Αλγοριθμικές Τεχνικές για Δικριτηριακά-Βέλτιστες Διαδρομές

Νταλαγιώργος Αχιλλέας

<Διπλωματική Εργασία>

Επιβλέπων: Σπυρίδων Κοντογιάννης

Ιωάννινα, Οκτώβριος, 2021

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Σπυρίδων Κοντογιάννη, Καθηγητή στο Τμήμα Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, όπως και για την πολύτιμη βοήθεια και καθοδήγησή του σε όλη τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Ημερομηνία 9/10/2021

Συγγραφέας Νταλαγιώργος Αχιλλέας

Περίληψη

Το πρόβλημα της εύρεσης βέλτιστων διαδρομών (Shortest Path Problem, SPP) αποτελεί ένα από τα συνηθέστερα προβλήματα, τόσο στην επιστήμη της πληροφορικής, όσο και της καθημερινότητάς μας. Η εύρεση των δικριτηριακά βέλτιστων διαδρομών (Bi-criteria Shortest Path Problems, BSPP), αποτελεί μια επέκταση του παραπάνω προβλήματος. Η βασική διαφορά των SPPs με τα BSPPs είναι πως ενώ στην πρώτη περίπτωση ως λύση μπορεί να επιστραφεί μια διαδρομή, ενώ στη δεύτερη επιστρέφεται ένα σύνολο διαδρομών, το οποίο περιέχει όλες εκείνες τις διαδρομές για τις οποίες δεν υπάρχει κάποια διαφορετική διαδρομή, η οποία να παρουσιάζει καλύτερες τιμές και για τα δύο κριτήρια. Το σύνολο αυτό ονομάζεται κατά Pareto βέλτιστο σύνολο και ανά τα χρόνια έχουν αναπτυχθεί διαφορετικές προσεγγίσεις προκειμένου να οδηγηθούμε αποδοτικά στην εύρεση του. Οι τέσσερεις κυριότερες προσεγγίσεις που έχουν κυριαρχήσει είναι οι μέθοδοι διόρθωσης ετικέτας (label correcting), οι μέθοδοι ορισμού ετικέτας (label setting), η μέθοδος κ-συντομότερων μονοπατιών (k'th shortest path-ranking) και η μέθοδος των δύο φάσεων (two-phase). Έχουν αναπτυχθεί ποικίλοι αλγόριθμοι οι οποίοι ανήκουν στις παραπάνω κατηγορίες και όχι μόνο. Η εργασία αυτή ασχολείται με τη μελέτη, την παρουσίαση, την ανάπτυξη και την πειραματική αξιολόγηση δύο πρόσφατων, αλγορίθμων, του Path-Pairs A* (PPA*) και του Biobjective A* (BOA*), προκειμένου να διαπιστωθεί ποιος από τους δύο παρουσιάζει καλύτερη αποδοτικότητα και σε ποιες περιπτώσεις.

Λέξεις Κλειδιά: Πρόβλημα Εύρεσης Βέλτιστων Διαδρομών (SPP), Πρόβλημα Εύρεσης Δικριτηριακά Βέλτιστων Διαδρομών (BSPP), κατά Pareto βέλτιστο σύνολο, PPA*, BOA*.

Abstract

The Shortest Path Problem or SPP. is a common problem that concerns both computer science and our daily lives. Bi-criteria Shortest Path Problem or BSPP is an extension of the above problem. The primary difference between SPP's and BSSP's, is that the first problems return a single route as solution, while the second problems we return a set of individual routes. If a route is contained in this set, this means that there is no other route, which cost is less for both criteria. This set that mentioned above, which is the solution of BSPP's, is called Pareto optimal frontier and over the years different methods have been developed for it to be efficient. All these methods are categorized in four main approaches. Those are the Label Setting and Label Correcting approaches, the K'th Shortest Path or Ranking approach and the Two-Phase method. Various algorithms, which belong in those four main approaches have been developed. This diploma thesis deals with the study, presentation, implementation and experimental evaluation of two recently developed algorithms, the Bi-Objective A* and Path Pair A*. We compare the two algorithms, in order to determine, which one returns the most efficient results depending on the situation and parameters of the experiment.

Keywords: Shortest Path Problem (SPP), Bi-Objective Shortest Path Problem (BSPP), Pareto optimal frontier, Path Pair A* (PPA*), Bi-Objective A*(BOA*).

Περιεχόμενα

Κεφάλαιο 1. Εισαγωγή

Το *πρόβλημα εύρεσης Βέλτιστης Διαδρομής* δεν αποτελεί απλώς ένα επιστημονικό πρόβλημα του κλάδου της πληροφορικής, αλλά και ένα συνηθισμένο πρόβλημα της καθημερινότητας. Οι οδηγίες κατεύθυνσης σε έναν οδικό χάρτη αποτελούν ίσως το πιο απτό παράδειγμα του προβλήματος εύρεσης συντομότερης διαδρομής, στον πραγματικό κόσμο. Για την επίλυση του προβλήματος αυτού, έχουν αναπτυχθεί πασίγνωστες εφαρμογές που χρησιμοποιούντα καθημερινά από εκατομμύρια χρήστες, με το Google Maps να αποτελεί ίσως την δημοφιλέστερη αυτών. Ωστόσο, το πρόβλημα της Βέλτιστης Διαδρομής συναντάται και σε άλλες εξίσου σημαντικές, αλλά όχι τόσο προφανείς περιπτώσεις, όπως στα κατανεμημένα δίκτυα όπου το Amazon Web Service αποτελεί παράδειγμα, αλλά και σε δίκτυα διακομιστών αναμετάδοσης αλληλογραφίας – email services.

Για την αναπαράσταση οδικών χαρτών ή άλλων δικτύων μεταφορών, η **Θεωρία Γραφημάτων** αντιστοιχίζει ένα σημείο αναφοράς (π.χ., μια διασταύρωση, ένα σημείο εκκίνησης / τερματισμού, έναν σταθμό μετεπιβίβασης, κ.λπ.) του πραγματικού κόσμου ως μια κορυφή, ενώ ως ακμή αντιστοιχίζεται η απευθείας σύνδεση δύο σημείων αναφοράς που αντιστοιχούν στις κορυφές που βρίσκονται στα άκρα της ακμής. Κάθε ακμή θεωρείται ότι συνοδεύεται από κάποιον πραγματικό αριθμό (ή ακόμη κι από ένα διάνυσμα πραγματικών αριθμών) για την αποτύπωση του κόστους (π.χ., χρόνος, κατανάλωση καυσίμου, χιλιομετρική απόσταση) για τη διέλευσή της. Με αυτόν τον τρόπο λοιπόν δημιουργείται ένα εμβαρές γράφημα που αναπαριστά με συμπαγή και σαφή τρόπο ένα οδικό δίκτυο ή ένα δίκτυο δημόσιων μεταφορών. Επομένως, στη Θεωρία Γραφημάτων το πρόβλημα εύρεσης *«Βέλτιστου Μονοπατιού» (Shortest Path Problem, SPP),* ορίζεται ως το πρόβλημα εύρεσης μιας βέλτιστης διαδρομής μεταξύ δύο κορυφών ενός γραφήματος. Ως βέλτιστη θεωρούμε τη διαδρομή εκείνη στην οποία το άθροισμα των βαρών των ακμών ελαχιστοποιείται. Το πρόβλημα Βέλτιστου Μονοπατιού έχει

μελετηθεί εντατικά στην βιβλιογραφία [1] της Θεωρίας Γραφημάτων και έχουν βρεθεί πολλαπλές και ευρέως γνωστές λύσεις [24].

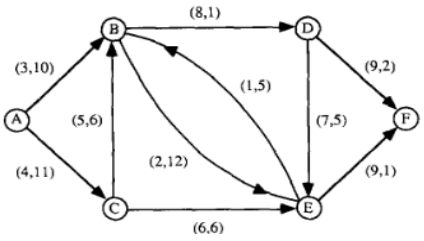
Ωστόσο, συχνά δεν αρκεί να περιορίσουμε το πρόβλημα σε ένα μόνο κριτήριο - κόστος. Στα οδικά δίκτυα, για παράδειγμα, η Βέλτιστη Διαδρομή θα μπορούσε να μετρηθεί τόσο ως προς την απόσταση, όσο και ως προς τον χρόνο ταξιδιού, ή ακόμη και ως προς το συνολικό χρηματικό κόστος. Έτσι γεννάται το πρόβλημα της εύρεσης των **Πολυκριτηριακά Βέλτιστων Διαδρομών,** το οποίο αποτελεί επέκταση του παραπάνω προβλήματος. Στη συγκεκριμένη εργασία θα ασχοληθούμε με μια υποκατηγορία των Πολυκριτηριακά Βέλτιστων Διαδρομών, τις **Δικριτηριακά Βέλτιστες Διαδρομές**.

Στις **Δικριτηριακά Βέλτιστες Διαδρομές** μας δίνεται ένα γράφημα *G = (V, E)* για το οποίο κάθε ακμή έχει δύο τιμές κόστους (π.χ., χρόνος διέλευσης και κατανάλωση καυσίμου). Επομένως, το πρόβλημα που καλούμαστε να επιλύσουμε, απαιτεί τον υπολογισμό της βέλτιστης διαδρομής που ισορροπεί μεταξύ και των δύο τιμών κόστους, των ακμών [3]. Λαμβάνοντας υπόψιν τα παραπάνω, αντιλαμβάνεται κανείς πως αυτό το καλά μελετημένο πρόβλημα [25] έχει πολλαπλές εφαρμογές στην καθημερινή ζωή, με προφανέστερη για άλλη μια φορά αυτή του οδικού δικτύου, αλλά και άλλων εφαρμογών, οι οποίες περιλαμβάνουν τον σχεδιασμό γραμμών μεταφοράς ενέργειας [26]. Όπως επίσης και τον σχεδιασμό του τρόπου μεταφοράς επικίνδυνων υλικών, προκειμένου να υπάρξει ισορροπία μεταξύ ελαχιστοποίησης της απόστασης της διαδρομής και του κινδύνου έκθεσης των κατοίκων στα υλικά αυτά [27].

Στη Θεωρία Γραφημάτων έχει αναπτυχθεί μια ποικιλία αλγορίθμων που δίνουν λύση στο πρόβλημα. Η επίλυση του προβλήματος εύρεσης δικριτηριακά βέλτιστης διαδρομής είναι πιο δύσκολη από την επίλυση του αντίστοιχου προβλήματος, της εύρεσης ελάχιστου μονοπατιού ενός κριτηρίου-κόστους. Η εύρεση των Δικριτηριακά Βέλτιστων Διαδρομών αποτελεί ένα NP-δύσκολο (NP-Hard) πρόβλημα, καθώς το μέγεθος του κατά Pareto βέλτιστου μετώπου αυξάνεται εκθετικά σε σχέση με τον αριθμό των κόμβων του γραφήματος, αλλά ακόμη και ο προσδιορισμός αν μία διαδρομή ανήκει στο κατά Pareto βέλτιστο μέτωπο. Αυτό λοιπόν, έχει ως αποτέλεσμα την ανάγκη ανάπτυξης τεχνικών που προσφέρουν γρήγορες και αποδοτικές λύσεις.

1.1 Το πρόβλημα των δικριτηριακά-βέλτιστων διαδρομών

Ας περιγράψουμε αρχικά το πρόβλημα των δικριτηριακά βέλτιστων διαδρομών. Έστω ότι έχουμε ένα ισχυρά συνδεδεμένο κατευθυνόμενο γράφημα *G=(V,E)*, όπου *V={1,…,n}* το σύνολο των κόμβων του γραφήματος και *E={(i,j),(k,l),…(x,y)}* το σύνολο των ακμών που ενώνουν κόμβους που ανήκουν στο V. Κάθε ακμή *(i,j)* που ανήκει στο *E* έχει δύο τιμές κόστους, κάθε μία από τις οποίες αφορά ένα διαφορετικό χαρακτηριστικό. Σε έναν οδικό χάρτη, για παράδειγμα, το ένα κόστος αφορά την απόσταση και το άλλο τον εκτιμώμενο χρόνο ταξιδιού. Σκοπό μας αποτελεί να βρούμε τη βέλτιστη διαδρομή μεταξύ δύο συγκεκριμενών κόμβων, ενός αρχικού και ενός τελικού.



Εικόνα 1.1 *Κατευθυνόμενο γράφημα με 2 κόστη*

Όμως είναι εξαιρετικά απίθανο να υπάρξει μια κατευθυνόμενη διαδρομή, τέτοια ώστε να ελαχιστοποιεί και τις δύο τιμές κόστους ταυτόχρονα. Έτσι, παράγεται ένα σύνολο από διαδρομές, στο οποίο καμία διαδρομή του συνόλου δεν είναι αυστηρά καλύτερη από καμία άλλη και για τις δύο τιμές κόστους. Αυτό το σύνολο λύσεων καλείται **μέτωπο κατά Pareto βέλτιστων διαδρομών** (***Pareto optimal frontier)***. Δυστυχώς το πρόβλημα, το οποίο καθορίζει αν μια διαδρομή ανήκει στο Pareto optimal frontier, συμπεριλαμβάνεται στην **NP-Hard** κλάση προβλημάτων [28]. Συγκεκριμένα, το πρόβλημα αυτό θεωρείται δυσεπίλυτο, καθώς το μέγεθος του Pareto μετώπου ενδέχεται να είναι ακόμη και εκθετικό σε σχέση με τον αριθμό των κόμβων του γραφήματος [3].

1.2 Ορισμοί

Στο κεφάλαιο αυτό παρατίθενται όλοι οι απαραίτητοι ορισμοί που χρησιμοποιούνται στην παρούσα εργασία.

1.2.1 Κυριαρχία (Dominance)

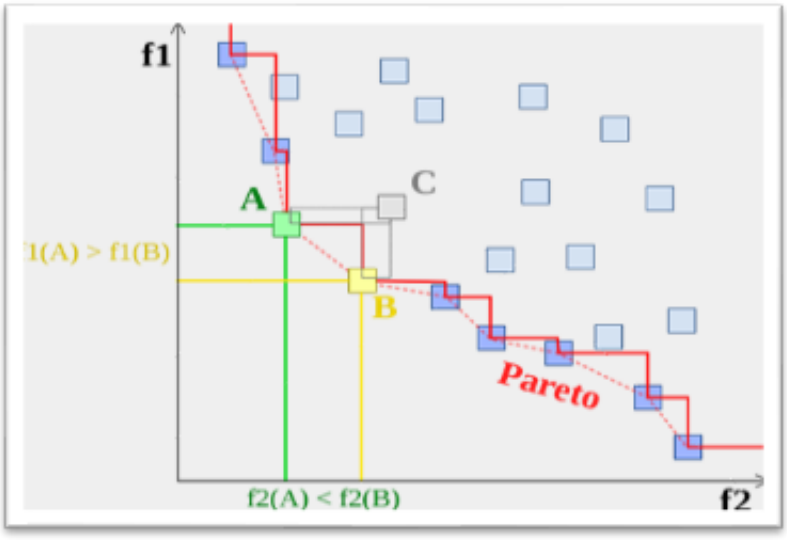
Έστω ότι π_v και π'_v δύο μονοπάτια-διαδρομές από την αφετηρία προς μια κορυφή *v*. Έστω c_1 και c_2 οι συναρτήσεις κόστους των δύο κριτηρίων, ανά διαδρομή.

