594.44	2	8.99	250	53
11	126.32	792.49	1	11.098	110	90
12	508.83	274.13	1	9.582	110	96
13	431.79	371.02	3	5.219	500	28
14	915.94	853.11	2	6.562	250	14
15	709.02	678.73	1	10.094	110	42
16	890.66	754.38	2	8.16	250	102
17	588.89	417.26	2	8.199	250	14
18	636.83	320.90	3	2.179	500	91
19	342.21	372.30	2	7.221	250	103
20	823.79	275.35	2	8.455	250	48
21	305.15	679.56	2	8.478	250	102
22	307.54	581.63	2	8.435	250	6
23	201.32	970.20	2	8.622	250	67
24	265.16	415.51	1	10.393	110	55
25	650.92	330.49	2	6.513	250	84
26	154.61	966.71	2	8.104	250	31
27	388.70	825.90	2	8.247	250	77
28	647.61	305.40	3	2.178	500	26

29	848.30	823.28	2	6.838	250	14
30	63.84	556.91	1	10.781	110	18

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	4153	3999	3279	4465	4302
1	44	50	26	54	67
2	63	35	37	50	84
3	55	68	96	80	61
4	63	78	62	99	29
5	39	100	52	77	33
6	69	95	33	66	28
7	53	77	100	20	37
8	34	27	71	72	17
9	28	85	42	49	57
10	59	20	10	72	27
11	48	45	87	37	60
12	92	54	29	13	83
13	43	90	27	34	13
14	42	91	67	23	91
15	53	78	95	65	60
16	63	85	88	76	80
17	13	30	14	44	75
18	49	36	84	41	78
19	27	69	92	89	30
20	30	17	88	82	57
21	43	72	86	50	31
22	29	52	26	31	70
23	89	51	96	71	30
24	73	18	66	94	54
25	79	50	13	61	70
26	65	84	84	54	90
27	12	19	99	33	64
28	29	61	100	18	40
29	94	52	57	41	22
30	61	48	33	27	24

Στη λύση του IRP προβλήματος 2 για 30 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :

Για την περίοδο 1

- 0(0), 15(11), 7(11), 14(28), 29(80), 8(9), 10(6), 22(23), 21(0), 23(22), 26(34), 30(43), 24(18), 13(15), 5(3), 28(3), 0(0)

Για την περίοδο 2

- 0(0), 1(29), 25(45), 28(61), 9(64), 5(152), 12(50), 17(29), 0(0)
- 0(0), 10(20), 22(52), 21(13), 6(71), 23(51), 26(84), 11(3), 30(48), 3(46), 24(18), 13(90), 0(0)
- 0(0), 15(78), 8(27), 4(56), 29(52), 14(91), 7(77), 16(46), 0(0)

Για την περίοδο 3

- 0(0), 4(62), 8(71), 27(53), 6(33), 23(96), 26(84), 11(87), 10(10), 0(0)
- 0(0), 2(30), 16(88), 7(100), 14(90), 29(57), 15(95), 0(0)
- 0(0), 12(29), 13(27), 19(85), 24(66), 3(96), 30(33), 21(86), 22(26), 17(14), 0(0)
- 0(0), 20(87), 28(116), 18(78), 1(26), 25(13), 0(0)

Για την περίοδο 4

- 0(0), 20(139), 9(106), 5(110), 28(42), 18(42), 25(61), 0(0)

- 0(0), 1(54), 12(13), 19(89), 22(31), 21(50), 10(72), 0(0)
- 0(0), 15(65), 2(50), 16(76), 7(20), 29(41), 4(99), 8(72), 17(44), 0(0)
- 0(0), 13(34), 24(94), 3(80), 30(27), 11(37), 26(54), 23(71), 6(66), 27(33), 0(0)

Για την περίοδο 5

- 0(0), 2(84), 16(80), 7(37), 14(91), 29(22), 15(60), 0(0)
- 0(0), 17(75), 10(27), 21(31), 22(70), 12(83), 18(77), 1(67), 25(70), 0(0)
- 0(0), 4(29), 8(17), 27(64), 6(28), 23(30), 26(90), 11(60), 30(24), 3(61), 24(54), 19(30), 13(13), 0(0)

