ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΕΦΑΡΜΟΓΗ ΜΕΘΕΥΡΕΤΙΚΩΝ ΜΕΘΟΔΩΝ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

Διπλωματική Εργασία

του

Αθανάσιου Τσιφτσή

Θεσσαλονίκη, Νοέμβριος 2018

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΕΦΑΡΜΟΓΗ ΜΕΘΕΥΡΕΤΙΚΩΝ ΜΕΘΟΔΩΝ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

Αθανάσιος Τσιφτσής

Πτυχίο Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών, ΑΠΘ, 2014

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Άγγελος Σιφαλέρας

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ..../..../....

Ονοματεπώνυμο 1	Ονοματεπώνυμο 2	Ονοματεπώνυμο 3
.....	.....	.....

Αθανάσιος Τσιφτσής

.....

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως σκοπό την παρουσίαση της υλοποίησης και εφαρμογής διαφόρων μεθευρετικών μεθόδων για την εύρεση λύσης σε γνωστά προβλήματα βελτιστοποίησης. Ειδικότερα, οι μεθευρετικές μέθοδοι που έχουν υλοποιηθεί είναι η προσομοιωμένη ανόπτηση (Simulated Annealing, SA), η αναζήτηση με χρήση απαγορευμένων κινήσεων (Tabu Search, TS) και η βελτιστοποίηση αποικίας μυρμηγκιών (Ant Colony Optimization, ACO) που ανήκει στην οικογένεια μεθόδων εμπνευσμένων από τη φύση, και ειδικότερα τη Νοημοσύνη Σμήνους (Swarm Intelligence). Τα προβλήματα στα οποία εφαρμόζονται οι παραπάνω αλγόριθμοι είναι το πρόβλημα του πλανόδιου πωλητή (Travelling Salesman Problem, TSP) και το πρόβλημα σακιδίου (Knapsack Problem). Για την υλοποίηση των αλγορίθμων χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python (v.3.6).

Στο πρώτο μέρος της διπλωματικής εργασίας παρουσιάζονται τα παραπάνω προβλήματα καθώς και οι μεθευρετικές μέθοδοι. Ειδικότερα, για την κάθε οικογένεια προβλημάτων, γίνεται μία ιστορική αναδρομή, καταγράφεται η περιγραφή του προβλήματος καθώς και η μαθηματική μοντελοποίησή του ενώ γίνεται και αναφορά στις διαφορές παραλλαγές του προβλήματος. Σε ό,τι αφορά τις μεθευρετικές μεθόδους, μετά από μία σύντομη αναφορά στην ιστορία της μεθόδου, παρουσιάζονται τα κύρια βήματα του κάθε αλγορίθμου καθώς και τα ιδιαίτερα χαρακτηριστικά της κάθε μεθόδου.

Στο δεύτερο μέρος, παρουσιάζονται τα αποτελέσματα της εφαρμογής των μεθευρετικών μεθόδων στα παραπάνω προβλήματα καθώς και κάποια συμπεράσματα που εξάγονται από αυτά, τόσο για τον κάθε έναν αλγόριθμο ξεχωριστά όσο και συγκριτικά αποτελέσματα μεταξύ των αλγορίθμων.

Λέξεις Κλειδιά: Ευρετικές Μέθοδοι, Μεθευρετικές Μέθοδοι, Προβλήματα Βελτιστοποίησης, Προσομοιωμένη Ανόπτηση, Αναζήτηση με Απαγορευμένες Κινήσεις, Αποικία Μυρμηγκιών, Νοημοσύνη Σμήνους, Τοπική Αναζήτηση, Βελτιστοποίηση, Πρόβλημα Πλανόδιου Πωλητή, Πρόβλημα Σακιδίου

Abstract

This diploma thesis aims at presenting the implementation and application of various metaheuristic methods used in order to find solutions to some of the most famous optimization problems. In particular, the methods that have been implemented are Simulated Annealing (SA), Tabu Search (TS) and Ant Colony Optimization (ACO). The latter belongs to the family of methods inspired by nature (Swarm Intelligence). The problems in which the above algorithms are applied are the Traveling Salesman Problem (TSP) and the Knapsack Problem. The Python programming language was used to implement the algorithms (v.3.6).

In the first part of the diploma thesis the above problems are presented as well as the metaheuristic methods used. For each family of problems, a historical review is made while the description of the problem and its mathematical modeling are recorded. Also, the different variants of each problem are mentioned. Regarding the metaheuristic methods, after a brief reference to the history of each method, the main steps of each algorithm are presented, as well as the particular features of each method.

In the second part, we present the results of the implementation of the algorithms when applied in the above problems as well as some conclusions derived from them, both for each algorithm separately and comparative results between the algorithms.

Keywords: Heuristics, Metaheuristics, Optimization Problems, Simulated Annealing, Tabu Search, Ant Colony Optimization, Particle Swarm Optimization, Local Search, Travelling Salesman Problem, Knapsack Problem

Ευχαριστίες

Η παρούσα διπλωματική εργασία είναι αφιερωμένη σε όλους τους ανθρώπους που στήριξαν την συγκεκριμένη προσπάθεια.

Ιδιαίτερα, θα ήθελα να ευχαριστήσω από καρδιάς τον επιβλέποντα καθηγητή κ. Άγγελο Σιφαλέρα για την υπομονή που επέδειξε και την καθοδήγηση που μου παρείχε για την ολοκλήρωση του συγκεκριμένου εγχειρήματος.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συμβολισμοί

TSP : Travelling Salesman Problem

KS : Knapsack Problem

ACO : Ant Colony Optimization

SA : Simulated Annealing

TS : Tabu Search

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Το θέμα που πραγματεύεται η εργασία είναι οι μεθευρετικές μέθοδοι ως εργαλείο εύρεσης λύσης σε προβλήματα βελτιστοποίησης. Ως προβλήματα βελτιστοποίησης εννοούνται εκείνα τα προβλήματα που μπορούν να έχουν πολλές λύσεις αλλά με τη χρήση διαφόρων μετρικών, γνωστών ως αντικειμενικών συναρτήσεων, αποτιμάται η ποιότητα της κάθε λύσης. Η εύρεση της καλύτερης δυνατής λύσης είναι αυτή που ελαχιστοποιεί ή μεγιστοποιεί (ανάλογα με το πρόβλημα και την χρήση της μετρικής για την ποιότητα της λύσης), πάντως βελτιστοποιεί την τιμή της αντικειμενικής συνάρτησης.

Η χρήση μεθόδων ακριβείας για την εύρεση της βέλτιστης λύσης σε προβλήματα τέτοιου τύπου πολλές φορές είναι αρκετά κοστοβόρα σε υπολογιστική πολυπλοκότητα ενώ άλλες φορές δεν έχει αποδειχθεί ότι η λύση που νομίζεται ως βέλτιστη, πράγματι αποτελεί και την καλύτερη λύση. Επιπρόσθετα, υπάρχουν αρκετές περιπτώσεις κατά τις οποίες δεν ενδιαφερόμαστε τόσο για την εύρεση της καλύτερης δυνατής λύσης αλλά για μία αρκετά καλή λύση, που μπορεί απλά να ικανοποιεί κάποιες προϋποθέσεις που έχουμε θέσει. Συνήθως, αυτό γίνεται με στόχο να αποφύγουμε τη σπατάλη επιπρόσθετου χρόνου ώστε να βρούμε την κορυφαία λύση καθώς είμαστε ικανοποιημένοι με μία λιγότερο ποιοτική λύση, η οποία όμως βρίσκεται πάντα κοντά στη βέλτιστη και σίγουρα είναι αποδεκτή αφού θα ικανοποιεί τις ελάχιστες προϋποθέσεις που θα έχουν τεθεί, αρκεί αυτή να βρεθεί σε σύντομο χρονικό διάστημα ή με τη χρήση συγκεκριμένης ποσότητας πεπερασμένων πόρων (π.χ. υπολογιστική ισχύς). Αυτήν ακριβώς την ανάγκη, δηλαδή την εύρεση αρκούντως ικανοποιητικής λύσης σε σύντομο χρονικό διάστημα, ικανοποιούν οι ευρετικές και μεθευρετικές μέθοδοι επίλυσης τέτοιων προβλημάτων.

1.2 Σκοπός – Στόχοι

Σκοπός της εργασίας είναι η υλοποίηση των πιο ευρέως διαδεδομένων μεθευρετικών μεθόδων για την επίλυση κάποιων από τα πιο πολυσυζητημένα προβλήματα βελτιστοποίησης. Επίσης, μέσα στους στόχους της παρούσας διπλωματικής εργασίας, είναι η σύγκριση των υλοποιημένων μεθόδων για την επίλυση του ίδιου προβλήματος ώστε να εξαχθούν συμπεράσματα σχετικά με την αποτελεσματικότητα του κάθε αλγορίθμου. Επιπρόσθετα, επιχειρείται και μία σύγκριση των αποτελεσμάτων του κάθε αλγορίθμου όταν ανατίθενται διαφορετικές τιμές στις παραμέτρους του, ώστε να ανοίξει η συζήτηση σχετικά με την επιλογή των καταλληλότερων τιμών που θα πρέπει να έχουν οι παράμετροι ώστε η μέθοδος να είναι όσο το δυνατόν πιο επιτυχημένη.

Ειδικότερα, τα προβλήματα που θίγονται είναι αυτό του Πλανόδιου Πωλητή (Travelling Salesman Problem, εφεξής TSP) και αυτό του 0-1 διδιάστατου προβλήματος σακιδίου (0-1 2-dimensional Knapsack Problem, εφεξής KP). Λεπτομερής περιγραφή του κάθε προβλήματος δίνεται αναλυτικά παρακάτω. Ακόμη, σε ό,τι αφορά τις υλοποιηθείσες μεθευρετικές μεθόδους, αυτές είναι η Προσομοιωμένη Ανόπτηση, η αναζήτηση με απαγορευμένες κινήσεις, γνωστή και ως αναζήτηση Tabu, και η βελτιστοποίηση με χρήση αποικίας μυρμηγκιών.

1.3 Βασική Ορολογία

Βελτιστοποίηση (Optimization): Ο όρος βελτιστοποίηση στα εφαρμοσμένα μαθηματικά αναφέρεται στην αναζήτηση βέλτιστων παραμέτρων ενός – συνήθως περίπλοκου – συστήματος. Προβλήματα βελτιστοποίησης απαντώνται σε πολλά επιστημονικά πεδία όπως π.χ. στη φυσική, στη χημεία, στην οικονομία κ.α.. Στα μαθηματικά διατυπώνεται ένα πρόβλημα βελτιστοποίησης σαν πρόβλημα ελαχιστοποίησης ή μεγιστοποίησης μιας αντικειμενικής συνάρτησης μίας μεταβλητής ή πολλών μεταβλητών.

Συνδυαστική βελτιστοποίηση (Combinatorial Optimization): Γνωστή και ώς διακριτή βελτιστοποίηση, είναι ένα αντικείμενο που συνίσταται στην εξεύρεση του βέλτιστου από ένα πεπερασμένο σύνολο αντικειμένων.

Ευρετικές μέθοδοι (Heuristic methods): Είναι μέθοδοι που σχεδιάστηκαν για τη γρηγορότερη επίλυση ενός προβλήματος, όταν οι κλασσικές μέθοδοι είναι πολύ αργές ή για την εύρεση μιας προσεγγιστικής λύσης, όταν οι κλασσικές μέθοδοι αδυνατούν να βρουν οποιαδήποτε λύση.

Μεθευρετικές μέθοδοι (Metaheuristic methods): Είναι μέθοδοι ή ευρετικές μέθοδοι υψηλού επιπέδου σχεδιασμένες να βρίσκουν, να δημιουργούν ή να επιλέγουν μια μέθοδο ή μια ευρετική χαμηλότερου επιπέδου, η οποία μπορεί να παρέχει μια αρκετά καλή λύση σε ενα πρόβλημα βελτιστοποίησης, ιδίως με ελλιπείς ή ατελείς πληροφορίες ή με περιορισμένη υπολογιστική ικανότητα.

Προσεγγιστικοί αλγόριθμοι (Approximation Algorithms): Αλγόριθμοι που εκτελούνται συνήθως σε πολυωνυμικό χρόνο βρίσκοντας λύσεις εγγυημένα κοντά στη βέλτιστη λύση.

Αναπαράσταση Λύσης: Κωδικοποιεί εναλλακτικές υποψήφιες λύσεις που θα μελετηθούν. Είναι πολύ σημαντικό να επιλεγεί σωστή αναπαράσταση λύσης ούτως ώστε να μη συγκαταλέγονται στο χώρο αναζήτησης μη εφικτές λύσεις και να μην υπάρχουν πολλαπλές αναπαραστάσεις της ίδιας λύσης.

Συνάρτηση Αξιολόγησης ή Αντικειμενική Συνάρτηση (evaluation function): Επιστρέφει μια συγκεκριμένη τιμή που εκφράζει την ποιότητα των συγκεκριμένων λύσεων που περιέχει η αναπαράσταση (ή τουλάχιστον, μια σύγκριση της ποιότητας δυο εναλλακτικών λύσεων).

Χώρος αναζήτησης: Υπονοείται από τον τρόπο αναπαράστασης λύσης. Είναι όλες οι πιθανές λύσεις που μπορούν να αναπαρασταθούν με τον επιλεγμένο τρόπο αναπαράστασης.

1.4 Διάρθρωση της μελέτης

Η μελέτη που έχει πραγματοποιηθεί περιλαμβάνει τέσσερις ενότητες που περιγράφονται συνοπτικά ως εξής:

Στην πρώτη ενότητα (Κεφάλαιο 1), που είναι εισαγωγική, παρουσιάζονται οι στόχοι της διπλωματικής εργασίας, οι ορισμοί κάποιων βασικότατων όρων που συναντώνται στη μελέτη του συκεκριμένου αντικειμένου και φυσικά γίνεται περιληπτική αναφορά στη διάρθρωση της εργασίας.

Στη δεύτερη ενότητα (Κεφάλαιο 2) παρουσιάζεται το θεωρητικό μέρος της μελέτης, το οποίο χωρίζεται σε δύο υποκεφάλαια. Στο πρώτο από αυτά παρουσιάζονται τα βασικά στοιχεία δύο πολύ γνωστών προβλημάτων συνδυαστικής βελτιστοποίησης. Αυτά είναι το Πρόβλημα του Πλανόδιου Πωλητή και το Πρόβλημα Σακιδίου. Στο δεύτερο υποκεφάλαιο αυτής της ενότητας παρουσιάζονται οι υλοποιηθείσες μεθευρετικές μέθοδοι προς επίλυση των παραπάνω προβλημάτων. Αυτές είναι η Αναζήτηση με χρήση Απαγορυμένων Κινήσεων (Tabu Search), ο αλγόριθμος βελτιστοποίησης με χρήση Αποικίας Μυρμηγκιών (Ant Colony Optimization) και η μεθευρετική μέθοδος της Προσομοιωμένης Ανόπτησης (Simulated Annealing).

Στην τρίτη ενότητα (Κεφάλαιο 3) παρουσιάζονται τα αποτελέσματα της εκτέλεσης των παραπάνω αλγορίθμων για την επίλυση των αναφερθέντων προβλημάτων καθώς και κάποια συμπεράσματα που αφορούν την ανάθεση διαφορετικών τιμών στις παραμέτρους των αλγορίθμων ούτως ώστε να παραχθεί το καλύτερο αποτέλεσμα μεταξύ των διαφόρων δοκιμών.

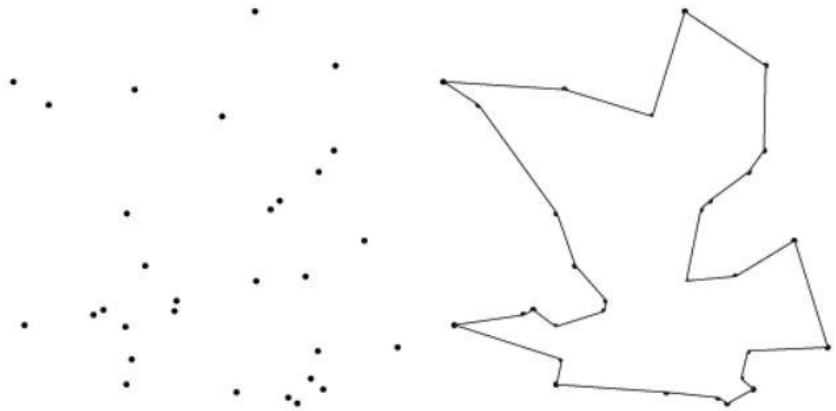
Τέλος, στην τέταρτη (Κεφάλαιο 4) και τελευταία ενότητα πραγματοποιείται μία μικρή σύνοψη της εργασίας ενώ επίσης γίνεται αναφορά και σε πιθανές μελλοντικές επεκτάσεις αυτής.

2 Βιβλιογραφική Επισκόπηση – Θεωρητικό Υπόβαθρο

2.1 Προβλήματα βελτιστοποίησης

2.1.1 Το Πρόβλημα του Πλανόδιου Πωλητή (Travelling Salesman Problem, TSP)

Ένα από τα πιο γνωστά προβλήματα στη συνδυαστική βελτιστοποίηση αλλά και στην επιστήμη των υπολογιστών είναι το Πρόβλημα του Πλανόδιου Πωλητή (TSP), στο οποίο είναι δεδομένα ένα σύνολο από πόλεις και αποστάσεις μεταξύ κάθε δεδομένου ζεύγους πόλεων και το ζητούμενο είναι η εύρεση εκείνου του μονοπατιού που επισκέπτεται κάθε πόλη μία φορά και επιστρέφει στην αρχική πόλη, με τη συνολική απόσταση που διανύθηκε να είναι η ελάχιστη.



Εικόνα 1 : Οι θέσεις των κόμβων και το μονοπάτι με την ελάχιστη απόσταση για ένα πρόβλημα πλανόδιου πωλητή 30 κόμβων

Το TSP, από την εποχή της καθιέρωσής του τη δεκαετία του 1930, είναι σημαντικό καθώς είναι ταυτόχρονα δύσκολο (NP – hard) και αντιπροσωπευτικό (NP – complete). Τα προβληματα NP – hard (Non – deterministic Polynominal hard) είναι δυσεπίλυτα υπολογιστικά προβλήματα υπό την έννοια ότι δεν έχουν βρεθεί γνωστές λύσεις σε πολυωνυμικό χρόνο, σε αντίθεση με την οικογένεια των προβλημάτων P που μπορούν να λυθούν σε ακριβή πολυωνυμικό χρόνο. Ως υποσύνολο των προβλημάτων NP, τα προβλήματα NP – complete είναι εκείνα των οποίων οι λύσεις είναι επαρκείς για να αντιμετωπίσουν οποιοδήποτε άλλο πρόβλημα NP σε πολυωνυμικό χρόνο. Έτσι, αν

θα μπορούσαν να βρεθούν αποτελεσματικές λύσεις για κάθε πρόβλημα NP – complete, συμπεριλαμβανομένου και του Προβλήματος του Πλανώδιου Πωλητή (TSP), τότε θα είμαστε σε θέση να επιλύσουμε κατηγορηματικά το αναπάντητο ερώτημα, αν P = NP.

2.1.1.1. Ιστορική αναδρομή

Τα ιστορικά στοιχεία για την πρώτη αναφορά στο συγκεκριμένο πρόβλημα δεν είναι σαφή. Υπάρχουν ενδείξεις ότι οι πρώτες αναφορές στο Πρόβλημα του Πλανόδιου Πωλητή έγιναν τον 18ο αιώνα. Στην ουσία, η πρώτη αναφορά περιέχει παραδείγματα περιηγήσεων, τα οποία δεν θεμελιώθηκαν μαθηματικά και δεν είχαν κανέναν επιστημονικό υπόβαθρο. Συγκεκριμένα, το 1832 τυπώθηκε ένα εγχειρίδιο στη Γερμανία, το οποίο, παρόλο που στο μεγαλύτερο μέρος του πραγματεύεται άλλα θέματα, στο τελευταίο κεφάλαιο αναφέρεται στην ουσία του TSP προβλήματος. Εκτός από το γεγονός ότι αναφερεται για πρώτη φορά το πρόβλημα TSP, περιλαμβάνονται και κάποια παραδείγματα διαδρομών μεταξύ Γερμανίας και Ελβετίας.