- Λέμε ότι το π_v **κυριαρχεί αυστηρά** στο π'_v, όταν ισχύει ότι: $\left(c_1(\pi_v) < c_1(\pi'_v)\right) \wedge \left(c_2(\pi_v) < c_2(\pi'_v)\right)$.
- Λέμε ότι το π_v **κυριαρχεί (ασθενώς)** στο π'_v, όταν ισχύει ότι το π_v δεν υπολείπεται του π'_v ως προς κανένα από τα δυο κριτήρια, αλλά υπερτερεί του π'_v ως προς τουλάχιστον ένα από αυτά: $\left(c_1(\pi_v) \leq c_1(\pi'_v)\right) \wedge \left(c_2(\pi_v) \leq c_2(\pi'_v)\right) \wedge \left(\left(c_1(\pi_v) < c_1(\pi'_v)\right) \vee \left(c_2(\pi_v) < c_2(\pi'_v)\right)\right)$.

1.2.2 Μέτωπο Κατά Pareto βέλτιστων διαδρομών

Σε ένα πρόβλημα πολυκριτηριακής συνδυαστικής βελτιστοποίησης, κατά Pareto βελτιστότητα ή αποδοτικότητα (Pareto optimality or efficiency) ονομάζεται μία κατάσταση κατά την οποία οι πόροι ενός συστήματος δεν μπορούν να ανακατεμηθούν με τέτοιον τρόπο ώστε οι παραγόμενες λύσεις του προβλήματος να βελτιώσουν (σε σχέση με τις παραγόμενες λύσεις πριν την ανακατανομή των πόρων) ένα κριτήριο χωρίς να επιδεινωθεί κάποιο άλλο. *Το* **μέτωπο των κατά Pareto βέλτιστων λύσεων** *(Pareto-optimal frontier, ή Pareto set, ή Pareto front)* του προβλήματος, αποτελεί ένα σύνολο λύσεων *Π*, για το οποίο ισχύει ότι καμιά από τις λύσεις του *Π* δεν κυριαρχείται από καμιά άλλη λύση, ενώ ταυτόχρονα κάθε λύσε εκτός του *Π* κυριαρχείται από τουλάχιστον μία λύση του *Π*. Με άλλα λόγια, δεν υπάρχουν περιθώρια για περαιτέρω βελτίωση των ήδη παρεχόμενων λύσεων από το *Π*.

Για το συγκεκριμένο πρόβλημα εύρεσης Δικριτηριακά Βέλτιστων Διαδρομών με το οποίο θα ασχοληθούμε στην παρούσα εργασία, ως το **μέτωπο κατά Pareto βέλτιστων διαδρομών** Π_v μίας κορυφής *v*, ορίζεται ένα σύνολο διαδρομών που συνδέουν την κορυφή εκκίνησης με την κορυφή *v*, έτσι ώστε καμία διαδρομή που ανήκει στο Π_v να μην κυριαρχείται αυστηρά από οποιαδήποτε άλλη διαδρομή από την κορυφή εκκίνησης μέχρι την κορυφή v και κάθε διαδρομή από την κορυφή εκκίνησης μέχρι την κορυφή v να κυριαρχείται ασθενώς από μία διαδρομή που ανήκει στο Π_v.



Εικόνα 1.2.2.1 *Pareto frontier: Τα κουτιά αντιπροσωπεύουν τα διανύσματα κόστους για τις εφικτές λύσεις του προβλήματος. Οι μικρότερες τιμές-κόστη προτιμώνται έναντι των μεγαλύτερων .Το κουτί C δεν ανήκει στο Pareto frontier επειδή κυριαρχείται και από τα δύο κουτιά A και B. Αντιθέτως τα κουτιά A και B δεν κυριαρχούνται αυστηρά από κανένα άλλο κουτί επομένως ανήκουν στο Pareto frontier.*

1.2.3 Έλεγχος Κυριαρχίας (Domination Check)

Οι αλγόριθμοι εύρεσης Δικριτηριακά Βέλτιστων Διαδρομών κάθε φορά που εντοπίζουν μία νέα διαδρομή για μία κατάσταση εκτελούν έναν έλεγχο κυριαρχίας προκειμένου να καθορίσουν αν η διαδρομή αυτή κυριαρχείται από κάποια από τις προηγούμενες διαδρομές, για την ίδια κατάσταση. Αν η συνθήκη αυτή αληθεύει η διαδρομή απορρίπτεται. Επίσης, ελέγχουν αν η διαδρομή αυτή κυριαρχεί κάποια ή κάποιες από τις ήδη υπάρχουσες διαδρομές και επομένως απορρίπτουν αυτές.

1.2.4 Κατά προσέγγιση Κυριαρχία (approximated Dominance)

Έστω π_v και π'_v δύο μονοπάτια από την κορυφή εκκίνησης έως μια κορυφή v και έστω c_1 και c_2 οι συναρτήσεις κόστους μονοπατιών, για τα δύο κριτήρια. Έστω ε_1 και ε_2 δύο πραγματικές τιμές μεγαλύτερες ή ίσες του μηδενός. Λέμε ότι το μονοπάτι π_v ***(ε₁, ε₂)-κυριαρχεί*** στο μονοπάτι π'_v, εάν ισχύει πως $c_1(\pi_v) \leq (1+\varepsilon_1) \cdot c_1(\pi'_v) \wedge c_2(\pi_v) \leq (1+\varepsilon_2) \cdot c_2(\pi'_v)$.

1.2.5 Κατά προσέγγιση Pareto μέτωπο

Το **κατά προσέγγιση Pareto μέτωπο** $\Pi_v^{(\varepsilon_1,\varepsilon_2)}$ για δύο τιμές ε_1 και ε_2 μεγαλύτερες ή ίσες του μηδενός, αποτελεί ένα υποσύνολο του κατά Pareto βέλτιστου μετώπου Π_v, τέτοιο ώστε κάθε μονοπάτι που ανήκει στο κατά Pareto βέλτιστο μέτωπο Π_v *(*ε_1*,*ε_2*)*-κυριαρχείται από τουλάχιστον ένα μονοπάτι που ανήκει στο κατά προσέγγιση Pareto μέτωπο $\Pi_v^{(\varepsilon_1,\varepsilon_2)}$.

Το κατά Pareto βέλτιστο μέτωπο μιας κορυφής *v* είναι ένα σύνολο από μονοπάτια τα οποία ενώνουν μια κορυφή εκκίνησης *v* με μια κορυφή προορισμό *u*, με τέτοιον τρόπο ώστε κανένα μονοπάτι που ανήκει στο Pareto-optimal frontier να μην κυριαρχείται από άλλο μονοπάτι από την *v προς την u*, αλλά και κάθε μονοπάτι από την v προς την *u* να κυριαρχείται από τουλάχιστον ένα μονοπάτι που ανήκει στο Pareto μέτωπο.

Ομοίως, για δύο τιμές $\varepsilon_1 \geq 0$ και $\varepsilon_2 \geq 0$ το *κατά προσέγγιση Pareto μέτωπο* ***(approximate Pareto frontier)*** αποτελεί ένα υποσύνολο του κατά Pareto βέλτιστου μετώπου, τέτοιο ώστε κάθε μονοπάτι που ανήκει στο Pareto-optimal frontier *(ε1,ε2)-κυριαρχείται* από ένα μονοπάτι που ανήκει στο *approximate Pareto frontier.*

Οι τιμές ε_1 και ε_2 ονομάζονται **παράγοντες προσέγγισης** (**approximation factors**).

1.2.6 Συνεπείς Ευρετικές Συναρτήσεις

Μια **ευρετική συνάρτηση** *h* είναι μια συνάρτηση της οποίας η *τιμή*, *h(v),* εκτιμά το κόστος μιας διαδρομής από την κορυφή *v* έως την κορυφή στόχο. Επομένως $h(v_{goal}) = 0$. Μια ευρετική συνάρτηση ονομάζεται **συνεπής** εάν για κάθε κορυφή *v*, η εκτίμησή της $h(v)$ είναι πάντα μικρότερη ή ίση με το εκτιμώμενο υπολειπόμενο κόστος $h(t)$ από οποιαδήποτε γειτονική κορυφή t προς τον στόχο, συν το πραγματικό κόστος *(c(v,t))* για την προσέγγιση αυτού του γείτονα από την *v.* Δηλαδή η h είναι συνεπής ευρετική συνάρτηση εάν: $h(v) \leq c(v,t) + h(t)$ για κάθε κορυφή *v* και κάθε έξω-γείτονα t της *v*.

1.2.7 Λεξικογραφική ταξινόμηση

Μία έννοια βελτιστοποίησης που χρησιμοποιείται στο πλαίσιο τις εύρεσης Δικριτηριακά Βέλτιστων Διαδρομών είναι η *λεξικογραφική ελαχιστοποίηση*. Στην περίπτωση των αλγορίθμων εύρεσης Δικριτηριακά Βέλτιστων Διαδρομών μεταξύ όλων των βέλτιστων πιθανών λύσεων προτιμάται μία διαδρομή έναντι των υπολοίπων. Αυτή είναι η *(k,l)-λεξικογραφικά μικρότερη* διαδρομή, όπου $k, l \in \{1,2\}$ είναι δείκτες των δύο διαφορετικών κριτηρίων κόστους.

Έστω ένα κριτήριο k, στο οποίο δίνουμε προτεραιότητα έναντι του άλλου κριτηρίου, έστω l. Έστω π και π' δύο κόμβοι-διαδρομές που περιέχουν τα k και l. Τότε λέμε πως το π *είναι (k,l)-λεξικογραφικά μικρότερο* από το π', $\boldsymbol{\pi \leq_{lex(k,l)} \pi'}$, εάν ισχύει $\pi_k < \pi'_k \vee (\pi_k = \pi'_k \wedge \pi_l < \pi'_l)$. Ονομάζουμε ένα μονοπάτι π, ***(k,l)-λεξικογραφικά* βέλτιστη λύση** αν κανένα άλλο μονοπάτι π' δεν είναι λεξικογραφικά μικρότερο από το π.

1.3 Σκοπός της εργασίας

Το αντικείμενο της εργασίας αυτής, αφορά την επεξήγηση του προβλήματος εύρεσης δικριτηριακά βέλτιστων διαδρομών και την υλοποίηση δύο αλγορίθμων οι οποίοι επιλύουν αποδοτικά το πρόβλημα, καθώς και την μεταξύ τους σύγκριση, προκειμένου να διαπιστωθεί, μέσω της πειραματικής αξιολόγησης, ποιος αλγόριθμος μεταξύ των δύο παρουσιάζει καλύτερη απόδοση και σε ποιες περιπτώσεις.

Συγκεκριμένα, μελετώνται και παρουσιάζονται αναλυτικά στην παρούσα εργασία δύο αλγόριθμοι, τους οποίους στη συνέχεια υλοποιήσαμε, και είναι ο **Bi-Objective A* (BOA*)** και ο **Path Pairs A* (PPA*)**. Όπως μπορεί να γίνει εύκολα αντιληπτό από τα ονόματά τους, και οι δύο αλγόριθμοι αποτελούν προσαρμογές του ευρέως διαδεδομένου αλγορίθμου **A*** για εύρεση μονοκριτηριακά βέλτιστων διαδρομών [28]. Ο αλγόριθμος A* αποτελεί τον πυρήνα πολλών ευρετικών αλγορίθμων αναζήτησης, λόγω των ισχυρών θεωρητικών του ιδιοτήτων, κυρίως όταν αυτές χρησιμοποιούνται σε συνδυασμό με συνεπείς ευρετικές συναρτήσεις [2]. Για την ανάπτυξη του αλγορίθμου Bi-Objective A* βασιστήκαμε στην εργασία των Carlos Hernandez Ulloa, William Yeoh, Jorge A. Baier, Han Zhang, Luis Suazo, Sven Koenig [2] και για τον αλγόριθμο Path Pairs A* βασιστήκαμε στην εργασία των Boris Goldin και Oren Salzman [3].

Με γνώμονα τη μελέτη των εργασιών αυτών, αναπτύξαμε αποδοτικές υλοποιήσεις των δύο αλγορίθμων χρησιμοποιώντας τη γλώσσα προγραμματισμού C. Έπειτα συγκρίναμε τα αποτελέσματα που παρήγαγαν κατά τον τερματισμό τους οι δύο αλγόριθμοι, στους οποίους είχαμε θέσει ως είσοδο το ίδιο σύνολο δεδομένων. Το σύνολο δεδομένων το οποίο χρησιμοποιήσαμε, προκειμένου να ελέγξουμε πειραματικά την απόδοσή τους, παρέχεται δημόσια από το Κέντρο Διακριτών Μαθηματικών και Θεωρητικής Επιστήμης Υπολογιστών (Center for Discrete Mathematics and Theoretical Computer Science (DIMACS)). Στη συνέχεια ακολούθησε η αξιολόγηση των αποτελεσμάτων αυτών και η εξαγωγή ορισμένων συμπερασμάτων.

Η εργασία μας βασίστηκε τόσο στις εργασίες που προαναφέρθηκαν όσο και σε πολλές άλλες, καθώς υπάρχει μεγάλο θεωρητικό υπόβαθρο για το πρόβλημα των δικριτηριακά Βέλτιστων Διαδρομών.

Κεφάλαιο 2. Θεωρητικό Υπόβαθρο προβλήματος

Τα προβλήματα εύρεσης βέλτιστης διαδρομής είναι από τα πιο εκτενώς μελετημένα προβλήματα βελτιστοποίησης. Τα προβλήματα εύρεσης δικριτηριακά βέλτιστων διαδρομών (Bi-Objective Shortest Path (**BSP**)) αποτελούν τη φυσική επέκταση των προβλημάτων εύρεσης μονοκρητιριακά βέλτιστων διαδρομών. Ανήκουν στην ευρύτερη κατηγορία των συνδυαστικών προβλημάτων πολυκριτηριακής βελτιστοποίησης **(Multiple Objective Combinational Optimization problems (MOCO))**. Τα BSP προβλήματα έχουν μελετηθεί εντατικά στη βιβλιογραφία, καθώς ο υπολογισμός τους αποτελεί μεταξύ άλλων θεμελιώδες πρόβλημα στα logistics και στις μεταφορές.