Δεδομένα του IRP προβλήματος 3 για 10 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	96.89	201.85	0	0	3000	0
1	58.57	569.31	3	7.884	300	37
2	962.40	195.10	3	3.651	300	95
3	616.56	583.70	1	17.769	110	48
4	86.63	476.31	2	11.847	150	110
5	561.27	517.81	3	4.307	300	61
6	616.52	823.10	1	19.495	110	62
7	963.84	732.23	2	11.72	150	96
8	574.30	69.06	3	8.927	300	47
9	371.16	672.13	3	5.265	300	29
10	452.15	643.48	3	3.12	300	89

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	1039	1323	1244
1	35	81	94
2	37	85	37
3	29	41	60
4	99	37	28
5	63	42	30
6	97	13	30
7	28	80	48
8	100	63	72
9	36	57	70
10	17	43	99

Στη λύση του IRP προβλήματος 3 για 10 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 4(26), 1(79), 9(64), 5(46), 3(22), 6(48), 7(12), 2(64), 8(116), 0(0)

Για την περίοδο 3

- 0(0), 4(28), 1(94), 9(70), 10(70), 6(30), 7(48), 3(60), 5(28), 8(72), 0(0)

Δεδομένα του IRP προβλήματος 3 για 10 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	96.89	201.85	0	0	5000	0
1	58.57	569.31	3	7.884	500	15
2	962.40	195.10	3	3.651	500	102
3	616.56	583.70	1	17.769	110	11
4	86.63	476.31	2	11.847	250	25
5	561.27	517.81	3	4.307	500	88
6	616.52	823.10	1	19.495	110	25
7	963.84	732.23	2	11.72	250	26
8	574.30	69.06	3	8.927	500	32
9	371.16	672.13	3	5.265	500	69
10	452.15	643.48	3	3.12	500	107

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	1368	1119	1416	1081	1323
1	35	81	94	37	85
2	37	29	41	60	99
3	37	28	63	42	30

4	97	13	30	28	80
5	48	100	63	72	36
6	57	70	17	43	99
7	47	58	71	72	57
8	39	99	49	77	26
9	13	67	56	73	21
10	51	93	99	43	29

Στη λύση του IRP προβλήματος 3 για 10 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 4(85), 1(101), 9(11), 10(80), 6(32), 7(21), 3(26), 8(7), 0(0)

Για την περίοδο 2

- 0(0), 5(103), 3(28), 6(87), 7(129), 2(5), 8(148), 0(0)

Για την περίοδο 3

- 0(0), 4(58), 1(131), 9(129), 10(99), 3(63), 5(20), 0(0)

Για την περίοδο 4

- 0(0), 5(72), 3(42), 6(43), 7(72), 2(159), 8(103), 0(0)

Για την περίοδο 5

- 0(0), 4(80), 1(85), 9(21), 10(29), 6(99), 7(57), 3(30), 5(36), 0(0)

Δεδομένα του IRP προβλήματος 3 για 20 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	452.15	126.32	0	0	6000	0
1	201.85	508.83	3	3.388	300	53
2	569.31	431.79	3	5.369	300	25
3	195.10	915.94	1	12.594	110	27
4	583.70	709.02	2	9.499	150	40
5	476.31	890.66	3	2.16	300	97
6	517.81	588.89	1	14.634	110	74
7	823.10	636.83	2	9.425	150	45
8	732.23	342.21	3	5.625	300	58
9	69.06	823.79	3	2.018	300	86
10	672.13	305.15	3	4.723	300	70
11	643.48	307.54	1	14.217	110	51
12	828.01	201.32	1	11.578	110	103
13	204.47	265.16	3	3.586	300	53
14	617.49	650.92	3	6.04	300	68
15	617.70	154.61	1	13.637	110	52
16	301.07	388.70	3	5.538	300	89
17	871.74	647.61	3	3.821	300	19
18	589.65	848.30	3	3.331	300	29
19	981.77	63.84	3	3.592	300	90
20	442.23	414.44	2	7.723	150	97

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	2426	2195	2663
1	35	98	35
2	36	42	79
3	42	91	77
4	48	22	85
5	81	78	75
6	89	83	40

7	48	16	55
8	84	78	79
9	44	76	42
10	97	30	64
11	59	56	41
12	65	30	30
13	17	46	77
14	65	22	24
15	60	79	12
16	55	81	99
17	62	68	96
18	64	17	14
19	55	44	50
20	26	54	67