Η πρώτη σαφής αναφορά του προβλήματος έγινε τον 19ο αιώνα από τον Ιρλανδό μαθηματικό Sir William Rowan Hamilton και τον Βρετανό μαθηματικό Thomas Kirkman. Ο Hamilton δημιούργησε ένα παιχνίδι γνωστό ως «Δωδεκάεδρο του Ταξιδιώτη», το οποίο έχει αρκετές ομοιότητες με το πρόβλημα TSP. Στο συγκεκριμένο παιχνίδι, σκοπός ήταν η εύρεση ενός κύκλου ο οποίος διέρχεται από τις κορυφές ενός δωδεκάεδρου με την προϋπόθεση ότι κάθε κορυφή – κόμβος μπορούμε να την επισκεφθούμε μόνο μια φορά ενώ στο τελικό στάδιο ο κύκλος έπρεπε να επιστρέψει στο αρχικό του σημείο.

Πριν από τον Sir W. R. Hamilton, οι Euler (1759) και Vandermonde (1771) έκαναν λόγο για το πρόβλημα της διαδρομής του ιππότη (Knight’s Tour). Η διαδρομή ενός ιππότη είναι ένας Χαμιλτονιανός κύκλος, οι κορυφές από τις οποίες διέρχεται είναι τα 64 τετράγωνα μιας σκακιέρας με δύο κορυφές παρακείμενες εάν και μόνο εάν ένας ιππότης μπορεί να μετακινηθεί με ένα μόνο βήμα από το ένα τετράγωνο στο άλλο.

Κατά την διάρκεια του 20ου αιώνα, περί το 1930 στη Βιέννη και στο Χάρβαρντ, μελετήθηκε και καθιερώθηκε η γενική μορφή του προβλήματος από τον Μαθηματικό Karl Menger (1902 – 1985), ο οποίος παρατήρησε τη μη βελτιστότητα της ευρετικής μεθόδου του πλησιέστερου γείτονα (Nearest Neighbor Heuristic). Η ονομασία TSP προήλθε λίγα χρόνια αργότερα από τον Αμερικάνο Μαθηματικό Hassler Whitney (1907

– 1989) του Πανεπιστημίου Princeton, παρόλο που δεν είναι επιστημονικά αποδεδειγμένο κάτι τέτοιο.

Τις επόμενες δεκαετίες, το πρόβλημα TSP γνώρισε ιδιαίτερη αναγνώριση τόσο στην Αμερική όσο και στην Ευρώπη ενώ σπουδαίοι Μαθηματικοί και άλλοι Επιστήμονες ανά τον κόσμο συνέβαλλαν στην εξέλιξή του. Για παράδειγμα, την δεκαετία του 1950 οι George Dantzig, Delbert Ray Fulkerson και Selmer M. Johnson αντιμετώπισαν το πρόβλημα ως πρόβλημα ακέραιου γραμμικού προγραμματισμού για την επίλυση ενός προβλήματος που περιλάμβανε τη σύνδεση 49 πόλεων. Αργότερα, τις δεκαετίες του 1970 και 1980, αρκετοί επιστήμονες χρησιμοποιώντας τεχνικές, όπως branch – and – bound και cutting – planes, κατάφεραν να δώσουν λύση σε προβλήματα που είχαν ως κόμβους αρκετές χιλιάδες πόλεις (Grötschel, Padberg και Rinaldi). Το 1972 ο Richard M. Karp έδειξε ότι το πρόβλημα του Χαμιλτονιανού κύκλου ήταν NP – complete, πράγμα που συνεπάγεται την NP – δυσεπιλυσιμότητα του TSP. Το γεγονός αυτό παρείχε μια επιστημονική εξήγηση για την υπολογιστική δυσκολία εύρεσης βέλτιστων διαδρομών.

Την δεκαετία του 1990 οι David Applegate, Robert E. Bixby, Vasek Chvatal και William J. Cook δημιούργησαν ένα λογισμικό επίλυσης του συγκεκριμένου προβλήματος, το οποίο ονομάσθηκε «Concorde TSP Problem». Το 1991, ο Gerhard Reinelt δημοσίευσε το TSPLIB, μια συλλογή προβλημάτων αναφοράς μεταβλητής δυσκολίας, η οποία έχει χρησιμοποιηθεί από πολλές ερευνητικές ομάδες για τη σύγκριση αποτελεσμάτων.

Τέλος, το 2004, οι ίδιοι επιστήμονες, Applegate, Bixby, Chvatal και Cook, μαζί με τον Keld Helsgaun κατάφεραν να λύσουν ένα πρόβλημα με 24978 κόμβους – πόλεις της Σουηδίας. Λίγο αργότερα, το 2006 ο Cook και η επιστημονική του ομάδα επίλυσαν ένα πρόβλημα 85900 κόμβων, οι οποίοι αναπαριστούσαν τη σύνδεση των τρανζίστορ σε ένα μικροεπεξεργαστή.

2.1.1.2. Μαθηματική μοντελοποίηση του TSP

Για τη διαμόρφωση ενός μαθηματικού προτύπου υποθέτουμε ότι οι κόμβοι ανήκουν σε ένα μη διατεταγμένο γράφο που είναι πλήρης. Έστω $i=1$ ο κόμβος της αφετηρίας και $i=2, \dots, n$ οι κόμβοι των πελατών. Από την υπόθεση, κάθε μη-διατεταγμένο ζεύγος $\{i, j\}$ με $i \neq j, i = 1, \dots, n, j = 1, \dots, n$ αντιστοιχεί σε ένα σύνδεσμο ή ακμή του γραφήματος. Σε κάθε τέτοιο σύνδεσμο αντιστοιχίζουμε ένα μέγεθος c_{ij} που

αναπαριστά το κόστος (όπως και αν νοείται αυτό) της διαδρομής του οχήματος από το i στο j ή αντίστροφα. Επειδή ένας τέτοιος σύνδεσμος δεν αντιστοιχεί πάντοτε σε κάποιο φυσικό τμήμα δρόμου θα υποθέσουμε ότι τα μεγέθη αυτά έχουν υπολογιστεί έτσι ώστε να αντιστοιχούν στη διαδρομή ελαχίστου κόστους μεταξύ των δυο κόμβων, οπότε ικανοποιούν τις δυο συνθήκες που ακολουθούν:

1. Συμμετρία: $c_{ij} = c_{ji}, i \neq j, i = 1, \dots, n, j = 1, \dots, n$
2. Τριγωνική Ανισότητα: $c_{ij} \leq c_{ik} + c_{jk}, i \neq j \neq k, i = 1, \dots, n, j = 1, \dots, n, k = 1, \dots, n$

οι οποίες υπό αυτές τις συνθήκες είναι φυσιολογικές, καθώς η μεν πρώτη καθιστά ανεξάρτητο το κόστος του απευθείας ταξιδιού μεταξύ i και j από την κατεύθυνση του ταξιδιού, ενώ η δεύτερη διατυπώνει ότι ο συντομότερος δρόμος μεταξύ δυο σημείων είναι ο απευθείας.

Αν ορίσουμε τις δυαδικές μεταβλητές

$$x_{ij} = \begin{cases} 1 \text{ εάν το όχημα κάνει χρήση του συνδέσμου } \{i,j\} \\ 0 \text{ αλλιώς, για κάθε } i,j \text{ όπου } i \neq j \end{cases}$$

τότε το πρότυπο διαμορφώνεται ως εξής:

$min \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_{ij}$

s.t

$\sum_{i=1, i \neq j}^{m} x_{ij} = 1$ for all $j = 1..m$

$\sum_{j=1, i \neq j}^{m} x_{ij} = 1$ for all $i = 1..m$

$\sum_{i \in K} \sum_{j \in K} x_{ij} \leq |K| - 1$ for all $K \subset \{2..m\}$

$x_{ij} = 0$ or 1 for all i, j

Οι δύο πρώτοι, κατά σειρά, περιορισμοί επιβάλλουν στη λύση να έχει δυο συνδέσμους σε κάθε κόμβο έτσι ώστε το όχημα να εισέλθει κατά μήκος του ενός και να εξέλθει κατά μήκος του άλλου.

Οι επόμενοι περιορισμοί αποβλέπουν στην εξάλειψη κυκλικών διαδρομών (υποκύκλων) που δεν διέρχονται από όλους τους κόμβους απαιτώντας από κάθε εν δυνάμει υποκύκλο, που αντιπροσωπεύεται από ένα κατάλληλο μη κενό υποσύνολο K των κόμβων, να διαθέτει στην λύση τουλάχιστον ένα σύνδεσμο που οδηγεί στο συμπληρωματικό του υποσύνολο $K' = \{1, \dots, n\} \setminus K$.

2.1.1.3. Κατηγοριοποίηση του TSP

Στη συνέχεια, θα ασχοληθούμε με τις παραλλαγές του προβλήματος. Σε γενικές γραμμες ταξινομείται σε α) συμμετρικό TSP, β) ασύμμετρο TSP και γ) πολλαπλό TSP.

2.1.1.3.1. Το συμμετρικό TSP (symmetric TSP – sTSP)

Έστω $V = \{v_1, \dots, v_n\}$ ένα σύνολο πόλεων, $A = \{(r,s) : r,s \in V\}$ το σύνολο ακμών και $d_{rs} = d_{sr}$ είναι ένα μέτρο κόστους που σχετίζεται με την ακμή $(r,s) \in A$. Το sTSP είναι το πρόβλημα της εύρεσης μιας κλειστής διαδρομής ελάχιστου μήκους, η οποία να επισκέπτεται κάθε πόλη μία ακριβώς φορά. Στην περίπτωση που οι πόλεις $v_i \in V$ δίνονται από τις συντεταγμένες τους (x_i, y_i) και d_{rs} είναι η Ευκλείδεια απόσταση μεταξύ r και s, προκύπτει το λεγόμενο «Ευκλείδειο TSP».

2.1.1.3.2. Το ασύμμετρο TSP (asymmetric TSP – aTSP)

Σε αυτή την περίπτωση, το κόστος διαδρομής από μια πόλη A σε μια πόλη B δεν είναι απαραίτητα το ίδιο με το κόστος διαδρομής από την πόλη B στην πόλη A. Αυτό αντικατοπτρίζεται διατυπώνοντας το ασύμμετρο TSP (aTSP) ως την εύρεση ενός ελάχιστου κατευθυνόμενου Χαμιλτονιανού κύκλου σε ένα σταθμισμένο γράφημα. Έστω ότι $D = (W, A), W = \{1, \dots, n\}, A \subseteq W \times W$ είναι ο διγράφος για τον οποίο το aTSP πρέπει να λυθεί. Έστω ότι d_{ij} είναι η απόσταση από τον κόμβο i στον j. Ορίζουμε ένα μη κατευθυνόμενο γράφημα $G(V,E)$ με:

$$V = W \cap \{n+1, \dots, 2n\}$$

$$E = \{(i, n+1)/i = 1, \dots, n\} \cup \{(n+i, j)/i, j \in A\}$$

Τα βάρη των ακμών υπολογίζονται ως ακολούθως:

$$c_{i,n+1} = -M, i = 1, \dots, n$$

$$c_{n+1,j} = d_{ij}, (i,j) \in A$$

όπου M ένας πολύ μεγάλος αριθμός.

Είναι εύκολο να δει κανείς ότι για κάθε κατευθυνόμενο Χαμιλτονιανό κύκλο στο D με μήκος d_D υπάρχει ένας Χαμιλτονιανός κύκλος στο G με μήκος $CG = d_D - nM$. Επιπρόσθετα, όλες οι ακμές με βάρος $-M$ περιέχονται σε ένα βέλτιστο Χαμιλτονιανό κύκλο στο G. Επομένως, αυτός ο κύκλος επιφέρει ένα κατευθυνόμενο Χαμιλτονιανό κύκλο στο D.

2.1.1.3.3. Το πολλαπλό TSP (multi TSP – mTSP)

Αντί για ένα μόνο πωλητή έχουμε m πωλητές διαθέσιμους, οι οποίοι βρίσκονται όλοι στην ίδια πόλη και πρέπει να επισκεφτούν τις πόλεις. Το κόστος της λύσης είναι η συνολική απόσταση που διανύεται από το σύνολο των πωλητών (όλοι πρέπει να ταξιδέψουν). Αυτή

είναι η βασική περίπτωση όταν, στην δρομολόγηση οχημάτων (Vehicle Routing Problem), οχήματα βρίσκονται σε ένα κοινό σταθμό και πρέπει να εξυπηρετήσουν όλους τους πελάτες.

2.1.1.4 Αναπαράσταση Λύσης

Στο πρόβλημα TSP n κόμβων μία αναπαράσταση λύσης που είναι ευρέως διαδεδομένη είναι η μετάθεση φυσικών αριθμών $1, 2, \ldots, n$, όπου κάθε αριθμός αντιστοιχεί σε μια πόλη που θα επισκεφτεί ο πωλητής με διαδοχική σειρά. Με αυτήν την αναπαράσταση, ο χώρος αναζήτησης αποτελείται από όλες τις πιθανές μεταθέσεις. Υπάρχουν $n!$ τέτοιες μεταθέσεις, κάτι που καθιστά προφανή πλέον το λόγο που το πρόβλημα πλανόδιου πωλητή είναι δυσεπίλυτο. Ωστόσο, για το συμμετρικό TSP, όπου το κόστος μετακίνησης από την πόλη i στην πόλη j είναι το ίδιο με το κόστος μετακίνησης από την πόλη j στην πόλη i, δεν μας ενδιαφέρει αν προσπελάσουμε τη λίστα κόμβων από αριστερά προς τα δεξιά ή αντίστροφα. Επομένως, ο χώρος αναζήτησης περιορίζεται κατά το ήμισυ. Επιπρόσθετα, δεν μας απασχολεί ο κόμβος εκκίνησης, καθώς πάντα το τέλος της διαδρομής καταλήγει σε αυτόν τον κόμβο. Αυτό σημαίνει ότι το μέγεθος του χώρου αναζήτησης μειώνεται κατά συντελεστή n. Τελικά, το πραγματικό μέγεθος του χώρου αναζήτησης για το συμμετρικό TSP είναι $\frac{(n-1)!}{2}$.

Με βάση την αναπαράσταση λύσης, προσδιορίζεται και ο στόχος που πρέπει να επιτύχουμε με την επίλυση του προβλήματος. Στο TSP , λοιπόν, ο στόχος είναι η εύρεση εκείνης της μετάθεσης των φυσικών αριθμών που αντιστοιχούν στις πόλεις του γραφήματος έτσι ώστε να ελαχιστοποιείται η απόσταση που διένυσε ο πωλητής, λαμβάνοντας υπόψη τον περιορισμό της επίσκεψης σε κάθε πόλη μόνο μία φορά και της επιστροφής στην πόλη που αποτέλεσε την αφετηρία της διαδρομής.

2.1.1.5. Συνάρτηση αξιολόγησης

Όπως σε κάθε πρόβλημα, έτσι και στο TSP, πρέπει να υπάρχει ένας τρόπος αξιολογείται η κάθε υποψήφια λύση. Έτσι, ορίζεται μία αντικειμενική συνάρτηση βάσει της οποίας γίνεται η αξιολόγηση. Αυτή η συνάρτηση αποτελεί μία απεικόνιση από των χώρο των υποψήφιων δυνατών λύσεων, όπως αυτές προκύπτουν από τον επιλεγμένο τρόπο αναπαράστασης λύσης, σε ένα συνολο αριθμών. Η τιμή που παίρνει αυτή η συνάρτηση για την κάθε υποψήφια λύση εκφράζει την ποιότητα της λύσης. Ανάλογα με το πρόβλημα και τη συνάρτηση, άλλοτε επιδιώκεται η μεγιστοποίηση και άλλοτε η ελαχιστοποίηση αυτής της συνάρτησης. Στο συγκεκριμένο πρόβλημα, η συνάρτηση

αξιολόγησης είναι η συνολική απόσταση της κάθε διαδρομής και επιδιώκεται η ελαχιστοποίησή της.

2.1.1.6 Κατασκευή αρχικής λύσης

Για να εφαρμοστεί οποιαδήποτε μεθευρετική μέθοδος πάνω σε ένα πρόβλημα Συνδυαστικής Βελτιστοποίησης για να βρει μία αρκετά ικανοποιητική λύση, θα πρέπει πρώτα με κάποιο τρόπο να έχει δημιουργηθεί μία αρχική λύση που απλά είναι εφικτή, ούτως ώστε αυτή να αποτελέσει το αρχικό σημείο εκκίνησης από την οποία θα ξεκινήσει η μεθευρετική μέθοδος για να εξερευνήσει το χώρο αναζήτησης, βελτιώνοντας την τιμή τιμή της αντικειμενικής συνάρτησης.

Για το πρόβλημα του πλανόδιου πωλητή, έχει προταθεί μεταξύ άλλων, η κατασκευή αρχικής λύσης μέσω μίας κατασκευαστικής ευρετικής μεθόδου (constructive heuristic) που ονομάζεται nearest-neighbour (NN). Σύμφωνα με αυτήν τη μέθοδο, ξεκινώντας από ένα δεδομένο κόμβο εκκίνησης, αναζητούμε τον κοντινότερο σε αυτόν κόμβο και συνδέουμε αυτούς τους δύο κόμβους. Στη συνέχεια, αναζητούμε για τον δεύτερο κόμβο, τον κοντινότερο γειτονικό κόμβο (εκτός του σημείου εκκίνησης) και βάζουμε και αυτόν στο μονοπάτι που κατασκευάζουμε. Συνεχίζουμε για κάθε νέο κόμβο συνδέοντάς τον με όποιον κόμβο (από αυτούς που δεν έχουν ήδη εισαχθεί στο μονοπάτι) είναι κοντινότερος, μέχρι να εξαντληθεί η λίστα των κόμβων οπότε συνδέουμε τον τελευταίο με τον κόμβο εκκίνησης για να κλείσει η διαδρομή.

2.1.2 Το Πρόβλημα του Σακιδίου (Knapsack Problem, KP)

Το Πρόβλημα του Σακιδίου είναι ένα από τα πιο γνωστά προβλήματα στη συνδυαστική βελτιστοποίηση. Ο ορισμός του προβλήματος είναι πολύ απλός. Ας υποθέσουμε ότι μας δίνεται ένα σύνολο n στοιχείων , το καθένα με συγκεκριμένο βάρος s_i και συγκεκριμένη αξία v_i. Επίσης, έστω ότι έχουμε ένα σακίδιο B, το οποίο έχει συγκεκριμένη χωριστικότητα αναφορικά με το βάρος. Το πρόβλημα έγκειται στο να βρεθεί ποια είναι η μέγιστη αξία των αντικειμένων που μπορούν να μεταφερθούν στο σακίδιο, $S \subseteq [n]$, με την προϋπόθεση ότι το συνολικό βάρος όλων των αντικειμένων που συλλέγονται δεν πρέπει να υπερβαίνει το βάρος του σακιδίου, δηλαδή στόχος είναι η μεγιστοποίηση του αθρόσματος $\sum_{i \in S} v_i$ υπό τη συνθήκη $\sum_{i \in S} s_i \leq B$. Υποθέτουμε ότι δεν υπάρχει αντικείμενο με μέγεθος μεγαλύτερο του B, επειδή δεν μπορούμε να διαλέξουμε τέτοιου είδους αντικείμενο ούτως ή άλλως.