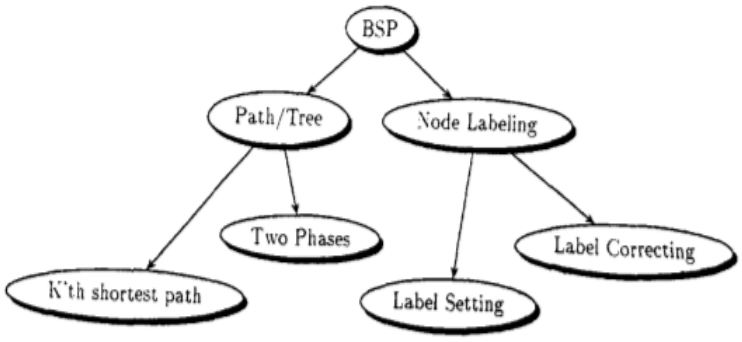
Ο Hansen [29] το 1980 ήταν ένας από τους πρώτους επιστήμονες που έδωσαν κάποιον αλγόριθμο επίλυσης για το BSP πρόβλημα. Έκτοτε ακολούθησαν πολλοί άλλοι ερευνητές, προσφέροντας μια μεγάλη ποικιλία μεθόδων και αλγορίθμων. Οι μέθοδοι και οι αλγόριθμοι που αναπτύχθηκαν, όπως διαπιστώθηκε περιείχαν κάποιες κοινές ιδέες ή χαρακτηριστικά. Αυτό οδήγησε στην κατηγοριοποίηση των αλγορίθμων που προσφέρουν λύση στο πρόβλημα, με βάση τη μεθοδολογία που αυτές ακολουθούσαν προκειμένου να βρεθούν οι βέλτιστες διαδρομές.

2.1 Κατηγοριοποίηση του προβλήματος

Για την επακριβή επίλυση των προβλημάτων εύρεσης βέλτιστης διαδρομής έχουν αναπτυχθεί δύο κύριες προσεγγίσεις. Η πρώτη εξ αυτών ονομάζεται μεθοδολογία **μονοπατιού/δέντρου(path/tree)** και η δεύτερη μεθοδολογία ετικετών κόμβων (**node labeling)**. Αυτές οι δύο προσεγγίσεις διασπώνται σε δύο μικρότερες υποκατηγορίες η κάθε μία. Η προσέγγιση μονοπατιού/δέντρου λοιπόν, διαιρείται στις προσεγγίσεις εύρεσης των k μονοκριτηριακά-βέλτιστων διαδρομών (***k'th shortest path_*** και στη ***μέθοδο δύο φάσεων (two phase method)***, ενώ η **node labeling** προσέγγιση διαιρείται

στις υποκατηγορίες αλγορίθμων που βασίζονται σε ορισμό ετικέτας (***label setting)*** και και αλγορίθμων που βασίζονται σε διόρθωση ετικέτας (***label correcting)*** [4].

Η ***k'th shortest path (ή ranking) μέθοδος*** προσέγγισης αρχικά αναπτύχθηκε από τους Carlyle και Wood [16]. Η μέθοδος δύο φάσεων για την επίλυση του BSP προβλήματος αναπτύχθηκε από τους Mote-Murthy & Olson [17]. Η label correcting προσέγγιση με επιλογή κόμβου, η οποία προσδιορίζεται ως η πιο επιτυχημένη προσέγγιση για την επίλυση των BSP προβλημάτων, αναπτύχθηκε από τους Skriver και Andersen[14], ενώ η μέθοδος με ορισμό ετικέτας (label setting) αναπτύχθηκε και παρουσιάστηκε ως ανώτερη από τους Guerriero και Musmanno [15].



Εικόνα 2: Κατηγοριοποίηση προσεγγίσεων για την επίλυση BSP προβλημάτων.

Πρόσφατα, το 2015 προτάθηκε μία νέα μέθοδος ακριβής επίλυσης του BSP προβλήματος για μεγάλης κλίμακας για οδικά δίκτυα [20], η οποία καλείται ***Αλγόριθμος Παλμού (Pulse Algorithm).*** Ο αλγόριθμος παλμού βασίζεται σε μία αναδρομική μέθοδο, η οποία χρησιμοποιεί στρατηγικές κλαδέματος που επιταχύνουν την εξερεύνηση του γραφήματος. Τα πειραματικά αποτελέσματα μάλιστα έχουν δείξει πως η μέθοδος αυτή είναι πολύ ανταγωνιστική.

2.1.1 Labeling προσέγγιση

Οι δύο πρώτες μέθοδοι ετικέτας που χρησιμοποιήθηκαν για την επίλυση των BSP προβλημάτων ήταν οι *label-correcting* και *label-setting* μέθοδοι που αναπτύχθηκαν από τους Hansen (1980) και Martins (1984), αντίστοιχα. Οι labeling προσεγγίσεις αποτελούν τις εξέχουσες προσεγγίσεις για την επίλυση των BSP προβλημάτων. Μπορούν να

χαρακτηριστούν ως μια επέκταση των μεθόδων που έχουν αναπτυχθεί για την επίλυση του μονοκριτηριακού προβλήματος εύρεσης βέλτιστης διαδρομής [9]. Η κύρια διαφορά τους, είναι πως στους label αλγορίθμους κάθε-ετικέτα αντιπροσωπεύει μια μερική διαδρομή από την αφετηρία μέχρι κάποιον ενδιάμεσο κόμβο (ή τον ίδιο τον προορισμό) στο γράφημα. Έτσι, όταν υπάρχουν δύο ή και περισσότερα κριτήρια, μπορεί να υπάρξουν πολλές ετικέτες στους ενδιάμεσους κόμβους του γραφήματος, αφού η κάθε ετικέτα αντιστοιχεί σε μία διαδρομή και όχι σε μια κορυφή του γραφήματος. Μάλιστα, οι ετικέτες που περιέχουν την ίδια κορυφή δεν κυριαρχούν απαραίτητα η μία την άλλη.

Η βασική λειτουργία των αλγορίθμων αυτών είναι η επέκταση της σχετικής αυτής μερικής διαδρομής, έτσι ώστε να καταλήξει ο αλγόριθμος σε μια νέα ετικέτα που αντιστοιχεί σε έναν διαφορετικό κόμβο, και ενδέχεται να κυριαρχείται (οπότε απλά αγνοείται), διαφορετικά «αποθηκεύεται» στη λίστα ετικετών αυτού του νέου κόμβου, για μελλοντικές επεκτάσεις. Οι αλγόριθμοι ετικέτας διαφέρουν στη σειρά με την οποία εκτελούνται αυτές οι στοιχειώδεις επεκτάσεις ετικετών και στις δομές δεδομένων που χρησιμοποιούνται για την αποτελεσματική διαχείριση της επιλογής και της κυριαρχίας ετικετών.

Η βασική διαφορά μεταξύ της label setting και της label correcting προσέγγισης είναι στο πώς διαχειρίζονται την ουρά προτεραιότητας (queue), η οποία χρησιμοποιείται για την αποθήκευση των ετικετών. Στην label setting περίπτωση μία ήδη οριστικοποιημένη ετικέτα παραμένει ίδια σε κάθε επανάληψη μετά την οριστικοποίησή της, ενώ στην label correcting περίπτωση όλες οι προσωρινές ετικέτες μπορούν να αλλάξουν τιμή, μέχρι το τέλος του αλγορίθμου, το οποίο σηματοδοτείται από την απουσία ακμής που να υποδεικνύει την ανάγκη αλλαγής κάποιας ετικέτας.

Δεδομένου πως υπάρχει συνήθως μεγάλο πλήθος ακμών σε σύγκριση με το πλήθος των κορυφών ενός γραφήματος, υπάρχουν πολλές πιθανότητες να υπάρχει εκθετικός αριθμός δέντρων. Για τον λόγο αυτόν, οι labeling προσεγγίσεις που συγκρίνουν τις τιμές των βαρών των ακμών στην περίπτωση των δύο κριτηρίων, σε κάθε κόμβο, φαίνεται να είναι πιο αποδοτικές από τις προσεγγίσεις path/tree.

2.1.1.1 Μέθοδος Διόρθωσης Ετικέτας (Label Correcting)

Η μέθοδος διόρθωσης ετικέτας αποτελεί μία από τις δύο υποκατηγορίες μεθοδολογίας ετικετών. Όμως μπορούμε να θεωρήσουμε πως και αυτή με την σειρά της διαιρείται σε δύο υποκατηγορίες. Ο διαχωρισμός των κατηγοριών γίνεται με βάση τον τρόπο διαχείρισης και επέκτασης των ετικετών. Έτσι αναπτύχθηκαν οι κατηγορίες της διόρθωσης με ***επιλογή ετικετών (label-selection)*** και της διόρθωσης με ***επιλογή κόμβου (node-selection)***. Κάθε ετικέτα λαμβάνει ξεχωριστή διαχείριση στην περίπτωση των διορθώσεων με επιλογή ετικέτας. Αντίθετα, όλες οι ετικέτες σε έναν επιλεγμένο κόμβο επεκτείνονται ταυτόχρονα κατά τη διόρθωση με επιλογή κόμβου (node-selection). Στη βιβλιογραφία έχει υπάρξει εκτενής έρευνα για το ποια στρατηγική αποτελεί πιο αποδοτική για την επίλυση των BSP [10], με την επιλογή κόμβου να είναι αυτή που επικράτησε. Η επιλογή ετικέτας χρησιμοποιείται κυρίως με την label setting προσέγγιση.

Ένας ενδεικτικός αλγόριθμος της μεθόδου διόρθωσης ετικετών με χρησιμοποίηση της μεθόδου επιλογής κόμβου είναι ο εξής:

Algorithm 4 Bi-objective Label Correcting

1: **input**: Graph $(\mathcal{V}, \mathcal{A})$, cost function $c = (c^1, c^2)$, and source node s.
2: $modNodes = \{s\}$: list of nodes with modified labels that have not yet been reconsidered, treated in FIFO order.
3: $Labels(s) = \{(0,0)\}$ and $Labels(i) = \emptyset, i \in \mathcal{V}\backslash\{s\}$: $Labels(i)$ is the list of labels at a particular node i.
4: **while** $modNodes$ is nonempty **do**
5: Remove first node i from $modNodes$. /* FIFO */
6: **for all** $a \in \mathcal{A}$ with $t(a) = i$ **do**
7: $j = h(a)$
8: $merge(Labels(i) + c_a, Labels(j))$ /* extend all labels at i by c_a and merge with labels at j, eliminating all dominated labels */
9: **if** the label set of j has changed and $j \notin modNodes$ **then**
10: Append j to $modNodes$. /* FIFO */
11: **end if**
12: **end for**
13: **end while**
14: **output:** Efficient path length from source node s to all other nodes, paths can be backtracked using labels.

Εικόνα 2.1.1.1.1 *Ψευδοκώδικας αλγορίθμου label correcting προσέγγισης*

- Αρχικά, ο μόνος κόμβος με ετικέτα είναι ο κόμβος πηγή *s*, με ετικέτα *Label(s)={(0,0)}*.
- Ο κόμβος *s* εισέρχεται σε μια ουρά προτεραιότητας FIFO *(γραμμ. 2)*.
- Αρχικοποιούμε μία λίστα για κάθε κόμβο *i* του γραφήματος, η οποία θα περιέχει όλα τα labels που αφορούν τον κόμβο αυτό*(γραμμ. 3)*.

- Από αυτό το σημείο ο αλγόριθμος λειτουργεί επαναληπτικά, ωσότου η ουρά προτεραιότητας FIFO μείνει άδεια μετά το τέλος μιας επανάληψης*(γραμμ. 5)*.
- Αφαιρεί τον πρώτο κόμβο, έστω *i* από την ουρά προτεραιότητας FIFO.
- Όλες οι ετικέτες του κόμβου *i* επεκτείνονται προς όλες τις εξερχόμενες ακμές του *i.* Έτσι δημιουργούνται νέα labels τα οποία αφορούν τους κόμβους γείτονες του *i*. Έστω *j* ένας κόμβος γειτονικός του *i*.
- Όταν μια ακμή εξέρχεται από έναν κόμβο με πολλαπλές ετικέτες, κάθε ετικέτα πρέπει να επεκτείνεται κατά την ακμή αυτή και να ελέγχεται για κυριαρχία με τις ετικέτες του κόμβου προορισμού της ακμής. Αυτή η λειτουργία ονομάζεται *συγχώνευση* (*merging*). Δηλαδή οι νέες ετικέτες ενός κόμβου *j* που δημιουργούνται από την επέκταση του κόμβου i προς τις εξερχόμενες ακμές του, συγχωνεύονται με τις ήδη υπάρχουσες, εάν υπάρχουν, ετικέτες του *j(γραμμ. 8)*.
- Αν η λίστα των ετικετών του *j* αλλάξει μετά την συγχώνευση και ο κόμβος *j* δεν περιέχεται στην ουρά προτεραιότητας FIFO, ο αλγόριθμος εισάγει τον *j* στην ουρά προτεραιότητας*(γραμμ. 9)*.
- Η συγχώνευση αποτελεί επίσης και τον έλεγχο κυριαρχίας του αλγορίθμου καθώς με τη διαδικασία αυτή παραμένουν στην λίστα των ετικετών ενός κόμβου *j* μόνο οι ετικέτες με τις μικρότερες τιμές κόστους και απαλείφονται οι υπόλοιπες.
- Οι νέες ετικέτες που δημιουργούνται, οι οποίες κυριαρχούνται (dominated labels) απαλείφονται από τις ετικέτες που εκτείνονται από τον κόμβο *i* σε ένα κόμβο *j* και τις ετικέτες που ήδη περιλαμβάνουν τον κόμβο *j*.
- Οι υπόλοιπες ετικέτες οι οποίες δεν κυριαρχούνται, «αφορούν» τον κόμβο *j* και εισέρχονται στην ουρά προτεραιότητας.
- Όταν δεν υπάρχουν πια άλλοι κόμβοι στην ουρά προτεραιότητας ο αλγόριθμος τερματίζει.
- Ο αλγόριθμος παράγει ως έξοδο τα βέλτιστα μονοπάτια από τον κόμβο πηγή προς όλους τους άλλους κόμβους*(γραμμ. 14)*.
- Η λίστα του τελικού κόμβου *t* θα περιέχει όλα τα μη-κυριαρχούμενα μονοπάτια από τον s στον *t. Οι* βέλτιστες διαδρομές μπορούν να ανακατασκευαστούν από τις τελικές λίστες ετικετών όλων των κόμβων, με την *οπισθοδρόμηση (backtracking)* των ετικετών.

Η περιγραφή του παραπάνω αλγορίθμου διόρθωσης ετικετών έγινε λαμβάνοντας υπόψιν την εργασία των Andrea Raith και Matthias Ehrgott [11].