Στη λύση του IRP προβλήματος 3 για 20 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 2(17), 6(15), 3(15), 5(62), 18(66), 4(30), 14(19), 7(19), 17(111), 8(26), 10(27), 11(8), 15(8), 0(0)

Για την περίοδο 2

- 0(0), 11(56), 10(30), 8(78), 12(22), 19(59), 15(79), 0(0)
- 0(0), 2(36), 6(83), 3(91), 9(76), 1(80), 16(47), 13(87), 0(0)

Για την περίοδο 3

- 0(0), 16(99), 1(35), 3(77), 5(75), 4(85), 14(24), 6(40), 20(50), 0(0)
- 0(0), 2(79), 7(55), 17(96), 8(79), 10(64), 11(41), 15(12), 0(0)

Δεδομένα του IRP προβλήματος 3 για 20 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	452.15	126.32	0	0	10000	0
1	201.85	508.83	3	3.388	500	49
2	569.31	431.79	3	5.369	500	10
3	195.10	915.94	1	12.594	110	0
4	583.70	709.02	2	9.499	250	62
5	476.31	890.66	3	2.16	500	17
6	517.81	588.89	1	14.634	110	38
7	823.10	636.83	2	9.425	250	35
8	732.23	342.21	3	5.625	500	105
9	69.06	823.79	3	2.018	500	77
10	672.13	305.15	3	4.723	500	27
11	643.48	307.54	1	14.217	110	50
12	828.01	201.32	1	11.578	110	82
13	204.47	265.16	3	3.586	500	44
14	617.49	650.92	3	6.04	500	19
15	617.70	154.61	1	13.637	110	3
16	301.07	388.70	3	5.538	500	73
17	871.74	647.61	3	3.821	500	33
18	589.65	848.30	3	3.331	500	80
19	981.77	63.84	3	3.592	500	17
20	442.23	414.44	2	7.723	250	24

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	2131	2288	2977	2953	2878
1	35	98	35	36	42

2	79	42	91	77	48
3	22	85	81	78	75
4	89	83	40	48	16
5	55	84	78	79	44
6	76	42	97	30	64
7	59	56	41	65	30
8	30	17	46	77	65
9	22	24	60	79	12
10	55	81	99	62	68
11	96	64	17	14	55
12	44	50	26	54	67
13	63	35	37	50	84
14	55	68	96	80	61
15	63	78	62	99	29
16	39	100	52	77	33
17	69	95	33	66	28
18	53	77	100	20	37
19	34	27	71	72	17
20	28	85	42	49	57

Στη λύση του IRP προβλήματος 3 για 20 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 11(46), 10(28), 12(12), 19(44), 17(36), 7(24), 14(36), 4(27), 5(38), 3(22), 6(38), 2(69), 20(4), 13(54), 0(0)
- 0(0), 15(60), 0(0)

Για την περίοδο 2

- 0(0), 2(42), 14(3), 18(50), 5(96), 3(85), 9(39), 1(119), 16(66), 0(0)
- 0(0), 11(64), 10(81), 15(78), 0(0)
- 0(0), 11(64), 17(128), 7(97), 14(3), 4(83), 6(42), 20(85), 0(0)

Για την περίοδο 3

- 0(0), 11(17), 10(105), 2(91), 20(42), 16(110), 13(100), 0(0)
- 0(0), 3(81), 5(66), 18(120), 4(40), 14(96), 6(97), 0(0)
- 0(0), 15(62), 12(80), 19(160), 0(0)

Για την περίοδο 4

- 0(0), 16(19), 1(36), 9(81), 3(78), 5(79), 4(48), 14(80), 6(30), 20(49), 0(0)
- 0(0), 2(77), 7(95), 17(94), 8(65), 10(56), 11(14), 15(99), 0(0)

Για την περίοδο 5

- 0(0), 2(48), 11(55), 10(68), 8(65), 12(67), 15(29), 0(0)
- 0(0), 13(71), 16(33), 1(42), 3(75), 5(44), 18(37), 4(16), 14(61), 6(64), 20(57), 0(0)

Δεδομένα του IRP προβλήματος 3 για 30 πελάτες και 3 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	672.13	414.44	0	0	9000	0
1	643.48	332.24	3	4.76	300	17
2	828.01	721.32	3	5.49	300	96
3	204.47	473.85	1	11.295	110	59
4	617.49	798.64	2	7.578	150	82