Υπάρχουν πολλές γνωστές παραλλαγές του προβλήματος σακιδίου. Το πρόβλημα του σακιδίου και οι παραλλαγές του είναι γνωστά NP – hard προβλήματα. Ο κύριος λόγος διαφοροποίησης είναι οι περιορισμοί που επιβάλλονται στη λύση όσον αφορά τον αριθμό αντιγράφων κάθε στοιχείου που μπορούν να μεταφερθούν στο σακίδιο ή στις μονάδες υποδιαίρεσης των αντιγράφων (σε περίπτωση που ο σάκος μπορεί να γεμίσει με μη ακέραιες τιμές ένα αντικείμενο). Για n υποψήφια αντικείμενα, όλοι οι πιθανοί συνδυασμοί αντικειμένων είναι 2^n, επομένως η επίλυση του προβλήματος σακιδίου με πλήρη απαρίθμηση των πιθανών λύσεων καθίσταται πρακτικά αδύνατη για μεγάλο n.

2.1.2.1. Ιστορική αναδρομή – Παραλλαγές

Το πρόβλημα του σακιδίου έχει μελετηθεί από τα τέλη του 17^{ου} αιώνα. Ο ορισμός του προβλήματος και η ονομασία του δόθηκαν από τον Μαθηματικό Tobias Dantzig (Numbers: The Language of Science, 1930). Έκτοτε, η έρευνά τους εντάθηκε για δύο σημαντικούς λόγους: Πρώτον, τα προβλήματα αυτά έχουν άμεση εφαρμογή στη βιομηχανία, στη μηχανική, στην οικονομική διαχείριση και γενικότερα σε οποιονδήποτε τομέα υπάρχει ένα μοναδικό και σπάνιο αγαθό – πόρος που τον διεκδικούν πολλοί και δεύτερον, για θεωρητικούς λόγους αφού τα Knapsack problems εμφανίζονται σε προβλήματα ακέραιου προγραμματισμού.

Το πρόβλημα Σακιδίου μελετάται εκτενώς και τις τελευταίες δεκαετίες καθώς παρουσιάζει θεωρητικό και πρακτικό ενδιαφέρον. Το θεωρητικό του ενδιαφέρον έγκειται στο γεγονός ότι η δομή του είναι τόσο απλή που από τη μια επιτρέπει την εκμετάλλευση

πολλών συνδυαστικών ιδιοτήτων και από την άλλη πολύπλοκα προβλήματα βελτιστοποίησης μπορούν να λυθούν μέσω μερικών knapsack – type υποπροβλημάτων. Από την πρακτική πλευρά, αυτού του είδους προβλήματα μπορούν να μοντελοποιήσουν πολλές καταστάσεις της βιομηχανίας, όπως προβλήματα διαχείρισης κεφαλαίου, φόρτωση φορτίου, cutting – stock problem κ.α.

Οι παραλλαγές των Knapsack problems χωρίζονται σε 2 μεγάλες κατηγορίες, οι οποίες με τη σειρά τους διαιρούνται σε υποκατηγορίες ανάλογα με τη διαθεσιμότητα ενός ή περισσοτέρων σακιδίων. Πιο συγκεκριμένα,

- Single Knapsack Problems: στην περίπτωση αυτή υπάρχει μόνο ένα σακίδιο που θα πρέπει να γεμίσει με το βέλτιστο σύνολο αντικειμένων. Στην κατηγορία αυτή ανήκουν τα εξής: α) 0 – 1 Knapsack problem, β) Bounded Knapsack problem, γ) Subset – sum problem και δ) Change – making problem.
- Multiple Knapsack Problems: αντίθετα εδώ είναι διαθέσιμα περισσότερα από ένα σακίδια. Στη κατηγορία αυτή συναντάμε τα εξής: α) 0 – 1 Multiple Knapsack problem, β) Generalized assignment problem και γ) Bin – packing problem.

2.1.2.2.0 – 1 Knapsack Problem

Το 0 – 1 ή Binary Knapsack Problem είναι το πρόβλημα εκείνο, όπου θα πρέπει να επιλεχθεί από ένα σύνολο αντικείμενων ένα υποσύνολο αυτών, με σκοπό το αντίστοιχο συνολικό κέρδος να είναι το μέγιστο, χωρίς όμως το συνολικό βάρος των αντικειμένων να ξεπερνά την καθορισμένη χωρητικότητα του σακιδίου.

2.1.2.2.1. Μαθηματική μοντελοποίηση

Έστω ότι διαθέτουμε ένα σακίδιο που μπορεί να αντέξει βάρος W. Ας θεωρήσουμε ότι διαθέτουμε n αντικείμενα, όπου το κάθε αντικείμενο i έχει αξία p_i και βάρος w_i. Τότε, αν επιπρόσθετα ορίσουμε δυαδικές μεταβλητές απόφασης x_i, οι οποίες παίρνουν τιμές 1 ή 0 ανάλογα με το αν το αντικείμενο i επιλέγεται να τοποθετηθεί εντός του σακιδίου ή όχι αντίστοιχα, το πρόβλημα του σακιδίου μπορεί να μοντελοποιηθεί ως εξής:

$$\max \sum_{i=1}^{n} p_i x_i$$

$$s.t. \sum_{i=1}^{n} w_i x_i \leq W$$

$$x_i \in \{0,1\}, \qquad i = 1, \dots, n$$

Η παραπάνω μαθηματική μοντελοποίηση μπορεί να επεκταθεί εύκολα και για πολυδιάστατα 0-1 προβλήματα σακιδίου καθώς ανάλογος περιορισμός με αυτόν του βάρους W μπορεί να εφαρμοστεί και για άλλα χαρακτηριστικά (διαστάσεις) του σακιδίου. Για παράδειγμα, αν υπήρχε περιορισμός και ως προς τον όγκο που χωρά το σακίδιο, δηλαδή επιπλέον ήταν γνωστά ο όγκος V που χωράει στο σακίδιο και ο όγκος v_i του κάθε αντικειμένου i, το πρόβλημα θα ήταν διδιάστατο και θα έπρεπε να ικανοποιείται επιπρόσθετα και ο περιορισμός $s.t. \sum_{i=1}^{n} v_i x_i \leq V$.

2.1.2.2.2. Αναπαράσταση Λύσης

Για το πρόβλημα σακιδίου με n αντικείμενα υποψήφια προς εισαγωγή έχουν υπάρξει διάφορες προσεγγίσεις σε ό,τι αφορά την αναπαράσταση λύσης. Οι δύο επικρατέστερες είναι οι ακόλουθες :

1) Η κάθε λύση μπορεί να αναπαρασταθεί ως ένα μονοδιάστατο διάστημα μήκους n, με την κάθε θέση του διανύσματος να είναι μία δυαδική μεταβλητή που μπορεί να πάρει τις τιμές 1 ή 0 ανάλογα με το αν το αντίστοιχο αντικείμενο έχει επιλεγεί να εισαχθεί στο σακίδιο ή όχι. Για παράδειγμα, αν έχουμε 5 αντικείμενα προς εισαγωγή και τελικά εισάγουμε το πρώτο και το τέταρτο, η λύση μας αναπρίσταται με το διάνυσμα [1 0 0 1 0].
2) Η κάθε λύση αναπαρίσταται με τη χρήση ενός διανύσματος το οποίο έχει μήκος ανάλογο με το πλήθος των αντικειμένων που συμμετέχουν στη λύση. Έτσι, για το παραπάνω παράδειγμα, η λύση αναπαρίσταται με αυτόν τον τρόπο ως [1 4]. Φυσικά, η αντιστοίχιση του κάθε αντικειμένου στον αριθμό που το αντιπροσωπεύει μπορεί να γίνει με διάφορους τρόπους. Για παράδειγμα, μπορεί η αρίθμηση των αντικείμενων να ξεκινά από το 0, επομένως η λύση του παραπάνω παραδείγματος σε αυτήν την περίπτωση θα είναι [0 3].

2.1.2.2.3. Συνάρτηση αξιολόγησης

Για το πρόβλημα σακιδίου, η συνάρτηση αξιολόγησης είναι η συνολική αξία των αντικειμένων που αποτελούν την κάθε λύση. Άρα, κάθε υποψήφια λύση, εφόσον είναι εφικτή, δηλαδή ικανοποιεί τους περιορισμούς του προβλήματος, αντιστοιχίζεται στην

τιμή που έχει το άθροισμα της αξίας των αντικειμένων που περιέχει. Όπως γίνεται άμεσα κατανοητό και από την περιγραφή του προβλήματος, επιδιώκεται η μεγιστοποίηση αυτής της συνάρτησης υπό τον περιορισμό του συνολικού βάρους του σακιδίου ή, αν πρόκειται για πολυδιάστατο πρόβλημα, και τυχόν επιπλέον περιορισμών.

2.1.2.2.4. Κατασκευή αρχικής λύσης

Για την κατασκευή αρχικής λύσης στο πρόβλημα του σακιδίου, μία από τις προσεγγίσεις που έχουν επικρατήσει είναι αυτή που ταξινομεί τα αντικείμενα κατά φθίνουσα σειρά σύμφωνα με την τιμή που έχει ο λόγος $\alpha\xi\acute{\iota}\alpha/\beta\acute{\alpha}\rho o\varsigma$. Στη συνέχεια, επιλέγονται τα πρώτα αντικείμενα αυτής της ταξινόμησης που ικανοποιούν τον περιορισμό του συνολικού βάρους του σακιδίου. Δηλαδή, αν με την εισαγωγή στο σακίδιο του k −στού κατά σειρά αντικειμένου παύει να ικανοποιείται ο περιορισμός του συνολικού βάρους που χωρά το σακίδιο, επιλέγονται τα πρώτα $(k-1)$ αντικείμενα. Με άλλα λόγια, τα πρώτα $(k-1)$ αντικείμενα της παραπάνω ταξινομημένης λίστας τα οποία χωρούν στο σακίδιο, επιλέγονται για να συμμετέχουν στην αρχική λύση, όπου $k \leq n$ και n το πλήθος του συνόλου των αντικειμένων του προβλήματος.

Η παραπάνω προσέγγιση μπορεί να επεκταθεί και σε προβλήματα με περισσότερους από έναν περιορισμό. Απλά στον παρονομαστή του λόγου θα πρέπει να τοποθετηθεί μία σχέση που περιέχει μέσα όλα τα μεγέθη για τα οποία υπάρχει περιορισμός. Για παράδειγμα, για το 0-1 πρόβλημα δύο διαστάσεων, στο οποίο οι δύο περιορισμοί αφορούν το βάρος και τον όγκο του σακιδίου, η συνάρτηση που μπορεί να χρησιμοποιηθεί για τηνταξινόμηση των αντικειμένων είναι ο λόγος $\alpha\xi\acute{\iota}\alpha/(\alpha \cdot \beta\acute{\alpha}\rho o\varsigma + \beta \cdot \acute{o}\gamma\kappa o\varsigma)$. Οι συντελεστές α και β επιλέγονται με τέτοιο τρόπο ώστε να εξισορροπούν τους όρους τους αθροίσματος καθώς, ανάλογα με τις μονάδες μέτρησης του κάθε μεγέθους μπορεί οι αριθμητικές τιμές μεταξύ των μεγεθών να έχουν πολύ μεγάλη διαφορά μεταξύ τους, τέτοια ώστε το απλό άθροισμα βάρος + όγκος να μην λαμβάνει κατ'ουσία υπόψη το ένα εκ των δύο μεγεθών. Αυτό μπορεί να γίνει πιο κατανοητό με το ακόλουθο παράδειγμα. Αν τα αντικείμενα του προβλήματος έχουν τιμές αξίας που κυμαίνονται περίπου στις $10\ \mu o\nu\acute{\alpha}\delta\varepsilon\varsigma$, τιμές βάρους που κυμαίνονται γύρω στα $100\ kg$ και τιμές όγκου που κυμαίνονται γύρω στα $0{,}1\ m^3$, ο λόγος $\frac{10}{100+0{,}1}$ ουσιαστικά δεν λαμβάνει σχεδόν καθόλου υπόψη τον όγκο ως μέγεθος. Αντιθέτως, αν επιλέξουμε $\alpha = 1$ και $\beta = 1000$, τότε ο λόγος $\frac{10}{1\cdot 100+1000\cdot 0{,}1} = \frac{10}{100+100}$, λαμβάνει υπόψη τα μεγέθη που ορίζουν τους περιορισμούς με ίδια βαρύτητα.

2.2. Μεθευρετικές μέθοδοι επίλυσης προβλημάτων Συνδυαστικής Βελτιστοποίησης

2.2.1. Αναζήτηση με χρήση απαγορευμένων κινήσεων, Tabu Search (TS)

Η αναζήτηση με χρήση απαγορευμένων κινήσεων, γνωστή και ως αναζήτηση Tabu, είναι ένας μεθευρετικός αλγόριθμος, που μπορεί να χρησιμοποιηθεί για την επίλυση συνδυαστικών προβλημάτων βελτιστοποίησης. Η αναζήτηση αυτή χρησιμοποιεί μια διαδικασία τοπικής αναζήτησης γειτονιάς για να μετακινηθεί από μια δυνητική λύση x σε μια βελτιωμένη λύση x' στη γειτονιά του x.

Οι τοπικές αναζητήσεις γειτονιάς λαμβάνουν μια πιθανή λύση σε ένα πρόβλημα και ελέγχουν τους άμεσους γείτονές του (δηλαδή λύσεις που είναι παρόμοιες) με την ελπίδα να βρεθεί μια βέλτιστη λύση. Οι τοπικές μέθοδοι αναζήτησης έχουν την τάση να κολλάνε σε υποβέλτιστες περιοχές ή σε ένα σταθερό πεδίο, όπου πολλές λύσεις είναι εξίσου κατάλληλες.

Η αναζήτηση Tabu ενισχύει την απόδοση μιας τοπικής αναζήτησης «χαλαρώνοντας» τον βασικό της κανόνα. Αρχικά, σε κάθε βήμα μπορούν να γίνουν αποδεκτές κινήσεις επιδείνωσης, εάν δεν υπάρχει διαθέσιμη βέλτιστη λύση (όπως όταν η αναζήτηση είναι κολλημένη σε αυστηρό τοπικό ελάχιστο). Αυτό το χαρακτηριστικό γνώρισμα της μεθόδου υπάρχει για να δίνεται διέξοδος από κατάσταση εγκλωβισμού σε τοπικά βέλτιστα, δηλαδή τόπικα μέγιστα ή ελάχιστα ανάλογα με το αν επιδιώκεται η μεγιστοποίηση ή η ελαχιστοποίηση της αντικειμενικής συνάρτησης. Επιπλέον, οι απαγορεύσεις (Tabu) εισάγονται για να αποθαρρύνουν την αναζήτηση από την επιστροφή σε λύσεις που είχαν βρεθεί προηγουμένως και άρα δεν αξίζει να επανεξεταστούν.

Η εφαρμογή της αναζήτησης Tabu χρησιμοποιεί δομές μνήμης που περιγράφουν τις λύσεις που έχουν βρεθεί ήδη μια φορά ή σύνολα κανόνων που παρέχονται από το χρήστη. Εάν μια πιθανή λύση έχει προηγουμένως βρεθεί εντός μιας συγκεκριμένης βραχυπρόθεσμης περιόδου ή εάν έχει παραβιάσει έναν κανόνα, επισημαίνεται ως "Tabu" (απαγορευμένη), έτσι ώστε ο αλγόριθμος να μην εξετάζει επανειλημμένα αυτήν την πιθανότητα, χάνοντας σημαντικό χρόνο αλλά και κινδυνεύοντας να παγιδευτεί σε ένα ατέρμονο βρόχο χωρίς να μπορεί να απεγκλωβιστεί.

Παρακάτω, φαίνεται ο αλγόριθμος της αναζήτησης με χρήση απαγορευμένων κινήσεων σε μορφή ψευδοκώδικα:

```
Given a feasible solution x* with objective function value z*:
Let x := x* with z(x) = z*.
Iteration:
while stopping criterion is not fulfilled do
      begin
      (1) select best admissible move that transforms x into x' with objective function value
         z(x') and add its attributes to the running list
      (2) perform tabu list management: compute moves (or attributes) to be set tabu, i.e.,
         update the tabu list
      (3) perform exchanges:
      x = x', z(x) = z(x');
      if z(x) < z* then
            z* = z(x), x* = x
      endif
endwhile
Result: x* is the best of all determined solutions, with objective function value z*.
```

Εικόνα 2 : Ο ψευδοκώδικας της μεθευρετικής μεθόδου Tabu Search

2.2.1.1. Ιστορική αναδρομή

Η αναζήτηση Tabu δημιουργήθηκε από τον Fred W. Glover το 1986 (Future Paths for Integer Programming and Links to Artificial Intelligence) και επισημοποιήθηκε το 1989 [Glover (1989) Tabu Search – Part 1, Glover (1990) Tabu Search – Part 2]. Η λέξη Tabu προέρχεται από τη "Tongan" – εθνική γλώσσα της Τόνγκα – για να υποδείξει πράγματα που δεν μπορούν να αγγιχτούν γιατί είναι ιερά.

Οι τρέχουσες εφαρμογές της TS εκτείνονται στους τομείς του σχεδιασμού πόρων, των τηλεπικοινωνιών, του σχεδιασμού VLSI, της οικονομικής ανάλυσης, του προγραμματισμού, του χωροταξικού σχεδιασμού, της διανομής ενέργειας, της μοριακής μηχανικής, της υλικοτεχνικής υποδομής, της ταξινόμησης των προτύπων, της ευέλικτης κατασκευής, της διαχείρισης των αποβλήτων, της μεταλλευτικής έρευνας και σε πληθώρα άλλων.

Τα τελευταία χρόνια, έχουν δημοσιευθεί σε αρκετά επιστημονικά περιοδικά διαφόρων επιστημονικών πεδίων άρθρα με υπολογιστικές μελέτες που τεκμηριώνουν επιτυχίες με την αναζήτηση Tabu, επεκτείνοντας έτσι τα σύνορα των προβλημάτων που μπορούν αποτελεσματικά να αντιμετωπιστούν αποδίδοντας λύσεις, των οποίων η ποιότητα ξεπερνά κατά πολύ εκείνη που επιτεύχθηκε με μεθόδους που εφαρμόστηκαν στο παρελθόν.

2.2.1.2. Δομές μνήμης

Οι δομές μνήμης που χρησιμοποιούνται στην αναζήτηση Tabu χωρίζονται σε τρεις κατηγορίες:

- Βραχυπρόθεσμες: κατάλογος των λύσεων που εξετάστηκαν πρόσφατα. Εάν μια πιθανή λύση εμφανίζεται στη λίστα των tabu, δεν μπορεί να επανεξεταστεί μέχρι να φτάσει σε ένα σημείο λήξης.
- Ενδιάμεσες: κανόνες εντατικοποίησης που αποσκοπούν στην κλίση της αναζήτησης προς πολλά υποσχόμενες περιοχές του χώρου αναζήτησης.
- Μακροπρόθεσμες: κανόνες διαφοροποίησης που οδηγούν την αναζήτηση σε νέες περιοχές (δηλ. όσον αφορά την επαναφορά όταν η αναζήτηση κολλήσει σε ένα σταθερό πεδίο λύσεων ή σε ένα υποβέλτιστο αδιέξοδο).

Οι βραχυπρόθεσμες, ενδιάμεσες και μακροπρόθεσμες δομές μνήμης μπορούν να επικαλύπτονται στην πράξη. Στις κατηγορίες αυτές, η μνήμη μπορεί να διαφοροποιηθεί περαιτέρω με μέτρα όπως η συχνότητα και το μέγεθος της αποτελεσματικότητας των αλλαγών που έγιναν.

Η βραχυπρόθεσμη δομή μνήμης από μόνη της μπορεί να είναι αρκετή για να επιτευχθεί λύση ανώτερη από εκείνη που διαπιστώνεται με τις συμβατικές τοπικές μεθόδους αναζήτησης, αλλά οι ενδιάμεσες και οι μακροπρόθεσμες δομές είναι συχνά απαραίτητες για την επίλυση δυσκολότερων προβλημάτων.