2.1.1.2 Μέθοδος Ορισμού Ετικέτας (Label setting)

Ενώ στην προσέγγιση διόρθωσης ετικετών κανείς θα μπορούσε να επιλέξει μεταξύ της επιλογής ετικετών και της επιλογής κόμβων, κατά την εφαρμογή ενός αλγορίθμου ορισμού ετικέτας (label setting) είναι εφικτή μόνο η μέθοδος επιλογής ετικετών, για λόγους ορθότητας του αλγορίθμου. Αυτό συμβαίνει, καθώς στην περίπτωση αυτή πρέπει να είναι εγγυημένο πως έχει επιλεγεί μια ετικέτα μίας διαδρομής, η οποία δεν κυριαρχείται.

Οι αλγόριθμοι ορισμού ετικέτας μοιάζουν αρκετά με τους αλγορίθμους διόρθωσης ετικέτας που αναλύθηκαν παραπάνω, υπάρχουν όμως κάποιες βασικές διαφορές, όπως ακριβώς και μεταξύ του αλγορίθμου του Dijkstra και του αλγορίθμου Bellman-Ford για την εύρεση βέλτιστης μονοκριτηριακής διαδρομής. Απ' όλες τις ετικέτες πρέπει κάθε φορά να επιλεχθεί για οριστικοποίηση της τιμής της αυτή που εγγυημένα θα παραμείνει μη-κυριαρχούμενη κατά την εξέλιξη του αλγορίθμου. Υπάρχουν διάφοροι τρόποι επιλογής που μπορούν να εξασφαλίσουν την παραπάνω συνθήκη [12,13]. Ο τρόπος επιλογής που έχει κυριαρχήσει και συμμορφώνεται με αυτή την απαίτηση, είναι η επιλογή της **λεξικογραφικά μικρότερης** ετικέτας, η οποία επιλέγεται σε κάθε επανάληψη, μεταξύ όλων των υπολοίπων διαθέσιμων (δηλαδή, ακόμη προσωρινών) ετικετών, προκειμένου να επεκταθεί. Όλες οι νέες ετικέτες που δημιουργούνται από την επέκταση της συγκεκριμένης ετικέτας είναι προσωρινές.

Ένας ενδεικτικός αλγόριθμος της μεθόδου ορισμού ετικέτας (label setting) είναι ο εξής:

Algorithm 5 Bi-objective Label Setting

input: Graph $(\mathcal{V}, \mathcal{A})$, cost function $c = (c^1, c^2)$, and source node s.
$Labels(s) = \{(0,0)\}$ and $Labels(i) = \emptyset, i \in \mathcal{V}\backslash\{s\}$: $Labels(i)$ is the list of labels at a particular node i.
$TentativeLabels = \{(0,0) \triangleright s\}$: contains all tentative labels and the node i the label is associated with indicated by $\triangleright\, i$.
while $TentativeLabels \neq \emptyset$ **do**
  Remove a lex(1,2)-best label $(l_1, l_2) \triangleright i$ from $TentativeLabels$.
  for all a with $t(a) = i$ **do**
    $j = h(a)$
    $merge((l_1, l_2) + c_a, Labels(j))$ /* extend label (l_1, l_2) at i by c_a and merge with labels at j, eliminating all dominated labels */
    if the label set of j has changed **then**
      Insert new label $(l_1, l_2) + c_a \triangleright j$ into $TentativeLabels$.
      Remove all deleted labels at j from $TentativeLabels$.
    end if
  end for
end while
output: Efficient path length from source node s to all other nodes, paths can be backtracked using labels.

Εικόνα 2.1.1.2.1 *Ψευδοκώδικας αλγορίθμου label setting προσέγγισης.*

- Αρχικά, ο μόνος κόμβος με ετικέτα είναι ο κόμβος πηγή *s*, με ετικέτα *Label(s)={(0,0)}*.
- Ο αλγόριθμος διατηρεί μια λίστα για κάθε κόμβο με όλες τις ετικέτες που αφορούν τον συγκεκριμένο κόμβο*(γραμμ. 2)*..
- Επίσης σε αυτή την προσέγγιση διατηρούμε και έναν δυαδικό σωρό (binary heap) ο οποίος περιέχει όλες τις προσωρινές ετικέτες, αλλά και τον κόμβο στον οποίο καθεμιά από αυτές αντιστοιχεί*(γραμμ. 3)*.
- Ο αλγόριθμος λειτουργεί επαναληπτικά εωσότου ο σωρός αδειάσει μετά το τέλος μιας επανάληψης.
- Σε κάθε επανάληψη, η μικρότερη *λεξικογραφικά(1,2)* ετικέτα αφαιρείται από τον σωρό. Έστω ότι η ετικέτα αντιστοιχεί σε ένα κόμβο *i* *(γραμμ. 5).*
- Η ετικέτα του κόμβου *i* επεκτείνεται προς όλες τους γείτονες κόμβους του *i* μέσω όλων των εξερχόμενων *(i, j)* ακμών του. Έστω *j* ένας κόμβος γειτονικός του *i.* Η διαδικασία είναι παρόμοια με τη διαδικασία επέκτασης που περιγράψαμε για τη μέθοδο διόρθωσης ετικετών (label correcting), με την βασική διαφορά πως μόνο μία ετικέτα επεκτείνεται κάθε φορά και συγκρίνεται με όλες τις ετικέτες που αφορούν τον κόμβο προορισμού *j*.
- Η νέα ετικέτα που δημιουργείται μετά την επέκταση συγχωνεύεται με τις ήδη υπάρχουσες ετικέτες τις λίστας ετικετών του κόμβου *j(γραμμ. 8)*.
- Η συγχώνευση και σε αυτή την περίπτωση αποτελεί και τον έλεγχο κυριαρχίας.
- Αν η λίστα ετικετών του j έχει αλλάξει τότε η νέα ετικέτα εισέρχεται στον σωρό και όλες οι ετικέτες οι οποίες ανήκαν στην λίστα ετικετών του j και άλλαξαν αφαιρούνται από τον δυαδικό σωρό*(γραμμ. 9,10,11)*.
- Επομένως οι ετικέτες που κυριαρχούνται διαγράφονται και από τη λίστα ετικετών του κόμβου j αλλά και από τον σωρό των προσωρινών ετικετών.
- Όταν δεν υπάρχουν πια άλλες ετικέτες στον σωρό, ο αλγόριθμος τερματίζει.
- Ο αλγόριθμος παράγει ως έξοδο τα βέλτιστα μονοπάτια από τον κόμβο πηγή προς όλους τους άλλους κόμβους*(γραμμ. 15)*.

Η περιγραφή του αλγορίθμου αυτού, όπως και του παραπάνω αλγορίθμου έγινε λαμβάνοντας υπόψιν την εργασία των Andrea Raith και Matthias Ehrgott [11].

2.1.2 Μέθοδος Μονοπατιού/Δέντρου (Path/Tree approach)

Στη μέθοδο του μονοπατιού εξετάζουμε τα διαφορετικά διανύσματα μονοπατιών και προσπαθούμε να βρούμε τα πιο αποδοτικά. Ομοίως, ερευνούμε τα m-διάστασης χαρακτηριστικά διανύσματα (incidence vectors) που περιγράφουν τα διαφορετικά συνδετικά δένδρα (spanning trees) της μεθόδου δέντρου. Η μέθοδος αυτή, σύμφωνα με τν μελέτη του Skriver [4], διασπάται σε δύο υποκατηγορίες: σε αυτή των K'th shortest path και στη μέθοδο των δύο φάσεων. Ωστόσο νεότερες έρευνες [5,11] θεωρούν πως αυτές οι δύο υποκατηγορίες δεν ανήκουν σε μία ευρύτερη προσέγγιση, αλλά είναι ανεξάρτητες μεταξύ τους.

2.1.2.1 Μέθοδος K'th shortest path (ή ranking)

Οι αλγόριθμοι οι οποίοι βασίζονται στην προσέγγιση αυτή για την επίλυση του BSP προβλήματος, παράγουν τα μονοπάτια σταδιακά, το ένα μετά το άλλο κατά αύξουσα σειρά μήκους. Οι Climaco και Martins ήταν αυτοί που ανέπτυξαν πρώτοι αυτή τη θεωρία μαζί με έναν αλγόριθμο επίλυσης του προβλήματος [6]. Λίγο αργότερα, ο Martins ανέπτυξε έναν ακόμη αλγόριθμο ranking προσέγγισης [7]. Σύμφωνα με τη βιβλιογραφία, οι προσεγγίσεις k-shortest path δεν θα μπορούσαν να εφαρμοστούν επιτυχώς σε προβλήματα BSP, καθώς το κόστος εύρεσης των διαδρομών, κατά αύξουσα σειρά μήκους είναι πολύ υψηλό [4,8]. Επομένως, οι k-shortest path προσεγγίσεις δεν είναι ανταγωνιστικές σε σύγκριση με τις labeling προσεγγίσεις [8] και γι' αυτό το λόγο δεν θα επεκταθούμε παραπάνω.

2.1.2.2 Two Phases Method

Μια εναλλακτική λύση για την προσέγγιση κατάταξης για την BSP ακεραίων προβλημάτων είναι η μέθοδος Δύο Φάσεων. Η προσέγγιση αυτή εκμεταλλεύεται τη δομή του προβλήματος με το να υπολογίζει ξεχωριστά της υποστηριζόμενες (supported) και τις μη υποστηριζόμενες (non - supported) λύσεις. Οι Raith and Ehrgott [11], οποίοι

εξέτασαν βαθιά την προσέγγιση δύο φάσεων, διαπίστωσαν πως η αποτελεσματικότητα και η αποδοτικότητα της μεθόδου αυτή εξαρτάται πολύ από την δομή του δικτύου.

Πιο αναλυτικά, στη φάση 1 υπολογίζονται αποκλειστικά οι supported αποδοτικές λύσεις του προβλήματος. Δηλαδή οι αποδοτικές λύσεις που μπορούν να ληφθούν ως βέλτιστες σε ένα μονοκρητιριακό πρόβλημα σταθμισμένου αθροίσματος [18,19]. Η μέθοδος σταθμισμένου αθροίσματος συνδυάζει όλες τις πολυκριτηρικές συναρτήσεις σε μία κλιμακωτή, σύνθετη συνάρτηση χρησιμοποιώντας το άθροισμα βαρών.

Εξίσωση σταθμισμένου αθροίσματος:

$$\boldsymbol{f}(\boldsymbol{x}) = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_M f_M(x) \qquad \text{Όπου } \sum_{\mathrm{i}=1}^{\mathrm{M}} \mathrm{w_i} = 1\,, w_i \in (0,1)$$

Χρησιμοποιούνται δύο διχοτομικές μέθοδοι, η label correcting μέθοδος διχοτόμησης **(LDIC)** και η label setting μέθοδος διχοτόμησης **(DDIC)**, για τη λύση του μονοκριτηριακού προβλήματος σταθμισμένου αθροίσματος που προκύπτει από τη διχοτομική προσέγγιση [11]. Η διχοτομική προσέγγιση μπορεί να μην μας επιτρέψει να βρούμε όλα τα σημεία που δεν κυριαρχούνται, σε περίπτωση που υπάρχουν περισσότερες από δύο λύσεις στην ίδια όψη. Ωστόσο, όλα τα ακραία σημεία θα υπολογιστούν.

Στη φάση 2, οι υπόλοιπες υποστηριζόμενες και μη υποστηριζόμενες αποδοτικές λύσεις υπολογίζονται με μια αριθμητική προσέγγιση, καθώς δεν υπάρχει θεωρητικός χαρακτηρισμός για τον αποτελεσματικό υπολογισμό τους. Αναμένεται ότι ο χώρος αναζήτησης για τη μέθοδο κατάταξης (ranking) στη φάση 2 είναι πολύ περιορισμένος λόγω των πληροφοριών που λαμβάνονται στη φάση 1, έτσι ώστε τα σχετικά προβλήματα να μπορούν να λυθούν πολύ πιο γρήγορα με τον τρόπο αυτό, παρά με την επίλυση του BSP μόνο με μια καθαρά αριθμητική προσέγγιση.

Κεφάλαιο 3. Υλοποίηση

Στο κεφάλαιο αυτό θα περιγράφει λεπτομερώς ο τρόπος με τον οποίο υλοποιήσαμε τους δύο αλγορίθμους **Bi-Objective A* (BOA*)** και **Path Pair A* (PPA*)**, βασιζόμενοι σε ήδη υπάρχουσες μελέτες [2,3].

3.1 Γλώσσα προγραμματισμού

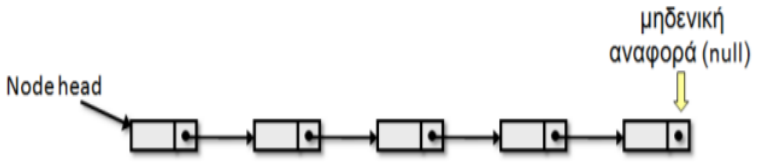
Αρχικά να αναφέρουμε πως η υλοποίηση και των δύο αλγορίθμων, έγινε στην γλώσσα προγραμματισμού C, η οποία αποτελεί μία διαδικαστική (procedural) γλώσσα προγραμματισμού. Δηλαδή, καθορίζει μια σειρά από καλά ορισμένα βήματα και διαδικασίες που οδηγούν στην ανάπτυξη προγραμμάτων και περιέχει μια αυστηρά καθορισμένη σειρά δηλώσεων συναρτήσεων και εντολών. Επίσης προσφέρει πρόσβαση σε κατώτερα επίπεδα στη δομή της μνήμης και της γλώσσας, που αντιστοιχεί σε αποδοτικές λειτουργίες του μηχανήματος και σε ελαχιστοποίηση του χρόνου εκτέλεσης. Το γεγονός αυτό λειτουργεί προς όφελός μας κατά την πειραματική αξιολόγηση, όπου απαιτείται η επανάληψη εκτέλεσης του αλγορίθμου για αρκετά παραδείγματα.

3.2 Δομή της υλοποίησης

Η υλοποίηση των δύο αλγορίθμων παρουσιάζει αρκετά κοινά στοιχεία. Θα μπορούσαμε να θεωρήσουμε πως η υλοποίηση των αλγορίθμων διαιρείται σε τρία μέρη.

1. Το πρώτο μέρος περιλαμβάνει την συλλογή-διάβασμα των πληροφοριών από ένα σύνολο δεδομένων. Στην υλοποίησή μας αυτό το σύνολο δεδομένων είναι χάρτες του οδικού δικτύου πολιτειών των ΗΠΑ. Οι αλγόριθμοι δημιουργούν ένα γράφημα διαβάζοντας τα δεδομένα από το σύνολο. Για τη δημιουργία του γραφήματος χρησιμοποιείται δομή δεδομένων με συνδεδεμένες λίστες. Οι συνδεδεμένες λίστες αποθηκεύουν ένα σύνολο στοιχείων σε κόμβους που

συνδέονται σε μια σειρά με συνδέσμους από κάθε κόμβο στον επόμενό του. Εξαιτίας της δομής αυτής, παρέχουν τη δυνατότητα γρηγορότερης εισαγωγής και διαγραφής στοιχείων, γι' αυτό και χρησιμοποιούνται ευρέως για την απεικόνιση γραφημάτων.