5	617.70	90.87	3	4.443	300	79
6	301.07	874.56	1	10.604	110	101
7	871.74	776.98	2	7.912	150	20
8	589.65	771.57	3	3.611	300	20
9	981.77	63.10	3	2.069	300	7
10	442.23	594.44	3	5.944	300	78
11	126.32	792.49	1	11.098	110	72
12	508.83	274.13	1	9.582	110	47
13	431.79	371.02	3	5.219	300	33
14	915.94	853.11	3	2.301	300	101
15	709.02	678.73	1	10.094	110	62
16	890.66	754.38	3	4.698	300	76
17	588.89	417.26	3	4.756	300	40
18	636.83	320.90	3	2.179	300	21
19	342.21	372.30	3	3.29	300	19
20	823.79	275.35	2	8.455	150	42
21	305.15	679.56	2	8.478	150	16
22	307.54	581.63	2	8.435	150	21
23	201.32	970.20	3	5.391	300	60
24	265.16	415.51	2	6.929	150	79
25	650.92	330.49	3	2.227	300	41
26	154.61	966.71	2	8.104	150	109
27	388.70	825.90	3	4.828	300	86
28	647.61	305.40	3	2.178	300	61
29	848.30	823.28	3	2.715	300	20
30	63.84	556.91	1	10.781	110	63

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3
0	3862	3392	3824
1	44	50	26
2	54	67	63
3	35	37	50
4	84	55	68
5	96	80	61
6	63	78	62
7	99	29	39
8	100	52	77
9	33	69	95
10	33	66	28
11	53	77	100
12	20	37	34
13	27	71	72
14	17	28	85
15	42	49	57
16	59	20	10
17	72	27	48
18	45	87	37
19	60	92	54
20	29	13	83
21	43	90	27
22	34	13	42
23	91	67	23
24	91	53	78
25	95	65	60
26	63	85	88
27	76	80	13
28	30	14	44
29	75	49	36
30	84	41	78

Στη λύση του IRP προβλήματος 3 για 30 πελάτες και 3 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :
Για την περίοδο 1

- 0(0), 7(79), 14(29), 29(109), 4(2), 8(80), 23(31), 21(27), 22(13), 30(21), 24(12), 19(41), 17(32), 0(0)
- 0(0), 9(190), 5(158), 18(24), 1(27), 25(54), 0(0)

Για την περίοδο 2

- 0(0), 1(50), 18(87), 25(65), 0(0)
- 0(0), 10(21), 6(40), 23(67), 26(39), 11(58), 30(41), 3(13), 24(53), 19(92), 13(65), 12(10), 0(0)
- 0(0), 15(29), 2(25), 16(13), 7(68), 29(31), 4(55), 8(52), 27(70), 21(117), 22(13), 17(27), 0(0)

Για την περίοδο 3

- 0(0), 12(34), 13(64), 0(0)
- 0(0), 15(57), 2(63), 4(68), 8(77), 10(28), 22(42), 19(54), 13(64), 24(78), 3(50), 30(78), 11(100), 26(88), 23(23), 6(62), 27(13), 0(0)
- 0(0), 17(48), 0(0)
- 0(0), 20(83), 28(27), 18(37), 1(26), 25(60), 0(0)

Δεδομένα του IRP προβλήματος 3 για 30 πελάτες και 5 χρονικές περιόδους

Σημείο	X	Y	Τύπος	h	Max I	I_0
0	672.13	414.44	0	0	15000	0
1	643.48	332.24	3	4.76	500	65
2	828.01	721.32	3	5.49	500	105
3	204.47	473.85	1	11.295	110	77
4	617.49	798.64	2	7.578	250	85
5	617.70	90.87	3	4.443	500	36
6	301.07	874.56	1	10.604	110	93
7	871.74	776.98	2	7.912	250	42
8	589.65	771.57	3	3.611	500	25
9	981.77	63.10	3	2.069	500	91
10	442.23	594.44	3	5.944	500	53
11	126.32	792.49	1	11.098	110	90
12	508.83	274.13	1	9.582	110	96
13	431.79	371.02	3	5.219	500	28
14	915.94	853.11	3	2.301	500	14
15	709.02	678.73	1	10.094	110	42
16	890.66	754.38	3	4.698	500	102
17	588.89	417.26	3	4.756	500	14
18	636.83	320.90	3	2.179	500	91
19	342.21	372.30	3	3.29	500	103
20	823.79	275.35	2	8.455	250	48
21	305.15	679.56	2	8.478	250	102
22	307.54	581.63	2	8.435	250	6
23	201.32	970.20	3	5.391	500	67
24	265.16	415.51	2	6.929	250	55
25	650.92	330.49	3	2.227	500	84
26	154.61	966.71	2	8.104	250	31
27	388.70	825.90	3	4.828	500	77
28	647.61	305.40	3	2.178	500	26
29	848.30	823.28	3	2.715	500	14
30	63.84	556.91	1	10.781	110	18