Η αναζήτηση Tabu συχνά συγκρίνεται με άλλες μεθευρετικές μεθόδους, όπως η προσομοιωμένη ανόπτηση (Simulated Annealing), οι γενετικοί αλγόριθμοι (Genetic Algorithms), οι αλγόριθμοι βελτιστοποίησης αποικιών (Ant Colony Optimization) ή η κατευθυνόμενη τοπική αναζήτηση. Επιπλέον, η αναζήτηση Tabu συνδυάζεται μερικές φορές με άλλες μεθευρετικές μεθόδους για τη δημιουργία υβριδικών μεθόδων ώστε να εξαχθεί ένα ακόμα καλύτερο αποτέλεσμα, που θα είναι πολύ κοντά στην πραγματικά βέλτιστη λύση δυσεπίλυτων προβλημάτων.

2.2.1.3. Μέθοδος τοπικής αναζήτησης

Δεδομένης μίας εφικτής λύσης στο εκάστοτε πρόβλημα βελτιστοποίησης στο οποίο εφαρμόζεται ο αλγόριθμος, πρέπει να γίνει με κάποιο τρόπο μία τοπική αναζήτηση σε άλλες λύσεις, γειτονικές της τρέχουσας, ώστε μέσα από αυτό το σύνολο λύσεων να επιλεχθεί η νέα τρέχουσα λύση του αλγορίθμου. Αυτομάτως, κανείς αναρωτιέται με ποιο ακριβώς τρόπο πραγματοποιείται αυτή η τοπική αναζήτηση γύρω από την τρέχουσα

λύση. Η αλήθεια είναι ότι έχουν προταθεί πολλές και διάφορες μέθοδοι τοπικής αναζήτησης. Όλες σχετίζονται άμεσα με τον τρόπο αναπαράστασης λύσης, καθώς ουσιαστικά πρέπει να συμβεί κάποια μικρή διαφοροποίηση της υπάρχουσας λύσης για να παραχθεί η επόμενη. Ασφαλώς, ο τρόπος εύρεσης της γειτονιάς εξαρτάται και από το πρόβλημα που κάθε φορά καλείται η μεθευρετική μέθοδος να επιλύσει, αφού ο τρόπος αναπαράστασης της λύσης είναι άρρηκτα συνδεδεμένος με το προς επίλυση πρόβλημα.

Στο πλαίσιο της παρούσας εργασίας, η μεθευρετική μέθοδος Tabu Search υλοποιήθηκε για την επίλυση τόσο του συμμετρικού προβλήματος του πλανόδιου πωλητή (sTSP) όσο και του διδιάστατου 0-1 προβλήματος σακιδίου (KP).

Για το sTSP χρησιμοποιήθηκε ως μέθοδος τοπικής αναζήτησης γειτονιάς η μέθοδος της αμοιβαίας μετάθεσης μεταξύ δύο κόμβων (swap) της τρέχουσας λύσης. Για παράδειγμα, αν σε ένα γράφημα τεσσάρων κόμβων, Α,Β,Γ,Δ η τρέχουσα λύση είναι η [Α, Β, Γ, Δ], τότε μια γειτονική λύση αυτής μπορεί να πραγματοποιηθεί με αμοιβαία μετάθεση δύο εκ των κόμβων του προβλήματος. Αν αυτοί οι κόμβοι είναι οι Β, Δ, τότε η γειτονική λύση που προκύπτει από την αμοιβαία μετάθεση αυτών των δύο κόμβων είναι η [Α, Δ, Γ, Β]. Η γειτονιά της τρέχουσας λύσης [Α, Β, Γ, Δ] αποτελείται από το σύνολο όλων των γειτονικών λύσεων. Οπότε, στο παράδειγμα που εξετάζεται, δεδομένου ότι ο πωλητής ξεκινά πάντα από τον κόμβο Α, η γειτονιά αποτελείται από τις λύσεις : [Α, Β, Δ, Γ], [Α, Γ, Β, Δ], [Α, Δ, Γ, Β].

Σε ό,τι αφορά το πρόβλημα KP, η μέθοδος τοπικής αναζήτησης που χρησιμοποιήθηκε είναι η δυνητική αντικατάσταση μεταξύ δύο υποψήφιων αντικειμένων : ενός που συμμετέχει στην τρέχουσα λύση, δηλαδή έχει επιλεχθεί για εισαγωγή στο σακίδιο και ενός άλλου που δεν συμμετέχει στην τρέχουσα λύση. Ωστόσο, αυτή η αντικατάσταση είναι δυνητική, δηλαδή επιτρέπει την εισαγωγή του ενός αντικειμένου χωρίς την εξαγωγή του άλλου αλλά και την εξαγωγή του ενός του αντικειμένου χωρίς την εισαγωγή του άλλου. Ασφαλώς, η κάθε κίνηση που έχει επιλεχθεί πρέπει πρώτα να εξεταστεί ως προς την εφικτότητά της, καθώς ενδέχεται να μην ικανοποιείται κάποιος από τους περιορισμούς που υπάρχουν στο πρόβλημα του σακιδίου, όπως η υπέρβαση της συνολικής χωρητικότητας του σακιδίου.

2.2.2. Μέθοδος βελτιστοποίησης αποικιών (Ant Colony Optimization)

Η βελτιστοποίηση αποικίας μυρμηγκιών (ACO – Ant Colony Optimization) είναι μια μεθευρετική μέθοδος βασισμένη στον πληθυσμό που μπορεί να χρησιμοποιηθεί για να βρει κατά προσέγγιση λύσεις σε δύσκολα προβλήματα βελτιστοποίησης. Είναι εμπνευσμένη από την συμπεριφορά αναζήτησης τροφής των μυρμηγκιών στις αποικίες τους.

Στη βελτιστοποίηση αποικίας μυρμηγκιών ένα σύνολο παραγόντων λογισμικού που ονομάζονται «τεχνητά μυρμήγκια» αναζητούν τις βέλτιστες λύσεις για ένα δεδομένο πρόβλημα βελτιστοποίησης, το οποίο έχει μετατραπεί σε πρόβλημα εύρεσης του μονοπατιού ελάχιστου κόστους σε ένα σταθμισμένο γράφημα. Τα «τεχνητά μυρμήγκια» χτίζουν σταδιακά λύσεις με την κίνησή τους στο γράφημα. Η διαδικασία επίλυσης είναι στοχαστική και επηρεάζεται από ένα μοντέλο φερομόνης, δηλαδή, ένα σύνολο παραμέτρων που σχετίζονται με τα χαρακτηριστικά του γραφήματος (κόμβους ή ακμές) οι τιμές των οποίων τροποποιούνται κατά το χρόνο της εκτέλεσης του αλγορίθμου από τα «τεχνητά μυρμήγκια».

Η ACO έχει εφαρμοστεί με επιτυχία σε πολλά κλασικά προβλήματα συνδυαστικής βελτιστοποίησης, καθώς και σε προβλήματα διακεκριμένης βελτιστοποίησης που έχουν στοχαστικές ή και δυναμικές συνιστώσες. Παραδείγματα αποτελούν τα αιτήματα δρομολόγησης σε δίκτυα επικοινωνιών και η στοχαστική εκδοχή του γνωστού προβλήματος συνδυαστικής βελτιστοποίησης του πλανόδιου πωλητή, που έχει παρουσιαστεί αναλυτικά παραπάνω. Επιπλέον, η ACO έχει επεκταθεί έτσι ώστε να μπορεί να χρησιμοποιηθεί για την επίλυση προβλημάτων βελτιστοποίησης συνεχών και μικτών μεταβλητών (Socha και Dorigo). Η βελτιστοποίηση αποικίας μυρμηγκιών είναι ίσως το πιο επιτυχημένο παράδειγμα πληροφοριακού συστήματος νοημοσύνης σμήνους με πολυάριθμες εφαρμογές σε προβλήματα του πραγματικού κόσμου.

2.2.2.1 Ιστορική αναδρομή

Ο αλγόριθμος ACO προτάθηκε αρχικά μέσα από τη διδακτορική διατριβή του Marco Dorigo το 1992. Είναι ο πρώτος αλγόριθμος που αποσκοπεί στην αναζήτηση μιας βέλτιστης διαδρομής σε ένα γράφο με βάση την συμπεριφορά των μυρμηγκιών που αναζητούν μια διαδρομή από την αποικία προς την περιοχή εύρεσης της τροφής τους. Η αρχική αυτή ιδέα διαφοροποιήθηκε και επεκτάθηκε ώστε να υπηρετήσει την επίλυση μιας ευρύτερης κατηγορίας υπολογιστικών προβλημάτων, έχοντας σαν αποτέλεσμα την

δημιουργία αρκετών προβλημάτων τα οποία βασίζονται στις διαφορετικές πτυχές της συμπεριφοράς των μυρμηγκιών.

Στην κλασσική μορφή του αλγορίθμου ο Marco Dorigo εξέδωσε το 1997 (Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem) μία νεότερη εκδοχή, με καλύτερα αποτελέσματα στις δοκιμαστικές συγκρίσεις. Η νέα εκδοχή, που είναι η αποικία μυρμηγκιών, διαφοροποιείται στον τρόπο επιλογής της διαδρομής από τα μυρμήγκια, στον κανόνα ανανέωσης της φερορμόνης στο τέλος κάθε γύρου (global updating rule) και κατά την πορεία των μυρμηγκιών κατά την διάρκεια του γύρου (local updating rule). Σε αυτήν την έρευνα, ο τελευταίος κανόνας δεν χρησιμοποιείται.

2.2.2.2. Οι κυριότερες μορφές του αλγορίθμου

Στη βιοβλιογραφία έχουν προταθεί αρκετοί αλγόριθμοι βασισμένοι στη λειτουργία των αποικιών μυρμηγκιών. Παρακάτω αναφέρονται μερικοί από αυτούς.

Ο πρωτότυπος **Ant System (AS)** είναι ο πρώτος ACO αλγόριθμος που προτάθηκε στη βιβλιογραφία (Dorigo, Maniezzo and Colorni 1991). Το κύριο χαρακτηριστικό του είναι ότι, σε κάθε επανάληψη, οι ποσότητες φερομόνης ενημερώνονται από όλα τα μυρμήγκια που έχουν καταφέρει, στην επανάληψη αυτή, να κατασκευάσουν μία λύση.

Οι δύο πλέον επιτυχημένες παραλλαγές του είναι ο MAX – MIN Ant System (Stutzle and Hoos 2000) και ο Ant Colony System (Dorigo and Gambardella 1997). Ο αλγόριθμος **MAX – MIN Ant System (MMAS)** είναι μια βελτίωση του αρχικού Ant System. Χαρακτηριστικά στοιχεία του είναι ότι μόνο το "καλύτερο" μυρμήγκι ενημερώνει τα μονοπάτια φερομόνης και ότι η ποσότητα της φερομόνης είναι περιορισμένη από άνω και κάτω όρια. Τα όρια αυτά συνήθως καθορίζονται εμπειρικά και αφορούν το συγκεκριμένο προς επίλυση πρόβλημα. Η πιο ενδιαφέρουσα συμβολή είναι αυτή του αλγορίθμου **Ant Colony System (ACS)** καθώς πραγματεύεται την εισαγωγή της τοπικής ενημέρωσης φερομόνης. Η ενημέρωση αυτή εκτελείται από όλα τα μυργμήγκια σε κάθε βήμα του αλγορίθμου. Το κάθε μυρμήγκι ενημερώνει μόνο την τελευταία ακμή που έχει ακολουθήσει. Ο κύριος στόχος της τοπικής ενημέρωσης είναι να διαφοροποιηθεί η αναζήτηση, η οποία πραγματοποιείται από τα μυρμήγκια που ακολουθούν κατά τη διάρκεια μιας επανάληψης. Με τη μείωση της συγκέντρωσης φερομόνης τα επόμενα μυρμήγκια ενθαρρύνονται να επιλέξουν άλλες ακμές και, ως εκ

τούτου, να παράγουν διαφορετικές λύσεις. Με τον τρόπο αυτό έχουμε πλουραλισμό λύσεων κατά τη διάρκεια μιας επανάληψης.

Άλλες γνωστές μορφές αλγορίθμου είναι οι εξής: Elitist Ant System, Rank – based Ant System (ASrank), Continuous Orthogonal Ant Colony (COAC) και Recursive Ant Colony Optimization.

2.2.2.3 Η ACO μέσα από το πρόβλημα του πλανόδιου πωλητή (TSP)

Ο ευκολότερος τρόπος για την κατανόηση του τρόπου λειτουργίας του αλγορίθμου βελτιστοποίησης αποικιών είναι μέσω ενός παραδείγματος. Ας δούμε την εφαρμογή του στο πρόβλημα του πλανώδιου πωλητή (TSP). Στο TSP μια σειρά από θέσεις (π.χ. πόλεις) και οι αποστάσεις μεταξύ τους είναι δεδομένες. Το πρόβλημα συνίσταται στην εύρεση μίας κλειστής περιοδείας ελάχιστου μήκους, που επισκέπτεται κάθε πόλη μία και μόνο μία φορά.

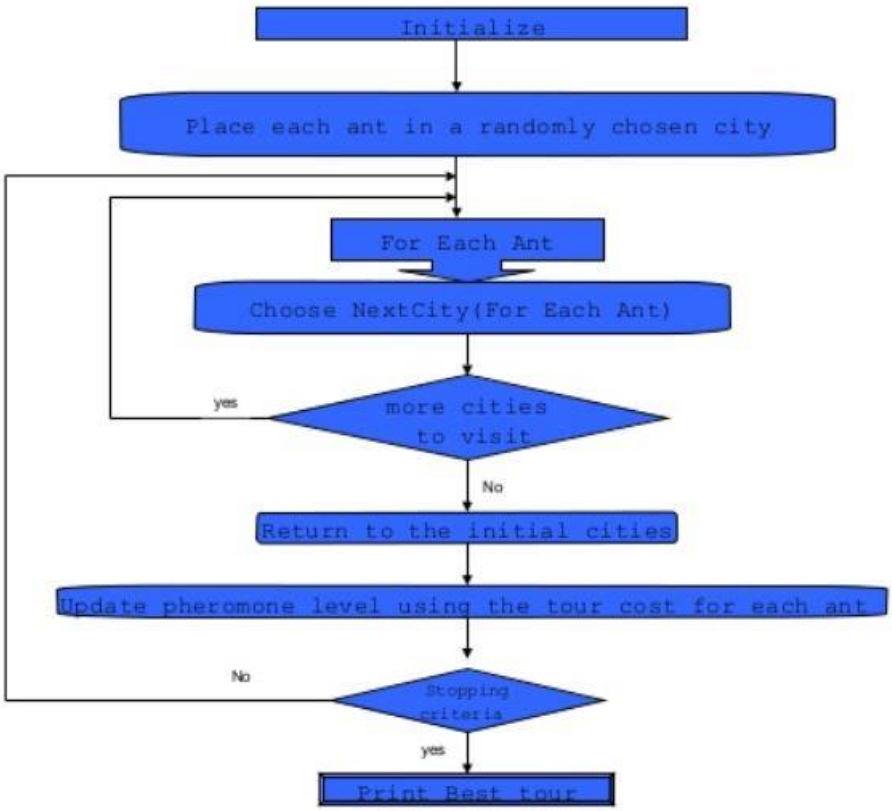
Για την εφαρμογή του ACO στο TSP θεωρούμε ότι το γράφημά μας ορίζεται από την συσχέτιση του συνόλου των πόλεων με το σύνολο των κορυφών του γραφήματος. Αυτό το γράφημα ονομάζεται γράφημα κατασκευής. Δεδομένου ότι στο TSP είναι δυνατό να κινηθεί κανείς από οποιαδήποτε δεδομένη πόλη σε οποιαδήποτε άλλη πόλη, το γράφημα κατασκευής είναι πλήρως συνδεδεμένο και ο αριθμός των κορυφών είναι ίσος με τον αριθμό των πόλεων. Ορίζουμε τα μήκη των ακμών μεταξύ των κορυφών τέτοια ώστε να είναι ανάλογα προς τις αποστάσεις μεταξύ των πόλεων που αντιπροσωπεύονται από αυτές τις κορυφές και έχουμε συνδέσει τιμές φερομόνης και ευρετικές τιμές με τις ακμές του γραφήματος. Οι τιμές φερομόνης τροποποιούνται κατά το χρόνο εκτέλεσης και αντιπροσωπεύουν την συσσωρευμένη εμπειρία της αποικίας μυρμηγκιών, ενώ οι ευρετικές τιμές είναι τιμές που εξαρτώνται από το ίδιο το πρόβλημα, και στην περίπτωση του TSP, έχουν οριστεί να είναι το αντίστροφο των μηκών των ακμών.

Τα μυρμήγκια κατασκευάζουν τις λύσεις ως εξής: κάθε μυρμήγκι ξεκινάει από μια τυχαία επιλεγμένη πόλη (κορυφή του γραφήματος κατασκευής) και σε κάθε στάδιο κατασκευής κινείται κατά μήκος των ακμών του γραφήματος. Κάθε μυρμήγκι διατηρεί τη μνήμη της διαδρομής του, και στα επόμενα βήματα επιλέγει μεταξύ των ακμών που δεν οδηγούν σε κορυφές που έχει ήδη επισκεφθεί. Ένα μυρμήγκι έχει κατασκευάσει μία λύση, όταν αυτό έχει επισκεφθεί όλες τις κορυφές του γραφήματος. Σε κάθε βήμα κατασκευής, ένα μυρμήγκι επιλέγει πιθανοτικά την ακμή που θα ακολουθήσει μεταξύ

εκείνων που οδηγούν σε κορυφές που ακόμα δεν έχει επισκεφτεί. Οι πιθανότητες του κανόνα επιλογής επηρεάζονται από τις τιμές φερομόνης και τις ευρετικές πληροφορίες: όσο υψηλότερη είναι η τιμή της φερομόνης και η ευρετική αξία που συνδέεται σε μία ακμή, τόσο μεγαλύτερη είναι η πιθανότητα ένα μυρμήγκι να επιλέξει τη συγκεκριμένη ακμή.

Αφού όλα τα μυρμήγκια έχουν ολοκληρώσει την περιοδεία τους, η φερομόνη στις ακμές ενημερώνεται. Αρχικά, κάθε μία από τις τιμές φερομόνης μειώνεται κατά ένα ορισμένο ποσοστό. Στη συνέχεια, κάθε ακμή λαμβάνει ένα ποσό πρόσθετης φερομόνης ανάλογο προς την ποιότητα των λύσεων στις οποίες ανήκει (υπάρχει μία λύση ανά μυρμήγκι). Η διαδικασία αυτή εφαρμόζεται κατ 'επανάληψη μέχρι να ικανοποιηθεί ένα κριτήριο τερματισμού.

Στο διάγραμμα ροής που ακολουθεί φαίνεται η διαδικασία που ακολουθείται κατά τη μέθοδο Ant Colony Optimization:



Εικόνα 3 : Το διάγραμμα ροής της μεθευρετικής μεθόδου Ant Colony Optimization

2.2.3 Μέθοδος Προσομοιωμένης Ανόπτησης (SA – Simulated Annealing)

Μία πολύ σημαντική μέθοδος επίλυσης προβλημάτων συνδυαστικής βελτιστοποίησης είναι και η Προσομοιωμένη Ανόπτηση. Ο όρος της ανόπτησης χρησιμοποιείται κυρίως στην θερμοδυναμική. Ανόπτηση είναι η διαδικασία θέρμανσης του μετάλλου και στη συνέχεια η σταδιακή ψύξη του με ελεγχόμενο ρυθμό ψύξης, έτσι ώστε το μέταλλο να αποκτήσει την ιδανική αναλογία των χαρακτηριστικών της σκληρότητας και ευθραστότητας. Η ανόπτηση επιδρά στην κρυσταλλική δομή του μετάλλου και αποδίδει στο μέταλλο τις επιθυμητές ιδιότητες. Στη συνδυαστική βελτιστοποίηση αυτό μπορεί να αναπαρασταθεί με την κίνηση κάποιου κριτηρίου στον εφικτό χώρο αναζήτησης ούτως ώστε όταν τελειώσει ο αλγόριθμος να έχουμε μία εφικτή λύση. Η συνεισφορά του είναι ότι αποφεύγει τον εγκλωβισμό σε τοπικά ακρότατα (κατά το δυνατόν). Ο τρόπος που το επιτυγχάνει είναι η αποδοχή και κακών λύσεων, με βάση κάποια πιθανότητα αποδοχής, p.