Εικόνα 3.2.1 *Απεικόνιση Συνδεδεμένης Λίστας*

2. Το δεύτερο μέρος της υλοποίησης των αλγορίθμων, αφορά την εύρεση των ευρετικών παραμέτρων, οι οποίες θα χρησιμοποιηθούν στο τρίτο μέρος για την εύρεση των Pareto optimal sets. Η ευρετική συνάρτηση της υλοποίησής μας αφορά την ελάχιστη υπολειπόμενη απόσταση όλων των ενδιάμεσων κόμβων του γραφήματος από τον κόμβο στόχο (target node). Προκειμένου να υπολογίσουμε τις ευρετικές αυτές παραμέτρους χρησιμοποιούμε μια αντίστροφη (reverse) εκδοχή του αλγορίθμου **Djikstra** [30]. Στην ουσία χρησιμοποιούμε τον Djikstra αλλά θέτοντας ως κόμβο έναρξης τον κόμβο-στόχο και αντιστρέφοντας τη φορά σε όλες τις κατευθυνόμενες ακμές του γραφήματος. Ο αλγόριθμος Djiksrta θα χρησιμοποιηθεί δύο φορές, μία για κάθε μετρική κόστους των ακμών.

Τα δύο πρώτα μέρη είναι κοινά και για τους δύο αλγορίθμους BOA* και PPA*, γι' αυτό και η ανάλυσή τους παρακάτω θα γίνει μόνο μια φορά.

3. Θα μπορούσαμε να θεωρήσουμε το τρίτο μέρος ως τον πυρήνα των δύο αλγορίθμων. Στις δύο επόμενες υπό-ενότητες θα υπάρξει εκτενής ανάλυση του πυρήνα αυτού. Από το σημείο αυτό και στο εξής όταν αναφερόμαστε στους αλγορίθμους BOA* και PPA* θα θεωρούμε το τρίτο μέρος των υλοποιήσεων μας.

Στο σημείο αυτό θα ξεκινήσουμε μια σύντομη ανασκόπηση των αλγορίθμων καλύτερης πρώτης αναζήτησης (**best-first search algorithms**), καθώς οι αλγόριθμοι που υλοποιήσαμε βασίζονται σε μεγάλο βαθμό σε αυτό το αλγοριθμικό πλαίσιο. Ένας best-first search αλγόριθμος υπολογίζει μια βέλτιστη διαδρομή από έναν κόμβο εκκίνησης σε έναν κόμβο στόχο διατηρώντας μια ουρά προτεραιότητας, που περιέχει όλους τους κόμβους που δεν έχουν επεκταθεί

ακόμη. Κάθε κόμβος *x* συσχετίζεται με μια κορυφή *v* μέσω της αντιστοίχισης *s(x) = v*. Η ουρά αυτή ταξινομείται σύμφωνα με μια συνάρτηση κόστους, η οποία ονομάζεται *f τιμή* του κόμβου. *Η f τιμή* αυτή προσδιορίζει το υπολογισμένο κόστος διαδρομής από την αρχική κορυφή μέχρι και την κορυφή του κόμβου που βρισκόμαστε, η οποία ονομάζεται και *g τιμή*, συν την ευρετική εκτίμηση (δηλαδή, την *h* τιμή) για την απόσταση της κορυφής του κόμβου έως την κορυφή στόχο.

Σε κάθε επανάληψη ο αλγόριθμος εξάγει τον πιο υποσχόμενο κόμβο από την ουρά προτεραιότητας, δηλαδή αυτόν με την λεξικογραφικά μικρότερη *f-τιμή*, ελέγχει την δυνατότητα να είναι μια καλύτερη λύση από αυτές που έχουν ήδη υπολογιστεί. Εάν αυτή η συνθήκη επιβεβαιωθεί και έχουμε φτάσει στον κόμβο στόχο, ο κόμβος προστίθεται στο σύνολο των λύσεων. Στους αλγορίθμους εύρεσης βέλτιστης διαδρομής με ένα μόνο κριτήριο, μόλις βρεθεί μια λύση, η αναζήτηση μπορεί να τερματιστεί. Εάν δεν έχουμε φτάσει στον κόμβο στόχο επεκτείνουμε τη διαδρομή που ο κόμβος αντιπροσωπεύει, προς κάθε έξω-γείτονα της αντίστοιχης κορυφής *s*(*x*). Έπειτα ελέγχουμε ξανά αν κάθε νέα διαδρομή (αντιστοιχεί σε καινούρια ετικέτα) *y* που θα σχηματιστεί, έχει τη δυνατότητα να είναι μια καλύτερη λύση από αυτές που έχουμε βρει μέχρι τώρα για την κορυφή *s*(*y*) στην οποία αντιστοιχεί. Μόνο αν συμβαίνει αυτό προσθέτουμε τον κόμβο *y* στην ουρά προτεραιότητας.

Διάφοροι αλγόριθμοι αναζήτησης με ένα μόνο κριτήριο, όπως *ο Dijkstra, ο A* και ο A*ε*, καθώς και οι αλγόριθμοι με δύο κριτήρια που εμείς εξετάζουμε, όπως οι BOA* και PPA*, εμπίπτουν στο παραπάνω πλαίσιο. Ωστόσο, διαφέρουν ως προς τον τρόπο που η σειρά προτεραιότητας ταξινομείται, αλλά και στον τρόπο που οι διάφορες συναρτήσεις υλοποιούνται.

Παρακάτω θα περιγράψουμε αναλυτικά τις λειτουργίες και τον τρόπο ταξινόμησης της ουράς προτεραιότητας των δύο best-first search αλγορίθμων που υλοποιήσαμε.

3.3 Αντίστροφος Djikstra με δομή σωρού

Όπως ήδη έχουμε αναφέρει, χρησιμοποιούμε τον reverse Djikstra αλγόριθμο προκειμένου να υπολογίσουμε τις ευρετικές τιμές για τον αλγόριθμό μας. Η reverse έννοια προκύπτει, καθώς αντιστρέφουμε τις εισερχόμενες και τις εξερχόμενες ακμές του

γραφήματος, προκειμένου να υπολογίσουμε την ελάχιστη απόσταση από την κορυφή στόχο προς όλες τις υπόλοιπες κορυφές του γραφήματος. Χρησιμοποιούμε δομή σωρού προκειμένου να μειώσουμε δραστικά τον χρόνο που χρειάζεται προκειμένου να υπολογιστούν οι ελάχιστες αποστάσεις.

Ο Αλγόριθμος Djikstra λοιπόν, διατηρεί δύο σύνολα, το ένα σύνολο περιέχει την λίστα κορυφών που έχουν ήδη συμπεριληφθεί στο ***Shortest Path Tree (SPT)*** [21], ενώ το άλλο τις κορυφές που δεν περιλαμβάνονται ακόμη *(visited και unvisited σύνολα)*. Επειδή το γράφημα αναπαρίσταται με λίστες γειτνίασης, όλες οι κορυφές του μπορούν να διασχισθούν σε *O(|V|+|E|)* χρόνο χρησιμοποιώντας ***Best First Search (BFS)*** [22]. Όπου *|V|* το μέγεθος του συνόλου των κορυφών και *|E|* το μέγεθος του συνόλου των ακμών. Η ιδέα της υλοποίησής μας είναι να διασχίσουμε όλες τις κορυφές του γραφήματος χρησιμοποιώντας *BFS* και χρησιμοποιούμε έναν min-Heap σωρό προκειμένου να αποθηκεύσουμε όλες τις κορυφές, για τις οποίες η μικρότερη απόσταση δεν έχει οριστικοποιηθεί ακόμη (δεν περιλαμβάνονται στο *SPT*). Ο min-Heap σωρός χρησιμοποιείται ως ουρά προτεραιότητας προκειμένου να επιλέγουμε από το σύνολο των κορυφών που δεν έχουν ακόμη «συναντηθεί» από τον αλγόριθμο, την κορυφή με την μικρότερη τιμή ετικέτας. Με τη δομή του σωρού η χρονική πολυπλοκότητα βασικών λειτουργιών, όπως η εξαγωγή της μικρότερης τιμής, είναι πολύ αποδοτική, συγκεκριμένα, *O(Log(|V|)) στη χειρότερη περίπτωση*.

Μια αναλυτική περιγραφή του αλγορίθμου είναι η παρακάτω:

- Δημιουργούμε έναν σωρό μεγέθους *|V|*. Κάθε κόμβος του σωρού περιέχει τον αριθμό και την τιμή απόστασης της κορυφής.
- Αρχικοποιούμε τον σωρό με την κορυφή πηγή ως ρίζα του σωρού. Στην περίπτωσή μας θέλουμε να υπολογίσουμε τις ελάχιστες αποστάσεις της κορυφής στόχου των αλγορίθμων BOA* και PPA* από όλες τις άλλες κορυφές. Επομένως, έχοντας ήδη αντιστρέψει τις εισερχόμενες με τις εξερχόμενες ακμές των κορυφών θεωρούμε ως κορυφή πηγή του αλγορίθμου reverse Djikstra την κορυφή στόχο των αλγορίθμων BOA* και PPA*.
- Η τιμή της απόστασης της κορυφής πηγής τίθεται ίση με μηδέν, ενώ οι τιμές απόστασης των υπόλοιπων κορυφών αρχικοποιούνται στο άπειρο.
- Ο αλγόριθμος λειτουργεί επαναληπτικά. Εξάγει από τον σωρό την κορυφή με την ελάχιστη απόσταση, έστω *u*. Για κάθε γειτονική κορυφή της *u*, έστω *v,* ελέγχουμε αν υπάρχει στον σωρό, δηλαδή η τιμή της

μικρότερης απόστασης της δεν έχει οριστικοποιηθεί ακόμη. Αν η v ανήκει στον σωρό και η τιμή απόστασης της κορυφής είναι μεγαλύτερη από το βάρος της ακμής *(u,v)* συν την τιμή απόστασης της κορυφής *u*, τότε διασχίζουμε τον σωρό και ενημερώνουμε την τιμή απόστασης της κορυφής *v* με *O(log(|V|))* χρονική πολυπλοκότητα, θέτοντας ως νέα τιμή απόστασης την τιμή του παραπάνω αθροίσματος.

- Κάθε φορά που μια κορυφή εξάγεται από τον σωρό, ο αλγόριθμος ανακατανέμει τις κορυφές που ανήκουν στον σωρό στις σωστές τους θέσεις έτσι ώστε η νέα ρίζα του σωρού να είναι η κορυφή με την μικρότερη πλέον τιμή.
- Ο αλγόριθμος τερματίζει όταν έχει ήδη εξάγει από τον σωρό όλες τις κορυφές του γραφήματος.
- Επιστρέφει μία λίστα με τις ελάχιστες αποστάσεις όλων των κορυφών. Κάθε κορυφή αντιστοιχεί σε μια θέση της λίστας.

3.4 Bi-Objective A* (BOA*)

Ο αλγόριθμος BOA* αποτελεί μια βελτίωση του αλγορίθμου **New Approach for Multi-Objective A* (NAMOA*).** Ο πρωταρχικός στόχος του αλγορίθμου είναι να πραγματοποιήσει όλους τους ελέγχους κυριαρχίας σε σταθερό χρόνο.

Όταν αναφερόμαστε σε έναν κόμβο του αλγορίθμου BOA*, στην ουσία αναφερόμαστε σε μία τετράδα τιμών, παρόμοια με τα labels που χρησιμοποιούνται ευρέως στην ερευνητική βιβλιογραφία που αφορά τις πολυκριτηριακά ή δικριτηριακά βέλτιστες διαδρομές. Κάθε κόμβος ***x*** περιέχει μία κορυφή του γραφήματος *s(x),* μία τιμή *g(x),* μία τιμή *f(x)* και μία κορυφή-γονιό *parent(x).* Ένας κόμβος αντιστοιχεί σε μία διαδρομή από την κορυφή εκκίνησης μέχρι την κορυφή *s(x)* κόστους *g(x).*

- Η *s(x)* τιμή αφορά την κορυφή του γραφήματος η οποία περιέχεται στον κόμβο *x*. Για να μην γίνει παρανόηση, επαναλαμβάνουμε πως ο κόμβος *x* αντιστοιχεί σε μία διαδρομή από την starting κορυφή μέχρι την κορυφή *s(x)* και όχι σε μια κορυφή του γραφήματος.

- Η *g(x)* τιμή αποτελεί μια δυάδα τιμών *g(x) = (g_1(x), g_2(x)).* Οι τιμές αυτές αναφέρονται στο κόστος διαδρομής από την κορυφή εκκίνησης μέχρι και την κορυφή *s(x)* του κόμβου *x*. Κάθε μία από τις $g_1(x)$ και $g_2(x)$ αφορούν τις δύο διαφορετικές τιμές των κριτηρίων των ακμών του γραφήματος.

- Η *f(x)* τιμή αποτελεί και αυτή μια δυάδα τιμών $f(x) = (f_1(x), f_2(x))$. Οι τιμές αυτές αναφέρονται στο άθροισμα του κόστους διαδρομής από την κορυφή εκκίνησης μέχρι και την κορυφή *s(x)* του κόμβου *x*, συν τις ευρετικές παραμέτρους $h(x) = (h_1(x), h_2(x))$ που, όπως αναφέρθηκε στο βήμα 2, αφορούν κάτω φράγματα των υπολειπόμενων αποστάσεων από την κορυφή $s(x)$ μέχρι την κορυφή στόχο. Οι τιμές αυτές, f_1, f_2, δηλαδή αντιπροσωπεύουν το κατώτατο όριο του συνολικού κόστους διαδρομής των λύσεων που μπορούν να παραχθούν από το συγκεκριμένο μονοπάτι. Ισχύει πως $f(x) = g(x) + h(x)$, ή $(f_1(x), f_2(x)) = (g_1(x) + h_1(x)$ *και* $g_2(x) + h_2(x))$. Κάθε μία από τις f_1, f_2 αφορούν τις δύο διαφορετικές μετρικές κόστους των ακμών του γραφήματος.

- Η τιμή *parent(x)* αποτελεί την κορυφή του γραφήματος η οποία επεκτάθηκε στην κορυφή *s(x).* Για να μην γίνει παρανόηση και εδώ να τονίσουμε πως η τιμή *parent(x)* αφορά μία απλή κορυφή του γραφήματος και όχι έναν κόμβο.