Τα προϊόντα διαθέσιμα προς αποστολή και οι ζητήσεις κάθε περιόδου

Σημείο	Περίοδος 1	Περίοδος 2	Περίοδος 3	Περίοδος 4	Περίοδος 5
0	4153	3999	3279	4465	4302

1	44	50	26	54	67
2	63	35	37	50	84
3	55	68	96	80	61
4	63	78	62	99	29
5	39	100	52	77	33
6	69	95	33	66	28
7	53	77	100	20	37
8	34	27	71	72	17
9	28	85	42	49	57
10	59	20	10	72	27
11	48	45	87	37	60
12	92	54	29	13	83
13	43	90	27	34	13
14	42	91	67	23	91
15	53	78	95	65	60
16	63	85	88	76	80
17	13	30	14	44	75
18	49	36	84	41	78
19	27	69	92	89	30
20	30	17	88	82	57
21	43	72	86	50	31
22	29	52	26	31	70
23	89	51	96	71	30
24	73	18	66	94	54
25	79	50	13	61	70
26	65	84	84	54	90
27	12	19	99	33	64
28	29	61	100	18	40
29	94	52	57	41	22
30	61	48	33	27	24

Στη λύση του IRP προβλήματος 3 για 30 πελάτες και 5 χρονικές περιόδους τα δρομολόγια και οι ποσότητες παράδοσης ανά περίοδο είναι :

Για την περίοδο 1

- 0(0), 2(0), 7(11), 14(119), 29(80), 15(11), 8(9), 10(6), 22(23), 21(0), 23(22), 26(34), 30(43), 24(18), 13(15), 5(3), 28(3), 0(0)

Για την περίοδο 2

- 0(0), 1(29), 25(45), 28(61), 12(50), 5(100), 9(170), 0(0)
- 0(0), 10(20), 22(52), 21(13), 6(71), 23(51), 26(84), 11(3), 30(48), 3(46), 24(18), 13(90), 0(0)
- 0(0), 15(78), 8(27), 4(56), 29(52), 7(77), 16(46), 0(0)
- 0(0), 17(29), 0(0)

Για την περίοδο 3

- 0(0), 4(62), 8(71), 27(53), 6(33), 23(96), 26(84), 11(87), 10(10), 0(0)
- 0(0), 2(30), 16(88), 7(100), 14(90), 29(57), 15(95), 0(0)
- 0(0), 12(29), 13(27), 19(85), 24(66), 3(96), 30(33), 21(86), 22(26), 17(14), 0(0)
- 0(0), 20(87), 5(52), 28(100), 18(78), 1(26), 25(13), 0(0)

Για την περίοδο 4

- 0(0), 1(54), 18(55), 28(58), 25(61), 0(0)
- 0(0), 10(72), 21(50), 22(31), 19(85), 13(34), 0(0)
- 0(0), 20(139), 5(110), 12(13), 19(85), 24(94), 3(80), 30(27), 11(37), 26(54), 23(71), 6(66), 27(33), 0(0)

- 0(0), 15(65), 2(50), 16(76), 7(20), 29(41), 4(99), 8(72), 17(44), 0(0)

Για την περίοδο 5

- 0(0), 2(84), 16(80), 7(37), 14(91), 29(22), 15(60), 0(0)
- 0(0), 17(75), 10(27), 21(31), 22(70), 13(13), 12(83), 18(64), 1(67), 25(70), 0(0)
- 0(0), 4(29), 8(17), 27(64), 6(28), 23(30), 26(90), 11(60), 30(24), 3(61), 24(54), 19(30), 0(0)