2.2.3.1 Ιστορική αναδρομή

Ο αλγόριθμος της Προσομοιωμένης Ανόπτησης προτάθηκε το 1983 από τους Kirkpatrick, Gelatt και Vecchi. Έχει εφαρμοστεί περισσότερο σε διακριτούς χώρους έρευνας (discrete space) και λιγότερο σε συνεχείς.

2.2.3.2 Η βασική προσέγγιση του αλγορίθμου

Η προσομοιωμένη ανόπτηση χρησιμοποιεί μία γειτονιά $N_{(s)}$ καινούργιων καταστάσεων, που μπορεί να φτάσει κάποιος από την τρέχουσα κατάσταση s. Ο αλγόριθμος λειτουργεί παίρνοντας μία αρχική κατάσταση s_0 και μία τυχαία διαταραχή στην αρχική αυτή λύση, έτσι ώστε να γεννηθεί μία καινούργια λύση s'. Η διαφορά της αντικειμενικής συνάρτησης για αυτές τις δύο καταστάσεις είναι $\delta = f(s') - f(s_0)$. Εάν, λοιπόν, στοχεύουμε στη μεγιστοποίηση της αντικειμενικής συνάρτησης και $\delta > 0$, δηλαδή η γειτονική λύση είναι καλύτερη της αρχικής, τότε η καινούργια κατάσταση είναι αποδεκτή με βεβαιότητα. Αντίθετα, αν $\delta < 0$, τότε η νέα κατάσταση γίνεται αποδεκτή με κάποια πιθανότητα. Από την άλλη, αν στοχεύουμε στην ελαχιστοποίηση της αντικειμενικής συνάρτησης και $\delta > 0$, τότε η νέα κατάσταση δεν είναι καλύτερη από την αρχική και επομένως είναι αποδεκτή όχι με βεβαιότητα αλλά με κάποια πιθανότητα ενώ αν $\delta < 0$, τότε η καινούργια κατάσταση είναι αποδεκτή με βεβαιότητα καθώς η τιμή της αντικειμενικής συνάρτησης για αυτήν την νέα κατάσταση είναι καλύτερη από την τιμή της αντικειμενικής συνάρτησης της αρχικής κατάστασης.

Η πιθανότητα αυτή με την οποία γίνεται αποδεκτή μία χειρότερη γειτονική λύση, καθώς επιδιώκεται η ελαχιστοποίηση της αντικειμενικής συνάρτησης, προέρχεται από τους νόμους της θερμοδυναμικής και είναι ίση με $p_{accept}(\delta) = e^{-\delta/t}$, όπου t η θερμοκρασία (κανονικά η πιθανότητα είναι $p_{accept}(\delta) = e^{-\delta/kt}$, με το k να αναφέρεται στη σταθερά Boltzmann, αλλά στη συνδυαστική βελτιστοποίηση δεν χρησιμοποιείται και για αυτόν το λόγο μπορεί να αναιρεθεί). Ο λόγος που μπορεί να γίνει αποδεκτή με κάποια πιθανότητα μία γειτονική λύση χειρότερη από την τρέχουσα είναι για να ξεφύγουμε από τυχόν τοπικό βέλτιστο στο οποίο έχουμε παγιδευτεί. Επιπλέον, θα πρέπει να τονιστεί ότι η πιθανότητα αποδοχής μειώνεται με το πέρασμα των επαναλήψεων, καθώς η θερμοκρασία μειώνεται σταδιακά. Αυτό σημαίνει ότι σε όσο πιο προχωρημένο στάδιο βρίσκεται η διαδικασία, τόσο πιο πολύ ευνοείται η περίπτωση να μετακινηθούμε προς την ελαχιστοποίηση της συνάρτησης παρά στην αποδοχή χειρότερης γειτονικής λύσης.

Παρακάτω παρουσιάζονται τα βήματα της μεθευρετικής μεθόδου της Προσομοιωμένης Ανόπτησης σε μορφή ψευδοκώδικα:

```
1  s ← initial solution
2  bestSolution ← s
3  t ← maxTemperature
4  while t > minTemperature do
5      iter ← 0
6      while iter < maxIterations do
7          s' ← select a random solution s' ∈ N(s)
8          Δ ← f(s') − f(s)
9          if Δ < 0 then
10             s ← s'
11             if f(s') < f(bestSolution) then
12                 bestSolution ← s'
13         else if rand(0, 1) < e^(−Δ/t) then
14             s ← s'
15         iter ← iter + 1
16     t ← t × (1 − α)
17 return bestSolution
```

Εικόνα 4 : Ο ψευδοκώδικας της Προσωμοιωμένης Ανόπτησης

Κατά τη διάρκεια της διαδικασίας της προσομοιωμένης ανόπτησης, υπάρχουν πολλοί παράγοντες που μπορούν να ρυθμιστούν και να δοκιμαστούν. Ο καθορισμός της

γειτονιάς που θα ψάξει να βρει βέλτιστο ο αλγόριθμος, η συνάρτηση μείωσης της θερμοκρασίας και ο αριθμός των εσωτερικών επαναλήψεων είναι παράγοντες που μπορούν να αλλάξουν δραστικά την αποτελεσματικότητα και την ταχύτητα του αλγορίθμου. Είναι επίσης πιθανό να υπάρχει στρατηγική αύξησης της θερμοκρασίας ή να επιτρέπεται επιπλέον αναζήτηση σε ένα τοπικό ελάχιστο.

2.2.3.3. Μέθοδος τοπικής αναζήτησης

Η μέθοδος τοπικής αναζήτησης που χρησιμοποιήθηκε στην υλοποιήση που πραγματοποιήθηκε στο πλαίσιο της εργασίας για την εύρεση γειτονικής λύσης της τρέχουσας κατά τη μέθοδο της προσομιωμένης ανόπτησης στην εφραμογή της για εύρεση υποβέλτιστης λύσης στο πρόβλημα Πλανόδιου Πωλητή είναι αυτή της αμοιβαίας μετάθεσης δύο κόμβων (swap). Η συγκεκριμένη μέθοδος εύρεσης γειτονιάς της τρέχουσας λύσης έχει παρουσιαστεί ήδη λεπτομερώς στο πλαίσιο της αντίστοιχης αναφοράς για τη μέθοδο τοπικής αναζήτησης που χρησιμοποιήθηκε στη μεθευρετική μέθοδο αναζήτησης με χρήση απαγορευμένων κινήσεων για την επίλυση του TSP.

3 Μεθοδολογία

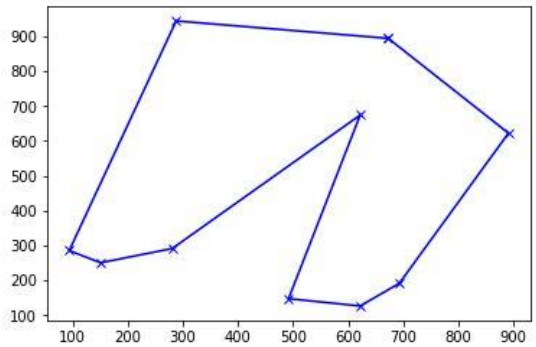
Σε αυτό το κεφάλαιο παρουσιάζονται τα αποτελέσματα από τις διάφορες εκτελέσεις των αλγορίθμων που έχουν πραγματοποιηθεί στο πλαίσιο της εργασίας. Ταυτόχρονα, επισημαίνονται και τα συμπεράσματα που εξάγονται από τις διάφορες εκτελέσεις και τα οποία με τη σειρά τους οδηγούσαν προς την πραγματοποίηση νέων εκτελέσεων με διαφοροποίηση των τιμών των παραμέτρων της κάθε μεθόδου.

3.1. Εφαρμογή του Simulated Annealing για το TSP

Στον αλγόριθμο της προσομοιωμένης ανόπτησης που υλοποιήθηκε, ζητείται από τον χρήστη η εισαγωγή του πλήθους των κόμβων του προβλήματος και δημιουργείται τυχαία ένας γράφος με τόσους κόμβους όσους επέλεξε ο χρήστης. Επιπρόσθετα, ζητείται από τον χρήστη το πλήθος των φορών που θα μειωθεί η θερμοκρασία, δηλαδή ζητείται να διαμορφώσει το πρόγραμμα μείωσης της θερμοκρασίας. Ασφαλώς, όσο πιο μεγάλος ο αριθμός των επαναλήψεων της διαδικασίας, τόσο πιο χρονοβόρα είναι και η μέθοδος.

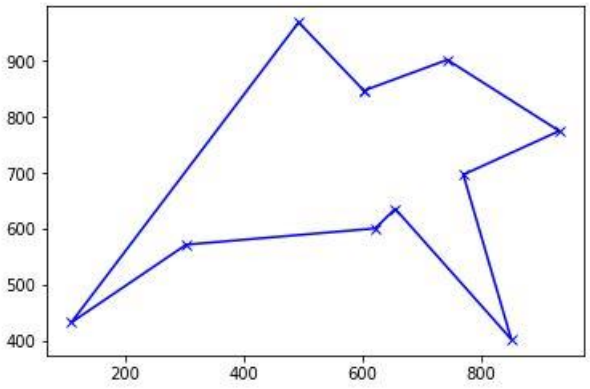
Πράγματι, από τις εκτελέσεις του αλγορίθμου που πραγματοποιήθηκαν επιβεβαιώνεται το παραπάνω συμπέρασμα. Επίσης, όπως είναι προφανές, όσο πιο μεγάλος είναι ο γράφος, δηλαδή όσο πιο μεγάλο είναι το πλήθος των κόμβων που εισάγει ο χρήστης ως δεδομένο, τόσο πιο πολύ χρόνο καταλαμβάνει η εκτέλεση του αλγορίθμου.

Αρχικά, δοκιμάσαμε εκτέλεση για ένα γράφημα 10 κόμβων και για 100 μειώσεις θερμοκρασίας. Το αποτέλεσμα της εκτέλεσης παράχθηκε μετά από 13.6096 ms και ήταν το ακόλουθο:



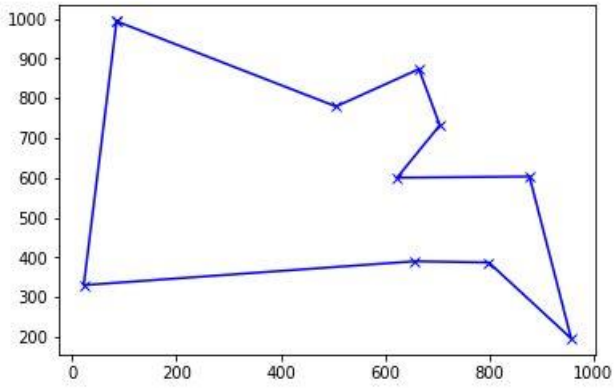
Εικόνα 5 : Το αποτέλεσμα εκτέλεσης προσομοιωμένης ανόπτησης για πρόβλημα TSP 10 κόμβων με 100 μειώσεις θερμοκρασίας

Στη συνέχεια, αποφασίσαμε τη δοκιμή για ίδιο πλήθος κόμβων αλλά με διαφορετικό πρόγραμμα μείωσης θερμοκρασίας, καθώς εισάγαμε ως πλήθος μειώσεων τις 1000. Το αποτέλεσμα που παράχθηκε μετά από 60.754716 ms ήταν το ακόλουθο :



Εικόνα 6 : Το αποτέλεσμα εκτέλεσης προσομοιωμένης ανόπτησης για πρόβλημα TSP 10 κόμβων με 1000 μειώσεις θερμοκρασίας

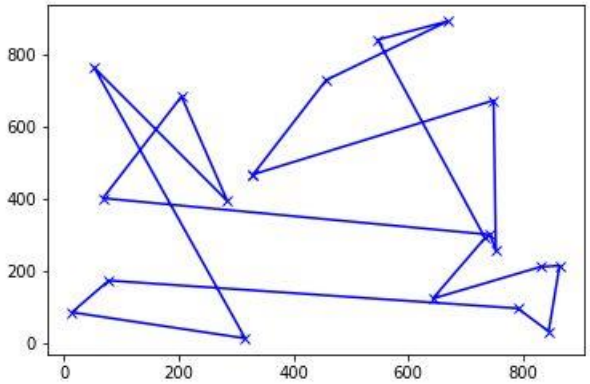
Όπως περιμέναμε, ο χρόνος εκτέλεσης ήταν αρκετά μεγαλύτερος καθώς αυξήθηκε περίπου 4,5 φορές. Μάλιστα, αν αυξήσουμε κατά ακόμα μία τάξη μεγέθους τον αριθμό των επαναλήψεων, εισάγοντας 10000 μειώσεις, τότε εξάγεται μετά από 585.721380 ms το παρακάτω αποτέλεσμα :



Εικόνα 7 : Το αποτέλεσμα εκτέλεσης προσομοιωμένης ανόπτησης για πρόβλημα TSP 10 κόμβων με 10000 μειώσεις θερμοκρασίας

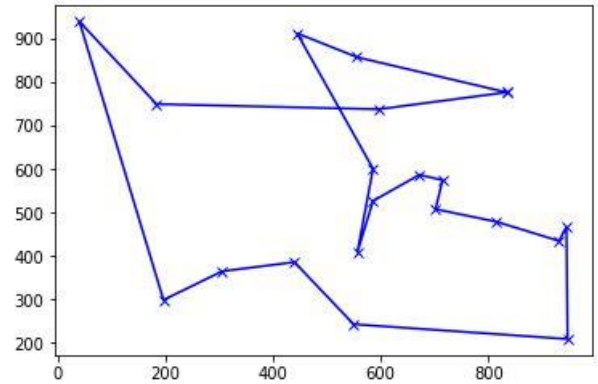
Παρατηρούμε ότι ο χρόνος εκτέλεσης εκτοξεύεται κατά 9,64 φορές.

Στη συνέχεια, τρέχουμε τη μέθοδο της προσομοιωμένης ανόπτησης για τον διπλάσιο αριθμό κόμβων με την ίδια ποικιλία επιλογών για το πλήθος των μειώσεων θερμοκρασίας. Αρχικά, στην περίπτωση 20 κόμβων και 100 επαναλήψεων, λαμβάνουμε το ακόλουθο γράφημα μετά από 9.048411 ms :



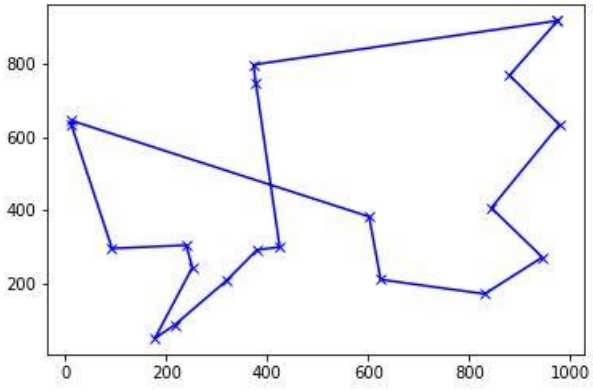
Εικόνα 8 : Το αποτέλεσμα εκτέλεσης προσομοιωμένης ανόπτησης για πρόβλημα TSP 20 κόμβων με 100 μειώσεις θερμοκρασίας

Προσδοκούμε ότι με την αύξηση των μειώσεων στις 1000 θα πάρουμε ένα καλύτερο αποτέλεσμα. Μετά από 62.545687 ms παράγεται το ακόλουθο γράφημα :



Εικόνα 9 : Το αποτέλεσμα εκτέλεσης προσομοιωμένης ανόπτησης για πρόβλημα TSP 20 κόμβων με 1000 μειώσεις θερμοκρασίας

Πράγματι, φαίνεται ότι με την αύξηση του προγράμματος μειώσης θερμοκρασίας παράγεται ένα σαφώς καλύτερο αποτέλεσμα ενώ και πάλι φαίνεται η αύξηση στο χρόνο εκτέλεσης. Τέλος, εκτελούμε για 20 κόμβους και 10000 επαναλήψεις, λαμβάνοντας ως έξοδο μετά από 587.313411 ms :



Εικόνα 10 : Το αποτέλεσμα εκτέλεσης προσομοιωμένης ανόπτησης για πρόβλημα TSP 20 κόμβων με 10000 μειώσεις θερμοκρασίας

Στον επόμενο πίνακα παρουσιάζονται συνοπτικά οι παράμετροι εισόδου και οι χρόνοι εκτέλεσης σε κάθε μία από τις παραπάνω εκτελέσεις:

Πλήθος Κόμβων	Πλήθος μειώσεων θερμοκρασίας	Συνολικός χρόνος εκτέλεσης
10	100	13.6096 ms
10	1000	60.754716 ms
10	10000	585.721380 ms
20	100	9.048411 ms
20	1000	62.545687 ms
20	10000	587.313411 ms

Πίνακας 1 : Συνοπτικός πίνακας εκτελέσεων Προσομοιωμένης Ανόπτησης

3.2 Εφαρμογή του TS για το Knapsack Problem

Αυτή η ενότητα είναι αφιερωμένη στην παρουσίαση των αποτελεσμάτων των εκτελέσεων του αλγορίθμου Tabu Search για την εύρεση λύσης σε ένα από τα πιο ευρέως διαδεδομένα προβλήματα συνδυαστικής βελτιστοποίησης, το πρόβλημα του σακιδίου. Όπως έχει ήδη αναφερθεί, η υλοποίηση που πραγματοποιήθηκε αφορά στο 0-1 διδιάστατο πρόβλημα σακιδίου, στο οποίο για το κάθε αντικείμενο είναι γνωστά η αξία του, το βάρος του και ο όγκος του. Επίσης, είναι γνωστά, το συνολικό βάρος και ο συνολικός όγκος που μπορεί να χωρέσει στο σακίδιο.

Ως παράμετροι εισόδου ζητούνται από τον χρήστη, εκτός φυσικά του αρχείου κειμένου που περιλαμβάνει όλα τα δεδομένα του προβλήματος, το μέγεθος της λίστας των απαγορευμένων κινήσεων καθώς και το μέγιστο πλήθος των ανεπιτυχών επαναλήψεων του αλγορίθμου, μετά από το οποίο ο αλγόριθμος τερματίζεται.

Για την κατασκευή αρχικής λύσης, χρησιμοποιήθηκε η κατασκευαστική ευρετική μέθοδος (constructive heuristic) του άπληστου αλγορίθμου κατά την οποία τα αντικειμένα ταξινομούνται με κριτήριο το λόγο $\frac{p}{w+v}$ και επιλέγονται τα k πρώτα αντικείμενα με τη μεγαλύτερη τιμή αυτού του λόγου που ικανοποιούν τους δύο περιορισμούς βάρους, W, και όγκου, V, του σακιδίου. Αυτό σημαίνει ότι δεν είναι δυνατή η προσθήκη του υπ'αριθμόν $(k+1)$ αντικείμενου επειδή δεν θα ικανοποιείται τουλάχιστον ένας εκ των περιορισμών.

Η τελική λύση είναι ένα διάνυσμα μη σταθερού μήκους το οποίο περιέχει τον άυξοντα αριθμό του κάθε αντικειμένου που εισάγεται στο σακίδιο. Με τον όρο «μη σταθερού μήκους» εννοούμε ότι μπορεί για το ίδιο ακριβώς πρόβλημα και την ίδια ακριβώς είσοδο άλλοτε να προκύψει διάνυσμα μήκους n και άλλοτε μήκους $m \neq n$, δηλαδή στην πρώτη περίπτωση n αντικείμενα να συμμετέχουν στη λύση ενώ στη δεύτερη περίπτωση m αντικείμενα να αποτελούν τη λύση.