3.4.1 Αναλυτική περιγραφή του αλγορίθμου BOA*

- Ο αλγόριθμος δέχεται ως είσοδο ένα γράφημα, δηλαδή το σύνολο των κορυφών και των ακμών του γραφήματος και τα δύο κόστη όλων των ακμών. Ως είσοδο δέχεται επίσης και τις κορυφές εκκίνησης και τερματισμού, αλλά και το σύνολο των ευρετικών παραμέτρων που υπολογίστηκαν στο βήμα 2 για κάθε ενδιάμεση κορυφή.

Algorithm 2: Bi-Objective A* (BOA*)

Input : A search problem $(S, E, \mathbf{c}, s_{start}, s_{goal})$ and a consistent heuristic function $\mathbf{h}$
Output: A cost-unique Pareto-optimal solution set

1 $sols \leftarrow \emptyset$
2 **for each** $s \in S$ **do**
3 $\quad g_2^{\min}(s) \leftarrow \infty$
4 $x \leftarrow$ new node with $s(x) = s_{start}$
5 $\mathbf{g}(x) \leftarrow (0, 0)$
6 $parent(x) \leftarrow$ null
7 $\mathbf{f}(x) \leftarrow (h_1(s_{start}), h_2(s_{start}))$
8 Initialize $Open$ and add x to it
9 **while** $Open \neq \emptyset$ **do**
10 $\quad$ Remove a node x from $Open$ with the lexicographically smallest f-value of all nodes in $Open$
11 $\quad$ **if** $g_2(x) \geq g_2^{\min}(s(x)) \vee f_2(x) \geq g_2^{\min}(s_{goal})$ **then**
12 $\quad\quad$ **continue**
13 $\quad g_2^{\min}(s(x)) \leftarrow g_2(x)$
14 $\quad$ **if** $s(x) = s_{goal}$ **then**
15 $\quad\quad$ Add x to $sols$
16 $\quad\quad$ **continue**
17 $\quad$ **for each** $t \in \mathrm{Succ}(s(x))$ **do**
18 $\quad\quad y \leftarrow$ new node with $s(y) = t$
19 $\quad\quad \mathbf{g}(y) \leftarrow \mathbf{g}(x) + \mathbf{c}(s(x), t)$
20 $\quad\quad parent(y) \leftarrow x$
21 $\quad\quad \mathbf{f}(y) \leftarrow \mathbf{g}(y) + \mathbf{h}(t)$
22 $\quad\quad$ **if** $g_2(y) \geq g_2^{\min}(t) \vee f_2(y) \geq g_2^{\min}(s_{goal})$ **then**
23 $\quad\quad\quad$ **continue**
24 $\quad\quad$ Add y to $Open$
25 **return** $sols$

Εικόνα 3.4.1.1 *Ψευδοκώδικας για την ανάπτυξη του αλγορίθμου BOA**

•Δημιουργεί μία λίστα από g_2^{min} τιμές προκειμένου να εξασφαλίσουμε τον έλεγχο κυριαρχίας σε σταθερό χρόνο *(γραμμ. 2,3)*. Η λίστα g_2^{min} περιέχει την μικρότερη g_2 τιμή για κάθε κορυφή του γραφήματος. Κάθε φορά που ένας κόμβος *x* επεκτείνεται η g_2^{min} τιμή της κορυφής s(x) που περιέχεται στον κόμβο x ενημερώνεται. Οι τιμές αυτές θα χρησιμοποιηθούν κατά τον έλεγχο κυριαρχίας (domination check).

- Αρχικοποιούμε μία λίστα *sols* η οποία μετά τον τερματισμό του αλγορίθμου θα περιέχει το κατά Pareto βέλτιστο σύνολο λύσεων του προβλήματος εύρεσης δικριτηριακά ελάχιστων διαδρομών *(γραμμ. 1)*.

- Δημιουργούμε έναν δυαδικό σωρό ο οποίος ονομάζεται *OpenList*. Στην *OpenList* αποθηκεύονται οι μη-κυριαρχούμενοι κόμβοι, ταξινομημένοι κατά την λεξικογραφικά μικρότερη *f τιμή*. Ο πρώτος κόμβος που αποθηκεύεται στην *OpenList* είναι ο κόμβος εκκίνησης (*source node*) *(γραμμ. 8)*.

- Ως κόμβο εκκίνησης source θεωρούμε τον κόμβο ο οποίος περιέχει την κορυφή εκκίνησης του γραφήματος. Οι *g*-τιμές του κόμβου θα είναι ίσες με μηδέν, οπότε οι *f* τιμές του θα είναι ίσες με τις ευρετικές παραμέτρους. Δηλαδή *g(source) = (0,0)* και *f(source) = (h_1, h_2)*. Ο κόμβος εκκίνησης είναι και ο πρώτος κόμβος ο οποίος εισέρχεται στην *OpenList*. Ένα παράδειγμα source node είναι το εξής:

- $\boldsymbol{source} = \left(\boldsymbol{s_{starting}}, \ (\mathbf{0}, \mathbf{0}), \ \left(\boldsymbol{h_1^{starting}}, \boldsymbol{h_2^{starting}} \right), \ \boldsymbol{null} \right)$

- Από το σημείο αυτό και στο εξής ο αλγόριθμος λειτουργεί επαναληπτικά. Στην αρχή κάθε επανάληψης αφαιρείται από την *OpenList* ο κόμβος *x = (s(x), g(x), h(x), parent(x))* που περιέχει τη λεξικογραφικά μικρότερη *f-τιμή (γραμμ. 10).*

- Ελέγχουμε αν ο συγκεκριμένος κόμβος που *x* με την *(f_1.f_2)- λεξικογραφικά* μικρότερη τιμή που εξήγαμε από την *OpenList* κυριαρχείται *(γραμμ. 11)*. Αν κυριαρχείται, αγνοούμε τον συγκεκριμένο κόμβο και συνεχίζουμε στην επόμενη επανάληψη. Αναλυτικά στο 2.

- Αν ο κόμβος *x* δεν κυριαρχείται, ενημερώνουμε την g_2^{min} λίστα με τη νέα μικρότερη g_2*-τιμή (γραμμ. 13).* Η τιμή θα είναι μικρότερη, καθώς ο έλεγχος κυριαρχίας *(γραμμ. 11)* δεν ήταν αληθής.

- Στο σημείο αυτό ελέγχουμε αν η κορυφή *s(x)* του κόμβου *x* είναι η κορυφή προορισμού s_{goal} *(γραμμ. 14)*. Αν είναι, προσθέτουμε τον κόμβο στην λίστα *sols* και συνεχίζουμε σε νέα επανάληψη, καθώς έχουμε βρει μία νέα λύση μη-κυριαρχούμενη λύση *(γραμμ. 15)*.

- Στην περίπτωση που η παραπάνω υπόθεση δεν ισχύει, επεκτείνουμε τον κόμβο *x* προς όλους τους έξω-γείτονες της κορυφής *s(x) (γραμμ. 17-21)*. Αναλυτικά στο 1. Πριν εντάξουμε στην *OpenList* τους κόμβους οι οποίοι θα δημιουργηθούν για κάθε κορυφή γείτονα, εκτελούμε ξανά έλεγχο κυριαρχίας*(γραμμ. 22)* για να ελέγξουμε αν οι συγκεκριμένοι κόμβοι κυριαρχούνται από τους κόμβους που έχουν ήδη επεκταθεί ή της λύσεις που έχουν ήδη βρεθεί. Η ανάλυση έλαβε χώρα στο 2. Αν όντως κυριαρχούνται από άλλους κόμβους τότε τους απορρίπτουμε, αλλιώς τους προσθέτουμε στην *OpenList*.

- Ο αλγόριθμος τερματίζει όταν η OpenList αδειάσει και επιστρέφει τελικά τη λίστα *sols,* η οποία περιέχει το σύνολο όλων των κατά Pareto βέλτιστων διαδρομών από την κορυφή αφετηρίας μέχρι την κορυφή προορισμού.

1. Ένας κόμβος x επεκτείνεται ως εξής: ο κόμβος επεκτείνεται προς όλες τις εξερχόμενες ακμές *(s(x), s(y))* της κορυφής, την οποία περιέχει. Ως ***y*** θεωρούμε έναν νέο κόμβο που θα δημιουργηθεί. Ο *y* περιέχει μια κορυφή *s(y),* γειτονική της κορυφής *s(x).* Η g *τιμή* του *y* θα ισούται με την g *τιμή* του *x* συν το κόστος της ακμής από την *s(x)* στην *s(y)* κορυφή. Δηλαδή, $g(y) = g(x) + c\big(s(x), s(y)\big)$. *H f(y) τιμή* θα ισούται με την *g(y) τιμή* συν την ευρετική τιμής της *s(y).* Δηλαδή, $f(y) = g(y) + h\left(s(y)\right)$. Τέλος, η τιμή *parent(y)* ισούται με την κορυφή *s(x).*

2. Ο έλεγχος κυριαρχίας $g_2(x) \geq g_2^{min}\big(s(x)\big)$ *or* $f_2(x) \geq g_2^{min}\left(s(v_{goal})\right)$ *(γραμμ. 11)*

    a. με την πρώτη ανισότητα εξετάζει αν έχει ήδη υπάρξει καλύτερη διαδρομή από την κορυφή εκκίνησης έως και την κορυφή *s(x)*. Αυτό συμβαίνει καθώς οι κόμβοι κατανέμονται στην OpenList με *(f_1f_2)-λεξικογραφική σειρά*, δηλαδή δίνεται προτεραιότητα στην *f_1-τιμή* έναντι της *f_2-τιμής*. Επομένως αν ένας κόμβος, ο οποίος περιείχε την κορυφή *s(x)* έχει εξαχθεί από την λίστα και έχει επεκταθεί θα είχε λεξικογραφικά μικρότερη *f_1-τιμή* από τον κόμβο που εξετάζεται, άρα αρκεί η σύγκριση να γίνει μόνο για τις *f_2-τιμές*. Όπως έχει αναφερθεί οι *f-τιμές* αποτελούν το άθροισμα των *g-τιμών* με τις h ευρετικές παραμέτρους $f(x) = g(x) + h(x)$. Επομένως για δύο κόμβους που περιέχουν την ίδια κορυφή οι *h-τιμές* τους είναι ίσες, άρα η ταξινόμηση γίνεται με βάση τις *g-τιμές* τους. Επομένως αρκεί να συγκρίνουμε μόνο την *g_2-τιμή* του x με την μικρότερη διαδρομή-κόμβο που έχει ήδη βρεθεί για την κορυφή s(x), δηλαδή την $g_2^{min}\big(s(x)\big)$ τιμή. Επομένως αν η τιμή αυτή είναι μικρότερη ή ίση με την g_2(x) τιμή τότε ο κόμβος x κυριαρχείται.
    b. Η δεύτερη ανισότητα εξετάζει αν έχει υπάρξει ήδη κάποια λύση η οποία είναι καλύτερη από την βέλτιστη λύση που πιθανώς μπορεί η διαδρομή αυτή να δώσει. Και πάλι εξαιτίας τις προτεραιότητας που δίνουμε στις *f_1-τιμές* έναντι των *f2-τιμών*, στην ταξινόμησή τους στον δυαδικό σωρό, αν κάποια λύση έχει ήδη βρεθεί θα έχει μικρότερη *f_1-τιμή* από την λύση που πιθανώς να προκύψει από αυτή τη διαδρομή. Επομένως αρκεί να συγκρίνουμε μόνο την *$f_2(x)$* τιμή, η οποία αποτελεί την βέλτιστη λύση που μπορεί να υπάρξει από την διαδρομή που ο κόμβος x αντιπροσωπεύει με την μικρότερη έως τώρα *f_2-τιμή* των λύσεων που έχουν ήδη βρεθεί δηλαδή της $g_2^{min}\left(s(v_{goal})\right)$ τιμής. Με το τρόπο αυτό βλέπουμε αν η

διαδρομή που ο κόμβος x αντιπροσωπεύει μπορεί να προσφέρει μια πιο αποδοτική λύση από της ήδη υπάρχουσες. Για τους κόμβους που περιέχουν την κορυφή στόχο ισχύει $g_2\left(s\left(v_{goal}\right)\right) = f_2(s(v_{goal}))$ αφού η τιμή h για την κορυφή στόχο είναι μηδέν. Επομένως αν $f_2(x) \geq g_2^{min}(s(v_{goal}))$ δεν μπορεί να υπάρξει καλύτερη διαδρομή από τις ήδη υπάρχουσες και ο κόμβος *x* κυριαρχείται.

3.4.2 Approximated BOA* (BOA*$_\varepsilon$)

Στην πειραματική αξιολόγηση, προκειμένου να συγκρίνουμε τους αλγορίθμους BOA* και PPA* υλοποιήσαμε μια παραλλαγή του BOA*. Ο αλγόριθμος που υλοποιήσαμε υπολογίζει ένα κατά προσέγγιση Pareto μέτωπο. Ο αλγόριθμος είναι παρόμοιος με τον BOA*. Η μοναδική διαφορά εντοπίζεται στον έλεγχο κυριαρχίας.

Συγκεκριμένα, στον BOA* ο έλεγχος κυριαρχίας *(αναλύθηκε παραπάνω)* είναι ο :

$$if\ g_2(x) \geq g_2^{min}(s(x))\ or\ f_2(x) \geq g_2^{min}(s(target)) \rightarrow x\ is\ dominated$$

Στην παραλλαγή που χρησιμοποιούμε, προσθέτουμε την τιμή *1+ε* στο δεύτερο μέλος της εξίσωσης αυτής και ο προσεγγιστικός έλεγχος κυριαρχίας γίνεται ως εξής:

$$\boldsymbol{if}\ g_2(x) \geq g_2^{min}(s(x))\ \boldsymbol{or}\ (1+\varepsilon) * f_2(x) \geq g_2^{min}(s(target))\ \boldsymbol{then}\ x\ is\ dominated$$

Οι λύσεις που προκύπτουν *ε-κυριαρχούν* τις λύσεις του κατά Pareto βέλτιστου συνόλου του BOA*.

3.5 Path Pair A* (PPA*)

Ο αλγόριθμος Path Pair A* των Goldin και Salzman [3] τον οποίο και υλοποιήσαμε αποτελεί επίσης μια από τις προσαρμογές του περίφημου A* αλγορίθμου. Ο PPA* βασίζεται πάνω στον Multi-Objective A* (MOA*), ο οποίος αποτελεί πολύ-κριτηριακή επέκταση του A* αλγορίθμου.

Σε αντίθεση με τους τυπικούς αλγόριθμους αναζήτησης που δημιουργούν σταδιακά τις συντομότερες διαδρομές από την κορυφή του γραφήματος προς τις υπόλοιπες κορυφές, ο PPA* δίνει λύση στον υπολογισμό του προβλήματος της εύρεσης ***της (ε₁,ε₂)-προσεγγιστικής δικριτηριακά βέλτιστης διαδρομής***.

Αυτό συμβαίνει καθώς κατά την υλοποίηση του αλγορίθμου κατασκευάζουμε ένα κατά προσέγγιση Pareto μέτωπο (***partial Pareto frontier***) για τους ενδιάμεσους κόμβους, το οποίο επιτρέπει την δραματική μείωση του χρόνου των υπολογισμών. Στη βιβλιογραφία υπάρχουν πολλές μέθοδοι που χρησιμοποιούνται για το κλάδεμα των ενδιάμεσων διαδρομών. Μάλιστα, πολλές από αυτές χρησιμοποιούν πολύ συντηρητικά όρια προκειμένου να πετύχουν τον σκοπό αυτό. Αντιθέτως στην υλοποίησή μας χρησιμοποιούμε την έννοια του Μερικού κατά Pareto μετώπου, ως μια απλή αλλά αποτελεσματική μέθοδο για να «κλαδέψουμε» της κατά προσέγγιση κυριαρχούμενες λύσεις.

Μία από τις βασικές διαφορές μεταξύ του PPA* και του BOA* είναι η έννοια του κόμβου. Στον Path Pair A*, όπως προδίδει και το όνομά του, ένας κόμβος *x* προσδιορίζεται από ένα ζεύγος μονοπατιών ***(Path Pair)*** προς μια συγκεκριμένη κορυφή *v = s(x).* Ως ένα μονοπάτι μπορεί να θεωρηθεί ένας κόμβος με την μορφή των κόμβων του αλγορίθμου BOA*, που είδαμε προηγουμένως. Ο πρώτος κόμβος αντιπροσωπεύει ένα μονοπάτι $\boldsymbol{\pi_v^{tl}}$ από την αρχική κορυφή μέχρι την κορυφή *v*, ενώ ο δεύτερος κόμβος αντιπροσωπεύει ένα μονοπάτι $\boldsymbol{\pi_v^{br}}$. Τα *tl* και *br* αποτελούν στενογραφίες για το ***top-left*** και ***bottom-right*** αντίστοιχα: $\boldsymbol{Path\text{-}Pair_v = (\pi_v^{tl}, \pi_v^{br})}$.

Επιπλέον στον αλγόριθμο PPA* εισάγεται μία ακόμη λειτουργία, αυτή της συγχώνευσης ***(merging***). Δύο ζεύγη μονοπατιών (π_v^{tl},π_v^{br}) *και* $(\pi'^{tl}_v,\pi'^{br}_v)$ συγχωνεύονται, δηλαδή δημιουργούν ένα νέο ζεύγος μονοπατιών, σύμφωνα με τον παρακάτω κανόνα:

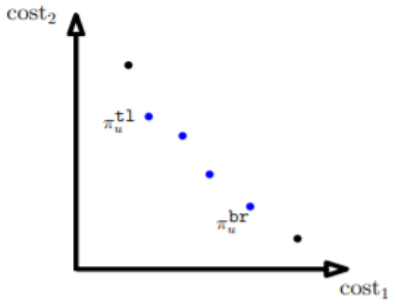
$$\pi^{tl}_{v\,new} = \pi^{tl}_v \quad \alpha\nu \quad c_1\left(\pi^{tl}_v\right) \le c_1\left(\pi'^{tl}_v\right) \qquad \pi^{br}_{v\,new} = \pi^{br}_v \; \alpha\nu \; c_2\left(\pi^{br}_v\right) \le c_2\left(\pi'^{br}_v\right)$$

$$\pi^{tl}_{v\,new} = \pi'^{tl}_v \quad \alpha\nu \quad c_1\left(\pi^{tl}_v\right) > c_1\left(\pi'^{tl}_v\right) \qquad \pi^{br}_{v\,new} = \pi'^{br}_v \; \alpha\nu \; c_2\left(\pi^{br}_v\right) > c_1\left(\pi'^{br}_v\right)$$

Η συγχώνευση ζευγών μονοπατιών συμβαίνει τόσο κατά την είσοδο ενός νέου κόμβου-ζεύγους στην ουρά προτεραιότητας, όσο και κατά την είσοδό του στο σύνολο πιθανών λύσεων.

3.5.1 Μερικό κατά Pareto μέτωπο

Στην υπό-ενότητα αυτή ορίζουμε την έννοια του μερικού κατά Pareto μετώπου *(Partial Pareto frontier, PPf).* Έστω ότι π_v^{tl} και π_v^{br} δύο μονοπάτια του Pareto frontier μιας κορυφής *v* τέτοια ώστε το κόστος του π_v^{tl} να είναι μικρότερο από το κόστος του π_v^{br}, δηλαδή, $\boldsymbol{c_1(\pi_v^{tl}) < c_1(\pi_v^{br})}$. Το μερικό κατά Pareto μέτωπο (*Partial Pareto frontier*) για ένα δεδομένο ζεύγος μονοπατιών $\boldsymbol{(\pi_v^{tl}, \pi_v^{br})}$ είναι ένα υποσύνολο του κατά Pareto μετώπου τέτοιο ώστε αν για το μονοπάτι π_v της κορυφής *v* ισχύει $\boldsymbol{c_1(\pi_v^{tl}) < c_1(\pi_v) < c1(\pi_v^{br})}$ τότε το μονοπάτι π_v ανήκει στο *Μερικό Pareto μέτωπο.* Τα μονοπάτια π_v^{tl} και π_v^{br} ονομάζονται ακραία μονοπάτια για το συγκεκριμένο μερικό κατά Pareto μέτωπο.



Εικόνα 3.5.1.1 *Το μερικό κατά Pareto σύνολο δύο μονοπατιών π_v^{tl} και π_v^{br} (μπλε) είναι το σετ όλων των μονοπατιών του Pareto frontier (μαύρα και μπλε), βρίσκονται ανάμεσα από τα π_v^{tl} και π_v^{br} μονοπάτια.*

3.5.2 $(\varepsilon_1, \varepsilon_2)$-φραγμένο Μερικό κατά Pareto μέτωπο

Ένα μερικό κατά Pareto μέτωπο λέμε πως είναι *$(\varepsilon_1,\varepsilon_2)$-φραγμένο εάν ισχύει :*

$$\varepsilon_1 \geq \frac{\{c_1(\pi_v^{br}) - c_1(\pi_v^{tl})\}}{c_1(\pi_v^{tl})} \quad \text{και} \quad \varepsilon_2 \geq \frac{\{c_2(\pi_v^{tl}) - c_2(\pi_v^{br})\}}{c_2(\pi_v^{br})}$$

Αν ένα μερικό κατά Partial μέτωπο Π είναι *$(\varepsilon_1, \varepsilon_2)$-φραγμένο* τότε κάθε μονοπάτι το οποίο ανήκει στο Π κυριαρχείται από τα δύο ακραία μονοπάτια του Π, το *top-left* μονοπάτι *(π^{tl})* και το *bottom-right* μονοπάτι *(π^{br})*.

3.5.3 Αναλυτική περιγραφή του αλγορίθμου PPA*

- Ο αλγόριθμος PPA* όπως και ο BOA* δέχεται ως είσοδο ένα γράφημα, δηλαδή το σύνολο των κορυφών και των ακμών του γραφήματος και τα δύο κόστη όλων των ακμών. Ως είσοδο δέχεται επίσης και την κορυφή εκκίνησης v_{start} και την κορυφή προορισμού v_{target} του γραφήματος, αλλά και το σύνολο των ευρετικών παραμέτρων που υπολογίστηκαν στο βήμα 2. Επιπλέον δέχεται ως είσοδο τις ε_1, ε_2 τιμές (παράγοντες προσέγγισης)

Algorithm 2 PP-A*

Input: $(G = (V, E), v_{\text{start}}, v_{\text{goal}}, c_1, c_2, h_1, h_2, \varepsilon_1, \varepsilon_2)$

1: solutions_pp← ∅ ▷ path pairs
2: OPEN ← new path pair $(v_{\text{start}}, v_{\text{start}})$
3: **while** OPEN ≠ ∅ **do**
4: $(\pi_u^{\text{tl}}, \pi_u^{\text{br}}) \leftarrow$ OPEN.extract_min()
5: **if** is_dominated_PP-A*$(\pi_u^{\text{tl}}, \pi_u^{\text{br}})$ **then**
6: **continue**
7: **if** $u = v_{\text{goal}}$ **then** ▷ reached goal
8: merge_to_solutions_PP-A*$(\pi_u^{\text{tl}}, \pi_u^{\text{br}}$, solutions_pp)
9: **continue**
10: **for** $e = (u, v) \in$ neighbors$(s(n), G)$ **do**
11: $\pi_v^{\text{tl}}, \pi_v^{\text{br}} \leftarrow$ extend_PP-A*$((\pi_u^{\text{tl}}, \pi_u^{\text{br}}), e)$
12: **if** is_dominated_PP-A*$(\pi_v^{\text{tl}}, \pi_v^{\text{br}})$ **then**
13: **continue**
14: insert_PP-A*$((\pi_v^{\text{tl}}, \pi_v^{\text{br}}), \text{OPEN})$
15: solutions← ∅
16: **for** $(\pi_{v_{\text{goal}}}^{\text{tl}}, \pi_{v_{\text{goal}}}^{\text{br}}) \in$ solutions_pp **do**
17: solutions ← solutions $\cup \{\pi_{v_{\text{goal}}}^{\text{tl}}\}$
18: **return** solutions

Εικόνα 3.5.3.1 *Ψευδοκώδικας για τον αλγόριθμο PPA**

- Δημιουργεί μία λίστα από g_2^{min} τιμές προκειμένου να εξασφαλίσουμε τον έλεγχο κυριαρχίας σε σταθερό χρόνο. Η λίστα έχει το ίδιο περιεχόμενο και τον ίδιο ρόλο όπως και στην περίπτωση του BOA* αλγορίθμου.

- Αρχικοποιούμε μία λίστα σωρό *sols_pp(γραμμ. 1)*, η οποία περιέχει το σύνολο των πιθανών λύσεων του αλγορίθμου. Ένας κόμβος – ζεύγος μονοπατιών που βρίσκεται στη λίστα μπορεί να αλλάξει μέχρι το τέλος του αλγορίθμου, αφού μπορεί να συγχωνευτεί με έναν άλλο κόμβο ο οποίος θα αποτελεί επίσης πιθανή λύση. Μετά το τέλος του αλγορίθμου η λίστα θα περιέχει όλα τα ζεύγη μονοπατιών που αποτελούν λύσεις και ανήκουν στο κατά Pareto βέλτιστο μέτωπο. Από τα ζεύγη αυτά, στο τέλος κρατάμε αυθαίρετα μόνο τα *top-left* μονοπάτια.

- Δημιουργούμε έναν δυαδικό σωρό ο οποίος ονομάζεται *OpenList(γραμμ. 2)*. Στην *OpenList* αποθηκεύονται ταξινομημένα τα ζεύγη μονοπατιών. Η ταξινόμηση γίνεται με βάση την λεξικογραφικά μικρότερη τιμή που προκύπτει από την f_1 *τιμή* του πρώτου μονοπατιού *(top-left μονοπατιού* π^{tl} *)*

και την f_2 *τιμή* του *(bottom-right μονοπατιού* π^{br}*).* Δηλαδή, το διάνυσμα με βάση το οποίο γίνεται η ταξινόμηση είναι $\left(\boldsymbol{f_1}(\boldsymbol{\pi_v^{tl}}), \boldsymbol{f_2}(\boldsymbol{\pi_v^{br}}) \right)$**.** Το πρώτο ζεύγος μονοπατιών *(Path Pair,* ***PP)*** που εισέρχεται στην *OpenList* αποτελείται από δύο μονοπάτια όμοια με τους κόμβους πηγή του BOA* αλγορίθμου.

- Για τον αρχικό *κόμβο-ζεύγος* που αποθηκεύεται στην OpenList οι g τιμές των θα είναι ίσες με μηδέν, οπότε οι f τιμές του θα είναι ίσες με τις ευρετικές παραμέτρους, για κάθε ένα από τα δύο μονοπάτια. Δηλαδή:
  $\boldsymbol{\pi_{starting}^{tl}} = \left(\boldsymbol{s}(\boldsymbol{starting}),\ (\mathbf{0},\mathbf{0}), \left(\boldsymbol{h_1}(\boldsymbol{starting}), \boldsymbol{h_2}(\boldsymbol{starting})\right), \boldsymbol{null}\right)$
  $\boldsymbol{\pi_{starting}^{br}} = \left(\boldsymbol{s}(\boldsymbol{starting}),\ (\mathbf{0},\mathbf{0}), \left(\boldsymbol{h_1}(\boldsymbol{starting}), \boldsymbol{h_2}(\boldsymbol{starting})\right), \boldsymbol{null}\right)$
  $\boldsymbol{PP_{source}} = \left(\boldsymbol{\pi_{starting}^{tl}}, \boldsymbol{\pi_{starting}^{br}}\right)$
  Αυτό αποτελεί ένα παράδειγμα του αρχικού *κόμβου-ζεύγους* το οποίο όμως μαρτυρά την μορφή όλων των *κόμβων που διαχειρίζεται ο PPA**.

- Από το σημείο αυτό και στο εξής ο αλγόριθμος PPA* λειτουργεί επαναληπτικά. Στην αρχή κάθε επανάληψης εξέρχεται από την *OpenList* το *PP* που περιέχει τη λεξικογραφικά μικρότερη $\left(f_1(\pi_v^{tl}), f_2(\pi_v^{br}) \right)$ *τιμή (γραμμ. 4)*, την οποία προσδιορίσαμε προηγουμένως**.**

- Ελέγχουμε αν το ο ο κόμβος που εξήγαμε από την OpenList *κυριαρχείται (γραμμ. 5).* Ο έλεγχος κυριαρχίας και στον PPA* γίνεται τόσο με τους κόμβους διαδρομές που έχουν επεκταθεί, όσο και με τις λύσεις που έχουν βρεθεί. Αν ο κόμβος κυριαρχείται, συνεχίζουμε στην επόμενη επανάληψη.

- Αν ο κόμβος δεν κυριαρχείται, ενημερώνουμε με την g_2^{min} λίστα με τη νέα μικρότερη g_2*-τιμή* για τις κορυφές που περιέχονται και στα δύο μονοπάτια.

- Έπειτα ελέγχουμε αν η κορυφή που εισέρχεται στον κόμβο είναι η κορυφή προορισμού*(γραμμ. 7)*.

- Αν είναι, ελέγχουμε αν το *PP* μπορεί να συγχωνευθεί με κάποιο διαφορετικό *PP* το οποίο ήδη ανήκει στην λίστα *sols_pp*. Αν μπορεί να συγχωνευτεί προσθέτουμε το νέο PP_{merged} στην *sols_pp* και αφαιρούμε το PP που προϋπήρχε και χρησιμοποιήθηκε για την συγχώνευση. Στην περίπτωση που

δεν είναι εφικτό να υπάρξει συγχώνευση, απλά προσθέτουμε το *PP* στη λίστα *sols_pp*. Η διαδικασία αυτή εξηγείται αναλυτικότερα στο 3.