Οι δοκιμές πραγματοποίηθηκαν σε τρία διαφορετικά σύνολα δεδομένων (data sets), ούτως ώστε να εξαχθούν όσο το δυνατόν πιο ασφαλή συμπεράσματα. Τα τρία διαφορετικά σύνολα δεδομένων είναι διαθέσιμα στο παράρτημα Α.

Από τις εκτελέσεις του αλγορίθμου παρατηρούμε ότι ο αλγορίθμος βελτιώνει σε αρκετά ικανοποιητικό βαθμό την αρχική λύση ενώ στις περισσότερες περιπτώσεις καταναλώνονται και οι περισσότεροι από τους πόρους σε ό,τι αφορά τις δύο διαστάσεις του προβλημάτος, δηλαδή το βάρος και τον όγκο.

3.2.1. Αποτελέσματα εκτελέσεων για το πρώτο σύνολο δεδομένων

Αναλυτικότερα, σε ό,τι αφορά το πρώτο σύνολο δεδομένων, η αρχική λύση που εξήγαγε η κατασκευαστική ευρετική μέθοδος που παρουσιάστηκε παραπάνω ήταν η [13, 7, 9, 20, 2, 4, 23, 25, 22, 26, 6, 21, 14, 12] με συνολική αξία 139157. Στους δύο ακόλουθους πίνακες αποτυπώνονται οι παραμέτροι εισόδου και τα αποτελέσματα του αλγορίθμου για τις δοκιμές που πραγματοποιήθηκαν :

Α.Α.	Μέγιστο Πλήθος Ανεπιτυχών Επαναλήψεων	Μήκος Λίστας Απαγορευμένων Κινήσεων
1	20	20
2	100	20
3	500	20
4	20	100
5	100	100
6	500	100
7	20	500
8	100	500
9	500	500

Πίνακας 2 : Οι είσοδοι των εκτελέσεων Tabu Search για Knapsack Problem για το σύνολο δεδομένων 1

Α.Α.	Συνολική Αξία Τελικής Λύσης	Βελτίωση Αξίας	Τελική Λύση	Χρόνος Εκτέλεσης (ms)
1	142903	3746	[25, 11, 4, 20, 26, 12, 6, 9, 7, 22, 2, 13, 23]	52.130402
2	**145738**	**6581**	**[2, 25, 26, 13, 23, 20, 11, 4, 9, 6, 22, 7]**	**357.746359**
3	142903	3746	[13, 12, 9, 23, 22, 7, 4, 26, 6,	1411.382430

			11, 2, 25, 20]	
4	142883	3726	[9, 7, 25, 4, 13, 22, 26, 16, 23, 20, 11, 2, 6]	52.836714
5	142903	3746	[13, 6, 9, 20, 2, 4, 11, 25, 22, 26, 7, 23, 12]	562.774442
6	142903	3746	[6, 2, 22, 4, 20, 25, 11, 7, 9, 13, 23, 12, 26]	2824.089067
7	142393	3236	[26, 22, 20, 7, 11, 6, 4, 2, 23, 13, 25, 9]	91.154950
8	142903	3746	[6, 13, 12, 9, 25, 7, 23, 20, 26, 2, 22, 4, 11]	611.651384
9	143058	3901	[7, 20, 2, 9, 13, 11, 22, 25, 10, 4, 23]	6490.160457

Πίνακας 3 : Τα αποτελέσματα των εκτελέσεων Tabu Search για Knapsack Problem για το σύνολο δεδομένων 1

Από τους παραπάνω πίνακες, φαίνεται ότι η καλύτερη λύση παράχθηκε κατά την εκτέλεση του δεύτερου σεναρίου, στο οποίο είχαν οριστεί ως παράμετροι εισόδου οι 100 ανεπιτυχείς επαναλήψεις ως μέγιστο πλήθος και το μήκος της λίστας tabu ήταν 20 απαγορευμένες κινήσεις. Η συνολική αξία της τελικής λύσης ήταν 145738, οπότε επήλθε βελτίωση συγκριτικά με την αρχική λύση κατά $145738 - 139157 = 6581$ μονάδες. Η τελική λύση αποτελείται από τα ακόλουθα αντικείμενα : [2, 25, 26, 13, 23, 20, 11, 4, 9, 6, 22, 7].

3.2.2. Αποτελέσματα εκτέλεσεων για το δεύτερο σύνολο δεδομένων

Σε ό,τι αφορά το δεύτερο σύνολο δεδομένων, η αρχική λύση που εξήγαγε η κατασκευαστική ευρετική μέθοδος που παρουσιάστηκε παραπάνω ήταν η [13, 7, 9, 23, 20, 2, 26, 4, 25, 21, 18] με συνολική αξία 121880. Στους δύο ακόλουθους πίνακες αποτυπώνονται οι παραμέτροι εισόδου και τα αποτελέσματα του αλγορίθμου για τις δοκιμές που πραγματοποιήθηκαν :

Α.Α.	Μέγιστο Πλήθος Ανεπιτυχών Επαναλήψεων	Μήκος Λίστας Απαγορευμένων Κινήσεων
1	20	20
2	50	20
3	100	20
4	20	50
5	50	50
6	100	50
7	20	100
8	50	100
9	100	100

Πίνακας 4 : Οι είσοδοι των εκτελέσεων Tabu Search για Knapsack Problem για το σύνολο δεδομένων 2

Α.Α.	Συνολική Αξία Τελικής Λύσης	Βελτίωση Αξίας	Τελική Λύση	Χρόνος Εκτέλεσης (ms)
1	133948	12068	[6, 22, 18, 4, 20, 2, 13, 9, 7, 23]	81.086138
2	134838	12958	[9, 16, 6, 22, 7, 20, 13, 4, 2, 12, 23]	173.082083
3	**134998**	**13118**	**[23, 20, 13, 22, 4, 10, 9, 7, 2, 6]**	**518.162106**
4	**134998**	**13118**	**[10, 23, 22, 9, 2,**	**102.785977**

			4, 7, 13, 6, 20]	
5	133838	11958	[7, 22, 9, 20, 23, 4, 6, 2, 13]	222.218624
6	134838	12958	[13, 23, 6, 4, 7, 22, 16, 9, 20, 2, 12]	399.034295
7	131563	9683	[23, 2, 26, 13, 4, 7, 9, 22, 20]	70.610195
8	133948	12068	[23, 9, 2, 22, 20, 6, 4, 13, 7, 18]	171.752221
9	**134998**	**13118**	**[7, 13, 23, 20, 4, 9, 22, 2, 6, 10]**	**350.966476**

Πίνακας 5 : Τα αποτελέσματα των εκτελέσεων Tabu Search για Knapsack Problem για το σύνολο δεδομένων 2

Από τους παραπάνω πίνακες, φαίνεται ότι η καλύτερη λύση παράχθηκε σε τρεις διαφορετικές περιπτώσεις : κατά την εκτέλεση του τρίτου, του τέταρτου και του τελευταίου σεναρίου. Η συνολική αξία της τελικής λύσης ήταν 134998, οπότε επήλθε βελτίωση συγκριτικά με την αρχική λύση κατά $134998 - 121880 = 13118$ μονάδες. Η τελική λύση αποτελείται από τα ακόλουθα αντικείμενα : [23, 20, 13, 22, 4, 10, 9, 7, 2, 6]. Μπορεί με μια πρώτη ματιά ο αναγνώστης να νομίσει ότι τρεις διαφορετικές λύσεις παρήγαγαν το ίδιο αποτέλεσμα σε ό,τι αφορά την αξία αλλά με μία πιο προσεκτική ματιά παρατηρούμε ότι οι λύσεις αποτελούνται από τα ακριβώς ίδια 10 αντικείμενα, που κατά αριθμητική σειρά είναι : [2, 4, 6, 7, 9, 10, 13, 20, 22, 23]. Η διαφορετική σειρά εμφάνισης των αντικειμένων στο διάνυσμα της τελικής λύσης οφείλεται στη σειρά με την οποία έγιναν οι μεταβάσεις από τη μία λύση στην επόμενη γειτονική λύση κατά την εκτέλεση του καθενός από τα σενάρια. Επομένως, μία βελτίωση του προγράμματος που δεν αλλοιώνει όμως τη μέθοδο θα μπορούσε να είναι η εμφάνιση της τελικής λύσης να γίνεται αφού τα αντικείμενα που αποτελούν τη λύση να είναι πρώτα ταξινομημένα κατά αριθμητική σειρά με βάση τον αύξοντα αριθμό του κάθε αντικειμένου.

3.2.3. Αποτελέσματα εκτελέσεων για το τρίτο σύνολο δεδομένων

Τέλος, για το τρίτο και τελευταίο σύνολο δεδομένων, η αρχική λύση που εξήγαγε η κατασκευαστική ευρετική μέθοδος ήταν η [13, 14, 16, 18, 20, 15, 21, 23, 1, 24, 28, 29, 31, 33, 0, 34, 37, 39, 40, 41, 30, 22, 17, 43, 5, 4, 7, 3, 51, 12] με συνολική αξία 567082. Στους δύο ακόλουθους πίνακες αποτυπώνονται οι παραμέτροι εισόδου και τα αποτελέσματα του αλγορίθμου για τις δοκιμές που πραγματοποιήθηκαν :

Α.Α.	Μέγιστο Πλήθος Ανεπιτυχών Επαναλήψεων	Μήκος Λίστας Απαγορευμένων Κινήσεων
1	50	50
2	100	50
3	500	50
4	50	100
5	100	100
6	500	100
7	50	500
8	100	500
9	500	500

Πίνακας 6 : Οι είσοδοι των εκτελέσεων Tabu Search για Knapsack Problem για το σύνολο δεδομένων 3

Α.Α.	Συνολική Αξία Τελικής Λύσης	Βελτίωση Αξίας	Τελική Λύση	Χρόνος Εκτέλεσης (ms)
1	584725	17643	[5, 4, 13, 30, 15, 39, 18, 14, 33, 28, 21, 23, 25, 29, 1, 37, 20, 22, 41, 24, 16, 3, 0]	3076.945419
2	**586945**	**19863**	**[25, 22, 14, 13, 18, 15, 0, 1, 16, 20, 28, 33, 24, 39, 29, 21, 4, 31,**	**9135.729245**

			30, 3, 5, 37, 23]	
3	**586945**	**19863**	**[23, 4, 21, 39, 22, 33, 1, 30, 37, 29, 31, 3, 15, 5, 28, 25, 13, 20, 24, 14, 18, 0, 16]**	**18250.297381**
4	580858	13776	[1, 34, 37, 16, 4, 21, 3, 15, 23, 30, 14, 13, 22, 18, 31, 20, 24, 25, 29, 40, 0, 41, 33, 28]	1701.784905
5	584725	17643	[25, 30, 22, 23, 29, 13, 28, 37, 5, 14, 18, 16, 21, 33, 15, 39, 4, 1, 0, 20, 41, 24, 3]	7431.291812
6	**586945**	**19863**	**[13, 23, 0, 24, 1, 3, 4, 29, 37, 25, 22, 28, 5, 30, 18, 16, 14, 20, 33, 39, 15, 21, 31]**	**72833.898281**
7	582021	14939	[41, 21, 7, 14, 3, 39, 25, 18, 0, 5, 15, 30, 16, 1, 22, 24, 31, 23, 33, 20, 13, 37, 29]	2727.797158
8	582616	15534	[34, 39, 17, 33, 20, 3, 0, 29, 31, 14, 30, 23, 18, 1, 5, 41, 16, 15, 37, 13, 22, 21, 25, 24]	3385.709968
9	**586945**	**19863**	**[39, 13, 31, 15, 29, 4, 20, 25, 37, 1, 14, 22, 23, 33, 21, 30, 28, 24, 5, 0, 3, 16, 18]**	**46556.020262**

Πίνακας 7 : Τα αποτελέσματα των εκτελέσεων Tabu Search για Knapsack Problem για το σύνολο δεδομένων 3

Από τους παραπάνω πίνακες, φαίνεται ότι η καλύτερη λύση παράχθηκε κατά την εκτέλεση τεσσάρων διαφορετικών σεναρίων : του δεύτερου, του τρίτου, του έκτου και του ένατου. Κοινό σημείο σε τρία από τα τέσσερα αυτά σενάρια είναι ότι το μέγιστο πλήθος ανεπιτυχών επαναλήψεων είναι τα 500. Αυτό επαληθεύει την υποψία ότι όσο μεγαλύτερο οριστεί αυτό το μέγεθος, τόσο πιο καλό θα είναι το αποτέλεσμα το αποτέλεσμα της τελικής λύσης καθώς το όριο τερματισμού του αλγορίθμου επεκτείνεται με αποτέλεσμα η μέθοδος να έχει περισσότερες ευκαιρίες να ερευνήσει το συνολικό χώρο των πιθανών λύσεων. Ασφαλώς, το αντίβαρο είναι ότι με την αύξηση αυτού του μεγέθους αυξάνεται και ο χρόνος εκτέλεσης του αλγορίθμου καθώς όπως είναι φυσικό χρειάζεται περισσότερο χρόνο για να φτάσει στην ικανοποίηση του κριτηρίου τερματισμού. Η συνολική αξία της τελικής λύσης ήταν 586945, οπότε επήλθε βελτίωση συγκριτικά με την αρχική λύση κατά $586945 - 567082 = 19863$ μονάδες. Η τελική λύση αποτελείται από τα ακόλουθα 23 αντικείμενα, εμφανιζόμενα κατά αύξουσα σειρά, σύμφωνα με το ID τους, δηλαδή τον αύξοντα αριθμό (Α.Α) τους : [0, 1, 3, 4, 5, 13, 14, 15, 16, 18, 20, 21, 22, 23, 24, 25, 28, 29, 30, 31, 33, 37, 39]. Όπως θα συμπεράνει ο παρατηρητικός αναγνώστης, στην τελική λύση συμπεριλαμβάνονται κάποια από τα πιο συμφέροντα αντικείμενα, ως προς το λόγο $\frac{αξία}{βάρος+όγκος}$, όπως ήταν αναμενόμενο.

3.3. Εφαρμογή του TS για το TSP

Στην ενότητα που ακολουθεί, παρουσιάζονται τα αποτελέσματα των εκτέλεσεων της μεθευρετικής μεθόδου Tabu Search, όπως υλοποιήθηκε σε Python για την επίλυση του προβλήματος του Πλανόδιου Πωλητή (TSP). Ως είσοδος, δίνεται ένα αρχείο κειμένου στο οποίο περιλαμβάνονται οι αποστάσεις κάθε ζεύγους κόμβων ενός συμμετρικού γράφου. Το παράδειγμα που χρησιμοποιήσαμε ως πρόβλημα TSP είναι το πέρασμα από 16 πόλεις που αποτελούν πρωτεύουσες κρατών της Ευρωπαϊκής Ένωσης, ορίζοντας ως εκκίνηση την πρωτεύουσα της Ελλάδας, Αθήνα. Αυτό σημαίνει ότι το αρχείο κειμένου θα πρέπει να περιλαμβάνει από μία φορά (καθώς πρόκειται για συμμετρικό TSP) τις αποστάσεις μεταξύ κάθε ζεύγους πόλεων που μπορεί να σχηματιστεί από αυτό το σύνολο των 16 πρωτευουσών. Άρα, περιλαμβάνονται $15 + 14 + 13 + \ldots + 1 = \sum_{i=1}^{15} i = 120$ γραμμές, οι οποίες έχουν τη μορφή :

$$A\ B\ dist_{AB}$$

όπου Α, Β πόλεις που ανήκουν στο σύνολο των κόμβων και $dist_{AB}$ ένας ακέραιος αριθμός που αντιστοιχεί στην απόσταση από την πόλη Α στην πόλη Β (άρα και από τη Β στην Α), εκφρασμένη σε μέτρα.

Ακολουθεί η λίστα των πόλεων που αποτελούν τους κόμβους του γραφήματος :

Α.Α.	**Γράμμα**	**Πόλη**	**Χώρα**
0	A	Αθήνα	Ελλάδα
1	B	Λονδίνο	Ηνωμένο Βασίλειο
2	C	Βερολίνο	Γερμανία
3	D	Μαδρίτη	Ισπανία
4	E	Ρώμη	Ιταλία
5	F	Παρίσι	Γαλλία
6	G	Βουκουρέστι	Ρουμανία
7	H	Βιέννη	Αυστρία
8	I	Βαρσοβία	Πολωνία
9	J	Βουδαπέστη	Ουγγαρία
10	K	Πράγα	Τσεχία
11	L	Σόφια	Βουλγαρία
12	M	Βρυξέλλες	Βέλγιο
13	N	Στοκχόλμη	Σουηδία

14	O	Άμστερνταμ	Ολλανδία
15	P	Ζάγκρεμπ	Κροατία

Πίνακας 8 : Οι 16 ευρωπαϊκές πόλεις που αποτελούν τους κόμβους του TSP

Στην υλοποίηση που πραγματοποιήθηκε, πέρα από τις αποστάσεις μεταξύ κάθε ζεύγους πόλεων, δίνονται ως παράμετροι εισόδου άλλα δύο μεγέθη:

1. Ο αριθμός των επαναλήψεων εκτέλεσης του αλγορίθμου.
2. Το μήκος της λίστας tabu, δηλαδή της δομής μνήμης στην οποία αποθηκεύονται απαγορευμένες κινήσεις. Ως απαγορευμένες κινήσεις ορίζονται οι κινήσεις (από μία λύση στη γειτονική της) που έχουν ήδη πραγματοποιηθεί σε προηγούμενη επανάληψη και επομένως δεν μπορούν να επαναπραγματοποιηθούν.

Αρχικά, δοκιμάζοντας εκτέλεση με μηδενικές παραμέτρους εισόδου, δηλαδή μηδενικό αριθμό επαναλήψεων και ουσιαστικά χωρίς μνήμη, παίρνουμε ως αποτέλεσμα τη λύση που προέκυψε από την κατασκευαστική ευρετική μέθοδο που χρησιμοποιήθηκε για να βρεθεί το σημείο εκκίνησης της μεθευρετικής μεθόδου. Στη συγκεκριμένη περίπτωση, ως κατασκευαστική μέθοδος αρχικής λύσης υλοποιήθηκε η μέθοδος του κοντινότερου γείτονα (nearest-neighbour). Το αποτέλεσμα που παράγεται με αυτές τις παραμέτρους εισόδου είναι [0, 11, 6, 9, 7, 10, 2, 8, 15, 4, 5, 12, 14, 1, 3, 13] με τιμή της αντικειμενικής συνάρτησης 16204610. Η αντικειμενική συνάρτηση είναι το συνολικό μήκος (σε μέτρα) της διαδρομής που σχηματίζεται από τη διαδοχική επίσκεψη των κόμβων της λύσης με τη σειρά που αυτοί εμφανίζονται. Βλέπουμε λοιπόν ότι η λύση που παρήγαγε η κατασκευαστική ευρετική μέθοδος του κοντινότερου γείτονα είναι $16204610\ μέτρα$ ή $16.204{,}61\ km$.

Στη συνέχεια, εκτελέστηκαν διάφορα σενάρια ώστε να επαληθεύσουμε ότι με την αύξηση των επαναλήψεων και τη χρήση δομής μνήμης που αποθηκεύει τις κινήσεις που γίνονται σε κάθε επανάληψη παράγεται καλύτερη λύση. Αρχικά, δοκιμάζουμε με 3 επαναλήψεις και μήκος λίστας tabu ίσο με 1. Το αποτέλεσμα που παράγεται είναι [0, 11, 6, 9, 7, 10, 15, 8, 2, 13, 14, 12, 5, 1, 3, 4] με συνολική απόσταση μονοπατιού $13407340\ μέτρα$. Βλέπουμε ότι ήδη με 3 μόλις επαναλήψεις παράχθηκε πολύ καλύτερο αποτέλεσμα καθώς η συνολική βελτίωση της αντικειμενικής συνάρτησης ήταν $16204610 - 13407340 = 2797270 μέτρα$ ή $2.797{,}27\ km$. Εκτελώντας στη συνέχεια διάφορα σενάρια αυξάνοντας κάθε φορά την τιμή τουλάχιστον μίας εκ των δύο παραμέτρων εισόδου, βλέπουμε ότι η καλύτερη λύση που παράγεται είναι η [0, 11, 6, 9,

15, 7, 10, 8, 2, 13, 14, 12, 1, 5, 3, 4] με συνολικό μήκος απόστασης $12416240\ μέτρα$, οπότε η συνολική βελτίωση από την αρχική λύση είναι $3788370\ μέτρα$ ή $2.788{,}37km$.

Στον πίνακα που ακολουθεί φαίνονται για το κάθε σενάριο ο αύξων αριθμός του σεναρίου, το πλήθος των επαναλήψεων, το επιλεγμένο μήκος της λίστας Tabu, το τελικό διάνυσμα που παράχθηκε ως λύση, η συνολική απόσταση του μονοπατιού της λύσης, η βελτίωση της αντικειμενικής συνάρτησης, δηλαδή η διαφορά της απόστασης της τελικής από την απόσταση της αρχικής λύσης που παρήγαγε η ευρετική κατασκευαστική μέθοδος nearest-neighbour, και η συνολική διάρκεια εκτέλεσης του προγράμματος.

Α.Α.	Πλήθος Επαναλήψεων	Μήκος Λίστας Tabu	Παραχθείσα Λύση	Συνολική Απόσταση (m)	Βελτίωση (m)	Χρόνος Εκτέλεσης (ms)
1	0	0	[0, 11, 6, 9, 7, 10, 2, 8, 15, 4, 5, 12, 14, 1, 3, 13]	16204610	0	1.756015
2	3	1	[0, 11, 6, 9, 7, 10, 15, 8, 2, 13, 14, 12, 5, 1, 3, 4]	13407340	2797270	24.956382
3	5	2	[0, 11, 6, 9, 7, 15, 10, 8, 2, 13, 14, 12, 1, 5, 3, 4]	12681630	3522980	52.636747
4	10	4	[0, 11, 6, 9, 15, 7, 10, 8, 2, 13, 14, 12, 1, 5, 3, 4]	12416240	3788370	79.661176
5	20	5	[0, 11, 6, 9, 15, 7, 10, 8, 2, 13, 14, 12, 1, 5, 3, 4]	12416240	3788370	280.155800

6	50	5	[0, 11, 6, 9, 15, 7, 10, 8, 2, 13, 14, 12, 1, 5, 3, 4]	12416240	3788370	726.393271
7	50	10	[0, 11, 6, 9, 15, 7, 10, 8, 2, 13, 14, 12, 1, 5, 3, 4]	12416240	3788370	1161.262398

Πίνακας 9 : Συνοπτικός πίνακας παρουσίασης των παραμέτρων εισόδου και των αποτελεσμάτων εκτέλεσης των σεναρίων για την επίλυση του TSP με Tabu Search

Παρατηρούμε από τον παραπάνω πίνακα ότι όπως ήταν αναμενόμενο, η αύξηση του αριθμού επαναλήψεων του Tabu Search αυξάνει τον χρόνο εκτέλεσης του προγράμματος. Το ίδιο συμβαίνει και με την αύξηση του μήκους της λίστας στην οποία αποθηκεύονται οι κινήσεις που έχουν ήδη γίνει και επομένως είναι απαγορευμένες για τις επόμενες επαναλήψεις.

Ένα ιδιαίτερο θέμα συζήτησης σχετικά με το μήκος της λίστας tabu είναι η συμεριφορά του αλγορίθμου όταν μία κίνηση πρέπει να καταχωρηθεί ως απαγορευμένη αλλά η λίστα έχει φτάσει στο μέγιστο μήκος της. Στον αλγόριθμο που υλοποιήθηκε επιλέχθηκε η στρατηγική της εισαγωγής αυτής της νέας κίνησης στη λίστα των απαγορευμένων κινήσεων αφού πρώτα αφαιρεθεί η παλαιότερη από τις ήδη αποθηκευμένες απαγορευμένες κινήσεις. Αυτό σημαίνει ότι οι πιο πρόσφατα αποθηκευμένες κινήσεις στη λίστα των απαγορευμένων κινήσεων θα αργήσουν να διαγραφούν από αυτή ενώ οι παλαιότερες απαγορευμένες κινήσεις με το πέρασμα των επαναλήψεων απαλείφονται και έτσι παύουν να είναι απαγορευμένες.

3.4. Εφαρμογή του ACO για το TSP

Παρακάτω ακολουθούν τα συμπεράσματα από την εφαρμογή του Ant Colony Optimization (ACO) στην προσπάθεια εύρεσης βέλτιστης λύσης για το πρόβλημα TSP. Ειδικότερα, ως είσοδος έχει δοθεί ένας διδιάστατος πίνακας αποστάσεων μεταξύ 16 πρωτευουσών πόλεων της Ευρωπαϊκής Ένωσης. Αναζητείται η βέλτιστη λύση με αρχικό κόμβο την πρωτεύουσα της Ελλάδας, Αθήνα και την επιστροφή σε αυτήν αφού πρώτα πραγματοποιηθεί μία και μοναδική επίσκεψη από κάθε μία από τις υπόλοιπες 15 πρωτεύουσες. Ο πίνακας των πόλεων που περιλήφθηκαν στο σύνολο των κόμβων έχει ήδη αποτυπωθεί στην προηγούμενη ενότητα (Πίνακας 1).

Στην υλοποίηση που πραγματοποιήθηκε, πέρα από τον διδιάστατο πίνακα αποστάσεων, δίνονται ως παράμετροι άλλα έξι μεγέθη:

1. Ο αριθμός των επαναλήψεων
2. Ο αριθμός των συνολικών μυρμηγκιών
3. Ο αριθμός των καλύτερων μυρμηγκιών
4. Το ποσοστό εξάτμισης της φερομόνης
5. Ο συντελεστής βαρύτητας της φερομόνης, alpha
6. Ο συντελεστής βαρύτητας της απόστασης, beta

Στον παρακάτω πίνακα εμφανίζονται τα αποτελέσματα των εκτελέσεων του αλγορίθμου για διάφορες τιμές των παραπάνω παραμέτρων με δεδομένο ότι έχουμε 1 μυρμήγκι.

Total Ants	**Best Ants**	**Iterations**	**Evaporation Rate**	**Alpha**	**Beta**	**Result (metres)**
1	1	1000	0.9	1	2	**14206020**
1	1	1000	0.9	2	1	15803860
1	1	1000	0.9	1	1	16741800
1	1	1000	0.8	1	2	**14152440**
1	1	1000	0.8	2	1	15366560
1	1	1000	0.8	1	1	15443950
1	1	500	0.9	1	2	**13923940**
1	1	500	0.9	2	1	16885070
1	1	500	0.9	1	1	15980380
1	1	500	0.8	1	2	**14267350**

1	1	500	0.8	2	1	15668200
1	1	500	0.8	1	1	15128290

Πίνακας 10 : Αποτελέσματα εκτέλεσης ACO με ένα μυρμήγκι

Από τα παραπάνω αποτελέσματα συμπεραίνουμε ότι όταν ο συντελεστής βαρύτητας της απόστασης είναι μεγαλύτερος από το συντελεστή βαρύτητας της φερομόνης, παράγονται καλύτερα αποτελέσματα. Αυτό το συμπέρασμα μας οδήγησε σε νέες εκτελέσεις με ακόμη μεγαλύτερο συντελστή βαρύτητας beta, με τα ακόλουθα αποτελέσματα:

Total Ants	**Best Ants**	**Iterations**	**Evaporation Rate**	**Alpha**	**Beta**	**Result (metres)**
1	1	1000	0.9	1	3	**12923060**
1	1	1000	0.8	1	3	13557050
1	1	500	0.9	1	3	14269380
1	1	500	0.8	1	3	13947580
1	1	1000	0.9	1	4	**12908980**
1	1	1000	0.8	1	4	13169500
1	1	500	0.9	1	4	13634020
1	1	500	0.8	1	4	13525840

Πίνακας 11 : Μεγαλύτερο beta εξάγει καλύτερο αποτέλεσμα

Από τα παραπάνω αποτελέσματα, επαληθεύεται επίσης το συμπέρασμα ότι όσο περισσότερες είναι οι επαναλήψεις του αλγορίθμου, τόσο πιο καλό είναι το αποτέλεσμα, καθώς οι 1000 επαναλήψεις επιφέρουν κατά κανόνα λύση που είναι πιο κοντά στη βέλτιστη σε σχέση με τις 500, καθώς υπενθυμίζεται ότι ο στόχος είναι η ελαχιστοποίηση της συνολικής απόστασης.

Στη συνέχεια, πραγματοποιήθηκαν κάποιες εκτελέσεις με ακριβώς τις ίδιες παραμέτρους ώστε να δούμε πόσο ασφαλή είναι τα μέχρι τώρα συμπεράσματα. Τα αποτελέσματα φαίνονται παρακάτω:

Total Ants	**Best Ants**	**Iterations**	**Evaporation Rate**	**Alpha**	**Beta**	**Result (metres)**
1	1	1000	0.9	1	4	12908980

1	1	1000	0.9	1	4	**12657670**
1	1	1000	0.9	1	4	12816280
1	1	1000	0.9	1	4	13125800
1	1	1000	0.9	1	4	**13257710**
1	1	1000	0.9	1	4	12923060
1	1	1000	0.9	1	4	12857670
1	1	1000	0.9	1	4	12977690

Πίνακας 12 : Πολλαπλές εκτελέσεις ίδιων παραμέτρων εισόδου

Βλέπουμε ότι δεν υπάρχουν μεγάλες διαφορές στα αποτελέσματα που έχουν προκύψει, καθώς η μεγαλύτερη διαφορά είναι περίπου 600km (δηλαδή το αποτέλεσμα 13257,710 – 12657,670). Το διαφορετικό αποτέλεσμα σε κάθε εκτέλεση του αλγορίθμου κρίνεται φυσιολογικό καθώς σημαίνει ότι ο αλγόριθμος δεν έχει ακόμα συγκλίνει. Δοκιμάζουμε, λοιπόν να αυξήσουμε τις επαναλήψεις :

Total Ants	**Best Ants**	**Iterations**	**Evaporation Rate**	**Alpha**	**Beta**	**Result (metres)**
1	1	2000	0.9	1	4	**12579210**
1	1	2000	0.9	1	4	12857670
1	1	2000	0.9	1	4	**12955790**
1	1	2000	0.9	1	4	12914360
1	1	2000	0.9	1	4	12895640
1	1	2000	0.9	1	4	12671130
1	1	2000	0.9	1	4	12816280

Πίνακας 13 : Περισσότερες επαναλήψεις οδηγούν σε καλύτερα αποτελέσματα

Με τις παραπάνω εκτελέσεις, επαληθεύουμε και πάλι ότι περισσότερες επαναλήψεις επιφέρουν ακόμα καλύτερα αποτελέσματα ενώ είναι ευδιάκριτο ότι οι εκτελέσεις των 2000 επαναλήψεων έχουν μέγιστη διαφορά 12955790 – 12579210 = 376580 m δηλαδή αρκετά μικρότερη από αυτή των 1000 επαναλήψεων. Άρα, πράγματι με την αύξηση των επαναλήψεων ο αλγόριθμος συγκλίνει.

Στη συνέχεια, κάνουμε κάποιες εκτελέσεις με περισσότερα μυρμήγκια και κάποια από αυτά θεωρούνται επίλεκτα. Συνεχίζουμε με τις τιμές των παραμέτρων που ήδη βρήκαμε και πειραματιζόμαστε με το ποσοστό εξάτμισης της φερομόνης:

Total Ants	**Best Ants**	**Iterations**	**Evaporation Rate**	**Alpha**	**Beta**	**Result (metres)**
10	10	2000	0.9	1	4	12657670
10	7	2000	0.9	1	4	12657670
10	5	2000	0.9	1	4	**12429700**
10	2	2000	0.9	1	4	**12429700**
10	10	2000	0.8	1	4	12657670
10	7	2000	0.8	1	4	**12429700**
10	5	2000	0.8	1	4	12654210
10	2	2000	0.8	1	4	12565750
5	5	2000	0.9	1	4	12730170
5	3	2000	0.9	1	4	12657670
5	1	2000	0.9	1	4	12657670
5	5	2000	0.8	1	4	12657670
5	3	2000	0.8	1	4	12657670
5	1	2000	0.8	1	4	12657670

Πίνακας 14 : Εκτελέσεις με περισσότερα από 1 μυρμήγκια

Από τις παραπάνω εκτελέσεις συμπεράναμε κατά τον χρόνο των εκτελέσεων ότι η ύπαρξη περισσότερων μυρμηγκιών επιβαρύνει χρονικά την εκτέλεση του αλγορίθμου, κάτι που κρίνεται αναμενόμενο. Παρατηρούμε επίσης ότι, ενώ με 5 μυρμήγκια δεν πέσαμε κάτω από τα 12657670 μέτρα, όταν διπλασιάσαμε τον αριθμό των μυρμηγκιών καταφέραμε να πέσουμε στα 12429700 μέτρα. Μάλιστα, αυτό συνέβη δύο φορές με ποσοστό εξάτμισης φερομόνης 0.9 και μία με 0.8 ενώ και στις τρεις περιπτώσεις επιλέχθηκε κάποιο πλήθος των μυρμηγκιών να είναι επίλεκτα. Αυτό σημαίνει ότι όταν όλα τα μυρμήγκια επιλέγεται να αφήνουν φερομόνη, τα αποτελέσματα επηρεάζονται αρνητικά από το ποσοστό φερομόνης που συνεισφέρουν τα μυρμήγκια που δεν έχουν καταφέρει καλή διαδρομή.

Τέλος, για να δούμε αν διαδραματίζει εν τέλει σημαντικό ρόλο το ποσοστό εξάτμισης της φερομόνης, επιλέγουμε να κάνουμε εκτελέσεις με αρκετά ακραίες τιμές για τη συγκεκριμένη παράμετρο :

Total Ants	**Best Ants**	**Iterations**	**Evaporation Rate**	**Alpha**	**Beta**	**Result (metres)**
10	5	2000	0.2	1	4	12565750
10	2	2000	0.2	1	4	12616240
10	7	2000	0.2	1	4	**12429700**

Πίνακας 15 : Εξέταση της σημασίας του ποσοστού εξάτμισης

Διαπιστώνουμε ότι παρά το γεγονός ότι επιλέχθηκε ιδιαίτερα χαμηλό ποσοστό, δεν διαφοροποιήθηκε ιδιαίτερα το αποτέλεσμα. Αυτό το συμπέρασμα επιβεβαιώνει και τον χαμηλό συντελεστή βαρύτητας της φερομόνης σε σύγκριση με αυτόν της απόστασης.

3.5. Σύγκριση μεθευρετικών μεθόδων TS και ACO για το TSP

Με δεδομένο ότι η εφαρμογή της μεθευρετικής μεθόδου Tabu Search καθώς και η εφαρμογή της μεθευρετικής μεθόδου της βελτιστοποίησης με χρήση αποικίας μυρμηγκιών πραγματοποιήθηκαν πάνω στο ίδιο ακριβώς πρόβλημα πλανόδιου πωλητή, δηλαδή πάνω στο πρόβλημα των 16 ευρωπαϊκών πόλεων που παρουσιάστηκε ήδη παραπάνω, παρουσιάζει ιδιαίτερο ενδιαφέρον η σύγκριση των αποτελεσμάτων αυτών των δύο μεθόδων.

Αρχικά, πρέπει να επισημάνουμε ότι η επιλογή ίδιου τρόπου αναπαράστασης λύσης μας επιτρέπει να συγκρίνουμε τις δύο μεθόδους καθώς εξαιτίας αυτού του γεγονότος, οι δύο αλγόριθμοι εξερεύνησαν τον ίδιο χώρο αναζήτησης.

Έπειτα, αξίζει να σημειωθεί ότι σημαντικό ρόλο διαδραματίζει το σημείο εκκίνησης από το οποίο θα ξεκινήσει η κάθε μέθοδος για να παράξει τελικά μία υποβέλτιστη λύση. Η αρχική λύση που θα παραχθεί από την κατασκευαστική ευρετική μέθοδο είναι το σημείο εκκίνησης του αλγορίθμου. Αν αυτή απέχει πάρα πολύ από την πραγματική βέλτιστη λύση του προβλήματος, θα είναι πολύ δύσκολο αν όχι αδύνατο για την μεθευρετική μέθοδο να φτάσει κοντά σε αυτήν σε σύντομο χρόνο εκτέλεσης καθώς θα χρειαστούν πολλές επαναλήψεις για να καλυφθεί η διαφορά. Δεδομένου ότι η κατασκευστική μέθοδος του κοντινότερου γείτονα είναι ευρέως χρησιμοποιούμενη, θεωρούμε ότι η αρχική λύση που παράχθηκε στην περίπτωση του Tabu Search είναι αρκετά καλή. Σε ό,τι αφορά την βελτιστοποίηση με χρήση αποικίας μυρμηγκιών, δεν χρησιμοποιείται αρχικά κάποια κατασκευαστική μέθοδος για τη δημιουργία αρχικής λύσης καθώς το κάθε μυρμήγκι κάνει ένα μονοπάτι και στη συνέχεια γίνεται η ενημέρωση της κάθε λύσης με βάση τη φερομόνη.

Παρατηρούμε, λοιπόν, με τη σύγκριση των αποτελέσματων που προέκυψαν από τις καλύτερες εκτελέσεις των δύο μεθόδων ότι για την αναζήτηση Tabu η καλύτερη λύση ήταν μονόπατι με συνολική απόσταση $12416240\ μέτρα$. Μάλιστα, βλέπουμε ότι η συγκεκριμένη λύση παράχθηκε χωρίς να αυξηθεί ιδιαίτερα ο αριθμός των επαναλήψεων και το μήκος της λίστας tabu. Πιο συγκεκριμένα, υπενθυμίζουμε ότι οι ελάχιστες τιμές των παραπάνω παραμέτρων εισόδου που παρήγαγαν αυτό το αποτέλεσμα ήταν πλήθος 10 επαναλήψεων και μήκος λίστας απαγορευμένων κινήσεων ίσο με 4. Στον αντίποδα, με τη χρήση Ant Colony Optimization, παρ’ό,τι πειραματιστήκαμε δοκιμάζοντας διάφορες τιμές για τις παραμέτρους εισόδου του αλγορίθμου, δεν κατέστη εφικτό η τιμή της αντικειμενικής συνάρτησης να κατέβει κάτω από τα $12429700\ μέτρα$. Παρά το

γεγονός ότι η απόσταση μεταξύ των δύο λύσεων είναι σχετικά μικρή, καθώς η Tabu Search εξήγαγε καλύτερο αποτέλεσμα κατά $12429700 - 12416240 = 13460\ μέτρα$ ή $13{,}46\ km$, συμπεραίνουμε ότι η αναζήτηση Tabu έτρεξε πολύ πιο ικανοποιητικά καθώς παρήγαγε αυτό το αποτέλεσμα πολύ γρήγορα και χωρίς να αυξηθούν αρκετά ο αριθμός των επαναλήψεων και το μέγεθος της λίστας tabu. Το γεγονός ότι η Ant Colony Optimization δεν κατάφερε να παράξει καλύτερο αποτέλεσμα ενδεχομένως να οφείλεται στον τρόπο επιλογής των παραμέτρων του συστήματος, καθώς ίσως το γεγονός ότι δεν έγινε δοκιμή με πάνω από 10 μυρμήγκια να διαδραμάτισε σημαντικό ρόλο στο αποτέλεσμα.

4. Επίλογος

Παρακάτω επιχειρείται μία σύντομη σύνοψη της μελέτης που πραγματοποιήθηκε καθώς και των συμπερασμάτων που εξάγονται από αυτήν ενώ παρουσιάζονται και τα περιθώρια επέκτασης της εργασίας σε πεδία που βρίσκονται εντός του γενικότερου αντικειμένου της Συνδυαστικής Βελτιστοποίησης και των Μεθευρετικών Μεθόδων για την επίλυση προβλημάτων που ανήκουν σε αυτήν την οικογένεια.

4.1. Σύνοψη και συμπεράσματα

Στόχος της εργασίας ήταν η υλοποίηση σημαντικών μεθευρετικών μεθόδων για την επίλυση γνωστών προβλημάτων βελτιστοποίησης με χρήση της γλώσσας προγραμματισμού Python. Τα συμπεράσματα τα οποία εξάγονται, πέρα από το γεγονός ότι η Python είναι μία γλώσσα προγραμματισμού που έχει χαρακτηριστικά τα οποία ευνοούν την υλοποίηση μεθευρετικών μεθόδων, αφορούν τον τρόπο βέλτιστης ρύθμισης (tuning) των παραμέτρων εισόδου του κάθε αλγορίθμου μέσω δοκιμών ώστε να προκύψει ένα θεωρητικά καλό αποτέλεσμα, κοντά στη βέλτιστη λύση του προβλήματος. Αναλυτικότερα συμπεράσματα έχουν παρουσιαστεί ήδη πιο εκτεταμένα στην παρουσίαση της μεθοδολογίας του κάθε αλγορίθμου. Επιπρόσθετα, ενδιαφέρον παρουσιάζει και η σύγκριση διαφορετικών μεθευρετικών τεχνικών για την εύρεση υποβέλτιστης λύσης του ίδιου ακριβώς προβλήματος πλανόδιου πωλητή, καθώς δοκιμάστηκαν οι μέθοδοι Tabu Search και Ant Colony Optimization στην εύρεση λύσης προβλήματος TSP που αποτελείται από 16 ευρωπαϊκές πρωτεύουσες.

4.2. Μελλοντικές Επεκτάσεις

Η παρούσα διπλωματική εργασία παρουσίαζει βασικές ευρετικές και μεθευρετικές μεθόδους σε πολύ γνωστά προβλήματα βελτιστοποίησης. Μπορεί κάλλιστα να επεκταθεί με την εφαρμογή ευρετικών και μεθευρετικών μεθόδων σε άλλα επίσης πολύ γνωστά προβλήματα βελτιστοποίησης, όπως το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem), το Πρόβλημα Τοποθέτησης σε Κάδους (Bin Packing Problem), το πρόβλημα Καλύμματος Κόμβων (Vertex Cover), το πρόβλημα Καλύμματος Ακμών (Edge Cover) κ.ά. . Επίσης, η συγκεκριμένη εργασία θα μπορούσε να επεκταθεί περαιτέρω με την υλοποίηση ακόμα περισσότερων αλγορίθμων. Παραδείγματα μπορούν να αποτελέσουν οι Γενετικοί Αλγόριθμοι (Genetic Algorithms) και η Διαδικασία Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (Greedy

Randomized Adaptive Search Procedure, GRASP). Σε αυτό το πλαίσιο, υπάρχει η δυνατότητα διεύρυνσης της διπλωματικής με αλγορίθμους βασισμένους στη Νοημοσύνη Σμήνους, όπως η Βελτιστοποίηση Σμήνους Σωματιδίων (Particle Swarm Optimization) και η Βελτιστοποίηση Αποικίας Μελισσών (Bee Colony Optimization).

Βιβλιογραφία

Aarts, E. H. L., Korst, J. H. M., & Laarhoven, P. J. M. (1988). *A quantitative analysis of the simulated annealing algorithm: A case study for the traveling salesman problem.* Journal of Statistical Physics, 50, 189–206.

Applegate, D., Bixby, R. E., Chvátal, V., & Cook, W. (2006). *The traveling salesman problem: A computational study.* Princeton: Princeton University Press.

Azizi, N., & Zolfaghari, S. (2004). *Adaptive temperature control for simulated annealing: A comparative study.* Computers and Operations Research, 31, 2439–2451.

Cerny, V. (1985). *Thermodynamics approach to the traveling salesman problem: An efficient simulation algorithm*. Journal of Optimization Theory and Applications, 45, 41–51.

Collins, N. E., Eglese, R. W., & Golden, B. L. (1988). *Simulated annealing- an annotated bibliography*. American Journal of Mathematical and Management Science, 8, 209–307.

Colorni A., Dorigo M., Maniezzo V. (1991). *Distributed Optimization by Ant Colonies.*Proceedings of ECAL91 - European Conference on Artificial Life, Paris, France,Elsevier Publishing, 134–142

Connolly, D. T. (1987). *Combinatorial optimization using simulated annealing*, report, London School of Economics, London, WC2A 2AE, presented at the Martin Beale Memorial Symposium, London, July, 1987.

D. Applegate, R. Bixby, V. Chvatal, W. Cook (2007). *The traveling salesman problem: a computational study*. Princeton University Press

Dantzig, T., Mazur, J. (1930), *Numbers: The Language of Science*.

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1954). *Solution of a large-scale traveling salesman problem.* Operations Research, 2, 393–410.

Dekkers, A., & Aarts, E. (1991). *Global optimization and simulated annealing*. Mathematical Programming, 50, 367–393.

Dorigo, M. & Stützle T.(2004). *Ant Colony Optimization*, MIT Press.

Dorigo, M., Gambardella, L .M. (1997). *Ant colony system: a cooperative learning approach to the traveling salesman problem.* IEEE Transactions on Evolutionary Computation, 1(1), 53-66.

Dorigo, M., Maniezzo,V., Colorni,A.(1996). *Ant system: optimization by a colony of cooperating agents.IEEE Transactions on Systems*, Man and Cybernetics, Part B: Cybernetics, 26(1), 29-41.

Dorigo, M., Stützle,T.(2003). *The ant colony optimization metaheuristic: Algorithms, applications and advances.* In: Handbook of Metaheuristics (pp.250-285), Springer US.

E. Lawler, J. Lenstra, A. RinooyKan, D. Shmoys (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* New York: Wiley-Interscience Publications, Vol. 3

Eglese, R. W. (1990). *Simulated annealing: A tool for operational research.* European Journal of Operational Research, 46, 271–281.

Engelbrecht, A. P. (2006). *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons.

Fiechter, C. N. (1990). *A parallel tabu search algorithm for large scale traveling salesman problems (Working Paper 90/1).* Switzerland: Department of Mathematics, Ecole Polytechnique Federale de Lausanne.

Gewen Huang, Yanguang Cai, Hao Cai, *Multi-agent ant colony optimization for vehicle routing problem with soft time windows and road condition*, MATEC Web Conf. Volume 173, 2018. Available at: https://doi.org/10.1051/matecconf/201817302020 (2018)

Glover, F. (1989). *Tabu search–part I.* ORSA Journal on computing, 1(3), 190-206.

Glover, F. (1990). *Tabu search–part II.* ORSA Journal on computing, 2(1), 4-32.

Glover, F., & Taillard,E. (1993). *A user's guide to tabu search.Annals of operations research*, 41(1), 1-28.

Glover, F. (2003). *Future paths for integer programming and links to artificial intelligence*, Elsevier, Computers & Operations Research, Volume 13, Issue 5, 1986, Pages 533-549.

Volume 13, Issue 5, 1986, Pages 533-549

Glover, F., Laguna M., Marti,R. (2007*). Principles of tabu search.* In: Approximation Algorithms and Metaheuristics, 23, 1-12.

Golden, B. L., & Stewart, W. R. (1985). *Empirical analysis of heuristics.*

E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, & D. B. Shmoys (Eds.), *The traveling salesman problem* (pp. 207–250). Chichester: John Wiley.

Gong, G., Liu, Y., & Qian, M. (2001). *An adaptive simulated annealing algorithm.* Stochastic Processes and their Applications, 94(1), 95–103.

Gutin, G., & Punnen, A. P. (Eds.). (2002*). The traveling salesman problem and its variations.* Dordrecht, The Netherlands: Kluwer.

Hanafi, S. (2001). *On the convergence of tabu search.* Journal of Heuristics,7(1),47-58.

Hartmanis, J. (1989). *Gödel, von Neumann and the P=?NP Problem*, The Structural Complexity Column, Department of Computer Science, Cornell University

Hoffman, A. J., & Wolfe, P. (1985). History. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, & D. B. Shmoys (Eds.), *The traveling salesman problem* (pp. 1–16). Chichester: John Wiley.

Ingber, L. (1989). *Very fast simulated annealing.* Mathematical Computing Modeling, 12, 967.

Johnson, D. S., & McGeoch, L. A. (2002). *Experimental analysis of heuristics for the STSP*. In G. Gutin & A. P. Punnen (Eds.), The traveling salesman problem and its variations (pp. 369–444). Dordrecht: Kluwer.

Johnson, D. S., & Papadimitriou, C. H. (1985). *Performance guarantees for heuristics.* In E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, & D. B. Shmoys (Eds.), The traveling salesman problem (pp. 145–180). Chichester: John Wiley.

Jünger, M., Reinelt, G., & Rinaldi, G. (1994). *The traveling salesman problem.* In M. Ball, T. Magnanti, C. Monma, & G. Nemhauser (Eds.), Handbook on operations research and the management sciences (pp. 225–330). Amsterdam: North Holland.

Jünger, M., Reinelt, G., & Rinaldi, G. (1997). *The traveling salesman problem.* In M. Dell'Amico, F. Maffioli, & S. Martello (Eds.), Annotated bibliographies in combinatorial optimization (pp. 199–221). New York: Wiley.

Kalampogia, A. (2017). *MPEG-4, H.264 and H.265 Video Bandwidth Prediction via Markovian Models and Simulated Annealing*, Master thesis, School of Electrical and Computer Engineering, Technical University of Crete

Karp, R., & Steele, J. M. (1985). *Probabilistic analysis of heuristics.* In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, & D. B. Shmoys (Eds.), The traveling salesman problem (pp. 181–205). Chichester: John Wiley.

Karthik Srinivasan, Saipriya Satyajit, Bikash K. Behera, Prasanta K. Panigrahi, *Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience*, Department of Quantum Physics, Cornell University. Available at: https://arxiv.org/abs/1805.10928 (May 28 2018)

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). *Optimization by simulated annealing.* Science, 220, 671–680.

Klaus Jansen, Stefan E. J. Kraft (2015). *A Faster FPTAS for the Unbounded Knapsack Problem*, Department of Computer Science, Kiel University

Lam, J., & Delosme, J. M. (1988a). *An efficient simulated annealing schedule: Derivation*, Technical Report 8816, Electrical Engineering Department, Yale, New Havan, CT, September.

Lam, J., & Delosme, J. M. (1988b). *An efficient simulated annealing schedule: Implementation and Evaluation*, Technical Report 8817, Electrical Engineering Department, Yale, New Havan, CT, September.

Lance Fortnow, *The status of the P versus NP problem*, Communications of the ACM, 2009. Available at: http://doi.acm.org/10.1145/1562164.1562186 (September 2009)

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (Eds.). (1985). *The traveling salesman problem.* Chichester, UK: John Wiley.

Lundy, M., & Mees, A. (1986). *Convergence of an annealing algorithm.* Mathematical Programming, 34, 111–124.

Onder Belgin, Ismail Karaoglan, Fulya Altiparmak, *Two-echelon vehicle routing problem with simultaneous pickup and delivery: Mathematical model and heuristic approach*, Elsevier (Publisher), Computers & Industrial Engineering, Volume 115, Pages 1-16. Available at: https://doi.org/10.1016/j.cie.2017.10.032 (January 2018)

Reeves, C. R. (1993). *Modern heuristic techniques for combinatorial problems.* New York: John Wiley.

Reinelt, G. (1991). *TSPLIB–A traveling salesman library*. ORSA Journal on Computing, 3, 376–384.

Reinelt, G. (1994). *The traveling salesman: Computational solutions for TSP applications.* Berlin: Springer-Verlag.

Rutenbar, R. A. (1989). *Simulated annealing algorithms: An overview.* IEEE circuits and Devices Magazine, 5, 1989.

Stephen A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the third annual ACM symposium on Theory of computing, 1971. Available at: http://doi.acm.org/10.1145/800157.805047 (May 3 1971)

Szu, H., & Hartley, R. (1987). *Fast simulated annealing*. Physics Letter A, 122, 157.

Tien D Kieu (2018). *The Travelling Salesman Problem and Adiabatic Quantum Computation: An Algorithm*, Swinburne University of Technology, Victoria, Australia

Triki, E., Collette, Y., & Siarry, P. (2004). *A theoretical study on the behavior of simulated annealing leading to a new cooling schedule*. European Journal of Operational Research, 166, 77–92.

Utkarsh Jaiswal, Shweta Aggarwa, *Ant Colony Optimization*, International Journal of Scientific & Engineering Research, 2011. Available at: https://www.ijser.org/viewPaperDetail.aspx?JUL1104 (July 2011)

Van Laahoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated annealing: Theory and practice.* Dordrecht: Kluwer Academic Publishers.

van Laarhoven P. J. M., & Aarts, E. H. L. (1987). *Simulated annealing: Theory and application,* Dordrecht: Reidel.

Viswanath Nagarajan (2017). *Dynamic Programming (Knapsack and Bin Packing)*, Lecture Notes, IOE 691: Approximation Algorithms

Παράρτημα Α - Διάφορα

Στο παρόν παράρτημα φιλοξενείται το υλικό υποστήριξης της διπλωματικής εργασίας όπως διάφορα δεδομένα που αποτέλεσαν την είσοδο των υλοποιημένων προγραμμάτων.

Α.1 Τα σύνολα δεδομένα για το πρόβλημα σακιδίου

Ακολουθούν τα τρία διαφορετικά σύνολα δεδομένων που χρησιμοποιήθηκαν ως είσοδοι για τις διάφορες εκτελέσεις του αλγορίθμου Tabu Search για την επίλυση του 0-1 διδιάστατου προβλήματος σακιδίου. Το κάθε σύνολο δεδομένων αποτελείται από:

1. το πλήθος των διαστάσεων του προβλήματος, το οποίο στην περίπτωσή μας θα είναι σε όλα τα σύνολα δεδομένων ίσο με 2, καθώς
2. το πλήθος των αντικειμένων από τα οποία πρέπει να επιλεχθούν αυτά που θα εισαχθούν στο σακίδιο
3. τα χαρακτηριστικά του σακιδίου, δηλαδή τη συνολική χωρητικότητά του ως προς τις δύο διαφορετικές μεταβλητές του προβλήματος, που είναι το βάρος και ο όγκος και
4. τα χαρακτηριστικά του κάθε αντικειμένου που είναι υποψήφιο προς εισαγωγή στο σακίδιο, δηλαδή αφενός την αξία του και αφετέρου το βάρος και τον όγκο του.

Α.1.1 Σύνολο Δεδομένων 1

1. πλήθος διαστάσεων προβλήματος : 2
2. πλήθος αντικειμένων : 28
3. χαρακτηριστικά σακιδίου : α) βάρος : 600, β) όγκος : 600
4. χαρακτηριστικά αντικειμένων :

Α/Α	Αξία	Βάρος	Όγκος
0	1898	45	30
1	440	5	20
2	22507	85	125
3	270	150	5
4	14148	65	80
5	3100	95	25
6	4650	30	35
7	30800	12	73

8	615	170	12
9	4975	20	15
10	1160	40	15
11	4225	25	40
12	510	20	5
13	11880	3	10
14	479	7	10
15	440	25	12
16	490	12	10
17	330	22	9
18	110	25	10
19	560	9	20
20	24355	165	60
21	2885	2	40
22	11748	85	50
23	4550	15	36
24	750	9	49
25	3720	2	40
26	1950	4	19
27	10500	100	150

Πίνακας 16 : Τα χαρακτηριστικά όλων των αντικειμένων του συνόλου δεδομένων 1

Α.1.2 Σύνολο Δεδομένων 2

1. πλήθος διαστάσεων προβλήματος : 2
2. πλήθος αντικειμένων : 28
3. χαρακτηριστικά σακιδίου : α) βάρος : 500, β) όγκος : 500
4. χαρακτηριστικά αντικειμένων :

Α/Α	Αξία	Βάρος	Όγκος
0	1898	30	45
1	440	20	0
2	22507	125	85
3	270	5	150
4	14148	80	65
5	3100	25	95
6	4650	35	30
7	30800	73	0
8	615	12	170
9	4975	15	0
10	1160	15	40
11	4225	40	25

12	510	5	20
13	11880	10	0
14	479	10	0
15	440	12	25
16	490	10	0
17	330	9	0
18	110	0	25
19	560	20	0
20	24355	60	165
21	2885	40	0
22	11748	50	85
23	4550	36	0
24	750	49	0
25	3720	40	0
26	1950	19	0
27	10500	150	100

Πίνακας 17 : Τα χαρακτηριστικά όλων των αντικειμένων του συνόλου δεδομένων 2

Α.1.3 Σύνολο Δεδομένων 3

1. πλήθος διαστάσεων προβλήματος : 2
2. πλήθος αντικειμένων : 105
3. χαρακτηριστικά σακιδίου : α) βάρος : 400, β) όγκος : 400
4. χαρακτηριστικά αντικειμένων :

Α/Α	Αξία	Βάρος	Όγκος
0	41850	0	75
1	38261	0	40
2	23800	0	365
3	21697	0	95
4	7074	0	25
5	5587	0	17
6	5560	0	125
7	5500	0	20
8	3450	0	22
9	2391	0	84
10	761	0	75
11	460	0	50
12	367	0	15
13	24785	5	0
14	47910	10	0
15	30250	10	12

16	107200	50	0
17	4235	2	10
18	9835	5	0
19	9262	5	50
20	15000	10	0
21	6399	5	0
22	6155	6	10
23	10874	11	0
24	37100	41	0
25	27040	30	50
26	4117	5	60
27	32240	40	150
28	1600	2	0
29	4500	6	0
30	70610	100	75
31	6570	10	0
32	15290	25	102
33	23840	39	0
34	16500	30	0
35	7010	13	40
36	16020	30	60
37	8000	15	0
38	31026	60	165
39	2568	5	0
40	2365	5	0
41	4350	10	0
42	1972	5	45
43	4975	15	0
44	29400	91	0
45	7471	24	0
46	2700	10	25
47	3840	15	0
48	22400	90	150
49	3575	15	0
50	13500	60	0
51	1125	5	0
52	11950	55	158
53	12753	60	0
54	10568	50	85
55	15600	75	95
56	20652	100	0
57	13150	65	89
58	2900	15	20
59	1790	10	0
60	4970	30	0
61	5770	35	0

62	8180	50	0
63	2450	15	0
64	7140	45	0
65	12470	80	80
66	6010	40	0
67	16000	110	110
68	11100	80	0
69	11093	80	15
70	4685	36	0
71	2590	20	60
72	11500	90	5
73	5820	50	135
74	2842	25	0
75	5000	50	0
76	3300	35	25
77	2800	30	0
78	5420	60	300
79	900	10	35
80	13300	150	100
81	8450	110	0
82	5300	70	0
83	750	10	25
84	1435	20	0
85	2100	30	0
86	7215	104	225
87	2605	40	25
88	2422	40	0
89	5500	94	0
90	8550	150	0
91	2700	50	0
92	540	10	0
93	2550	50	0
94	2450	50	0
95	725	16	5
96	445	10	0
97	700	20	60
98	1720	50	0
99	2675	90	100
100	220	10	0
101	300	15	0
102	405	39	0
103	150	20	0
104	70	20	0

Πίνακας 18 : Τα χαρακτηριστικά όλων των αντικειμένων του συνόλου δεδομένων 3