- Αν δεν περιέχεται στο *PP* η *κορυφή* προορισμού, τότε επεκτείνουμε και τα δύο μονοπάτια προς όλες τις εξερχόμενες ακμές των κορυφών που περιέχουν*(γραμμ. 10,11)*.

- Έτσι δημιουργούνται κάποια νέα *PP's* για τα οποία εκτελούμε *έλεγχο κυριαρχίας(γραμμ. 12)*.

- Έπειτα ελέγχουμε αν καθένα από αυτά τα PP_{new} μπορεί να συγχωνευθεί με κάποιο PP το οποίο περιέχει την ίδια κορυφή και υπάρχει ήδη στην OpenList*(γραμμ. 14)*. Στην περίπτωση αυτή ακολουθείται η ίδια διαδικασία όπως και στην περίπτωση εισαγωγής PP στην λίστα solutions_pp. Και αυτή η διαδικασία θα επεξηγηθεί πιο αναλυτικά σε παρακάτω κουκίδα.

- Ο αλγόριθμος τερματίζει όταν δεν θα υπάρχουν πια άλλα ζεύγη μονοπατιών στην OpenList.

- Έπειτα από τον τερματισμό του αλγορίθμου δημιουργούμε μια λίστα solutions στην οποία αποθηκεύουμε όλα τα *top-left* μονοπάτια του συνόλου των ζευγών μονοπατιών που ανήκουν στην λίστα solutions_pp. Τα μονοπάτια αυτά που υπάρχουν στην λίστα solutions είναι το *κατά προσέγγιση μερικό Pareto μέτωπο(γραμμ. 16,17).*

- Ο αλγόριθμος επιστρέφει τη λίστα solutions που, όπως αναφέραμε προηγουμένως, περιέχει τα μονοπάτια αυτά που αποτελούν το προσεγγιστικό μερικό κατά *Pareto μέτωπο.*

1) Ένα ζεύγος μονοπατιών επεκτείνεται ξεχωριστά για κάθε ένα από τα δύο μονοπάτια π_v^{tl} και π_v^{br} που ανήκουν στο PP_v και δημιουργείται έτσι ένα νέο ζεύγος μονοπατιών PP_{new}. Τα π_v^{tl} και π_v^{br} επεκτείνονται με τον τρόπο που εξηγήσαμε αναλυτικά στον αλγόριθμο BOA*.

2) Ο *έλεγχος κυριαρχίας* γίνεται με παρόμοιο τρόπο όπως στον αλγόριθμο BOA*. Υπενθυμίζουμε ότι υπάρχουν δύο τύποι ελέγχων κυριαρχίας που θέλουμε να πραγματοποιήσουμε.

a) Ο πρώτος έλεγχος εξετάζει εάν ένας κόμβος κυριαρχείται από έναν κόμβο που έχει ήδη επεκταθεί. Συγκρίνουμε την g_2 *τιμή* του *bottom-right* μονοπατιού με την g_2^{min} της κορυφής που περιέχεται στο μονοπάτι αυτό $c_2(\pi_v^{br}) \geq g_2^{min}(v)$. Αυτός ο έλεγχος κυριαρχίας είναι ορθός καθώς οι ευρετικές μας συναρτήσεις είναι συνεπείς και εξαιτίας της λεξικογραφικής ταξινόμησης του δυαδικού σωρού *OpenList*.

b) Ο δεύτερος έλεγχος εξετάζει εάν ένας κόμβος έχει τη δυνατότητα να φτάσει στον στόχο με μια λύση της οποίας το κόστος δεν κυριαρχείται από οποιαδήποτε υπάρχουσα λύση. Για τον λόγο αυτό συγκρίνουμε επίσης και την f_2 *τιμή* του *bottom-right* μονοπατιού με την g_2^{min} τιμή της *κορυφής στόχου*. $f_2(\pi_v^{br}) \geq g_2^{min}(v_{goal})$. Σε ένα ζεύγος μονοπατιών $PP_v = (\pi_v^{tl}, \pi_v^{br})$ *οι f-τιμές* του μας δείχνουν το κατώτερο όριο του μερικού κατά Pareto μετώπου που μπορεί να επιτευχθεί μέσω του PP_v. Η $g_2^{min}(v_{goal})$ τιμή μας δείχνει την ελάχιστη τιμή κόστους του δεύτερου κριτηρίου από τα μονοπάτια που ανήκουν στο τωρινό μερικό κατά Pareto μέτωπο. Εξαιτίας της λεξικογραφικής ταξινόμησης δίνουμε προτεραιότητα στα κόστη του πρώτου κριτηρίου, οπότε η σύγκριση δεν είναι απαραίτητη. Επομένως αν ο έλεγχος είναι αληθής το PP_v κυριαρχείται.

c) Αν κάποια από τις δύο g_2 ή f_2 τιμές είναι μεγαλύτερη η ίση από τις τιμές που τις συγκρίναμε αντίστοιχα, λέμε πως το PP_v κυριαρχείται. Οπότε είτε το PP_v είτε δεν *επεκτείνεται*(Εικ.3.7.3.1, *γραμμ. 5*), είτε δεν εισέρχεται στην *OpenList* *(γραμμ. 15)*, αναλόγως σε ποιο σημείο της επανάληψης λαμβάνει χώρα ο έλεγχος κυριαρχίας.

3) Η διαδικασία εισόδου ενός PP_{target} στην λίστα πιθανών λύσεων *solutions_pp* έχει ως εξής : Για καθένα από τα PP'_{target} τα οποία ήδη υπάρχουν στη λίστα *(γραμμ. 1)*, επαναληπτικά συγχωνεύουμε αυτά τα δύο ζεύγη μονοπατιών με τον τρόπο που εξηγήσαμε παραπάνω *(γραμμ. 2)* και ελέγχουμε αν το νέο ζεύγος μονοπατιών PP_{target}^{merged} που έχει προκύψει από την συγχώνευση είναι *$(\varepsilon_1, \varepsilon_2)$-φραγμένο* *(γραμμ.*

Input: ($\text{PP}_{\text{v}_{\text{goal}}}$, solutions_pp)
1: **for each** path pair $\tilde{\text{PP}}_{\text{v}_{\text{goal}}} \in$ solutions_pp **do**
2: $\text{PP}_{\text{v}_{\text{goal}}}^{\text{merged}} \leftarrow$ merge($\text{PP}_{\text{v}_{\text{goal}}}, \text{PP}_{\text{v}}$)
3: **if** $\text{PP}_{\text{v}_{\text{goal}}}^{\text{merged}}$.is_bounded($\varepsilon_1, \varepsilon_2$) **then**
4: solutions_pp.remove($\tilde{\text{PP}}_{\text{v}_{\text{goal}}}$)
5: solutions_pp.insert($\text{PP}_{\text{v}_{\text{goal}}}^{\text{merged}}$)
6: **return**
7: solutions_pp.insert($\text{PP}_{\text{v}_{\text{goal}}}$)
8: **return**

Εικόνα 3.5.3.2 *Ψευδοκώδικας για την εισαγωγή ενός PP στην solution_pp λίστα*

3). Αν το PP_{target}^{merged} που δημιουργείται είναι φραγμένο, τότε η επανάληψη τερματίζει και το PP_{target}^{merged} προστίθεται στην λίστα *solutions_pp* με τις πιθανές λύσεις *(γραμμ. 5)* και αφαιρείται από τη λίστα *solutions_pp* το PP'_{target} με το οποίο έγινε η συγχώνευση *(γραμμ. 4)* και προέκυψε το οριοθετημένο ζεύγος μονοπατιών PP_{target}^{merged} . Αν κανένα από τα PP_{target}^{merged} δεν μπορεί να οροθετηθεί, απλά προστίθεται στην λίστα *solutions_pp* το PP_{target} *(γραμμ. 7)*.

4) Η διαδικασία εισόδου ενός PP_v στην OpenList είναι παρόμοια με την παραπάνω διαδικασία. Η μόνη διαφορά είναι πως στην περίπτωση αυτή συγχωνεύουμε επαναληπτικά το PP_v που έχει δημιουργηθεί μετά την επέκταση ενός κόμβου με τα PP'_v που ανήκουν στην OpenList και περιέχουν την ίδια κορυφή v*(γραμμ. 2)*. Έπειτα και εδώ ελέγχουμε αν το PP_v^{merged} είναι φραγμένο*(γραμμ. 3)*. Αν το PP_v^{merged} ζεύγος μονοπατιών *είναι* *$(\varepsilon_1,\varepsilon_2)$-φραγμένο* τερματίζουμε την επανάληψη, προσθέτουμε το ζεύγος μονοπατιών στην OpenList*(γραμμ. 5)* και αφαιρούμε το PP'_v από αυτή*(γραμμ. 4)*. Δηλαδή αφαιρούμε το ζεύγος μονοπατιών το οποίο υπήρχε στην OpenList και με το οποίο το PP_v συγχωνεύθηκε και δημιουργήθηκε το *$(\varepsilon_1,\varepsilon_2)$-φραγμένο* PP_v^{merged} ζεύγος μονοπατιών. Στην περίπτωση που κανένα από τα συγχωνευμένα ζεύγη μονοπατιών δεν φράσσεται, προσθέτουμε στην λίστα απλώς το PP_v *(γραμμ. 7)*.

```
Input: (PP_v, OPEN)
1: for each path pair P̃P_v ∈ OPEN do
2:   PP_v^merged ← merge(P̃P_v, PP_v)
3:   if PP_v^merged.is_bounded(ε1, ε2) then
4:     OPEN.remove(P̃P_v)        ▷ remove existing path pair
5:     OPEN.insert(PP_v^merged)
6:     return
7: OPEN.insert(PP_v)
8: return
```

Εικόνα 3.5.3.3 *Ψευδοκώδικας για την εισαγωγή ενός PP στην OpenList*

Κεφάλαιο 4. Πειραματική αξιολόγηση

Κατά την πειραματική αξιολόγηση συγκρίνουμε τους αλγορίθμους BOA*$_\varepsilon$ και PPA*, προκειμένου να διαπιστώσουμε ποιος αλγόριθμος αποδίδει καλύτερα, σε χρόνο, και σε ποιες περιπτώσεις. Η πειραματική αξιολόγηση έλαβε χώρα σε ένα μηχάνημα 2.4GHz Intel(R) Core™ με 8GB Ram, σε Ubuntu(20.04) λειτουργικό σύστημα. Η μεταγλώττιση έγινε σε GCC-compiler. Η υλοποίηση των αλγορίθμων έγινε στη γλώσσα προγραμματισμού C. Ως σύνολα δεδομένων τα οποία λαμβάνουν ως είσοδο οι υλοποιήσεις μας, χρησιμοποιήσαμε οδικούς χάρτες από το 9'th DIMACS Implementation Challenge: Shortest Path.

Τα κόστη c_1 και c_2 αντιπροσωπεύουν τις αποστάσεις μεταξύ των κορυφών του γραφήματος και τον χρόνο που χρειάζεται προκειμένου να τις διανύσουμε, αντίστοιχα. Οι ευρετικές παράμετροι h_1, h_2 αντιπροσωπεύουν την ελάχιστη απόσταση και χρόνο από την κορυφή στόχο προς κάθε άλλη κορυφή. Επίσης οι *παράγοντες προσέγγισης* $\varepsilon_1, \varepsilon_2$ και ε λαμβάνουν τις ίδιες τιμές σε κάθε πείραμα *(*$\varepsilon_1 = \varepsilon_2 = \varepsilon$, $\varepsilon \in \{0, 0.01, 0.025, 0.5, 0.1\}$ *)*. Οι αλγόριθμοι που υλοποιήσαμε δέχονται τις ίδιες εισόδους και υπολογίζουν τις ίδιες ευρετικές αποστάσεις. Για τον λόγο αυτό στους χρόνους των πειραμάτων δεν προσμετράμε τον χρόνο που χρειάζεται προκειμένου να δημιουργηθεί το γράφημα, με δομή συνδεδεμένων λιστών, όπως και τον χρόνο που χρειάζεται προκειμένου να υπολογιστούν οι ευρετικές παράμετροι. Το άθροισμα των δύο αυτών χρόνων είναι περίπου 0,6 δευτερόλεπτα.

4.1 New York experiment

Παρακάτω παρουσιάζονται τα αποτελέσματα, που έδωσαν ως έξοδο οι αλγόριθμοι που υλοποιήσαμε για το σύνολο δεδομένων του οδικού δικτύου της Νέας Υόρκης. Το δίκτυο αυτό απαρτίζεται από *264.346* κόμβους και *733.846* ακμές. Οι τιμές που αφορούν τον χρόνο έχουν ως μονάδα μέτρησης τα δευτερόλεπτα. Παρακάτω παρουσιάζονται δύο πίνακες. Ο πρώτος αφορά τους μέσους όρους των τιμών που αναγράφονται σε αυτών, ενώ ο δεύτερος τις ενδιάμεσες τιμές.

Διαπιστώνουμε πως ο BOA* είναι πιο αποδοτικός από τον PPA* όσον αφορά τον μέσο όρο των τιμών (***πίνακας 4.1.1***), όταν πρέπει να παράγουμε ολόκληρο το *κατά Pareto βέλτιστο μέτωπο*, δηλαδή ο παράγοντας προσέγγισης ε έχει την τιμή μηδέν. Αυτό μπορεί να θεωρηθεί ως αναμενόμενο, καθώς όπως περιγράψαμε, ο αλγόριθμος PPA* αποθηκεύει για κάθε στοιχείο στην ουρά προτεραιότητας δύο διαδρομές και εκτελεί λειτουργίες οι οποίες είναι πιο απαιτητικές υπολογιστικά *(π.χ. merging)*. Όμως, όσο η τιμή του ε αυξάνεται (γίνεται μεγαλύτερη του μηδέν), τόσο πιο αποδοτικός γίνεται ο PPA*. Μάλιστα παρουσιάζει καλύτερες τιμές από τον BOA* για όλες τις τιμές που είναι μεγαλύτερες του μηδενός.

	$BOA^*_\varepsilon\ num_{avg}\ sols$	$PPA^*\ num_{avg}\ sols$	$BOA^*_\varepsilon\ time_{avg}$	$PPA^*\ time_{avg}$
ε = 0	111,692308	111,692308	3,57069331	15,2637937
ε = 0,01	18,8461538	17,1153846	2,66975254	2,5104545
ε = 0,025	9,73076923	9,03846154	1,79358408	1,254187
ε = 0.05	5,57692308	5,23076923	1,01408985	0,63659258
ε = 0.1	3,34615385	3,26923077	0,50088681	0,38264635

Πίνακας 4.1.1 *Μέσοι όροι αριθμού λύσεων και χρόνου εκτέλεσης για BOA*$_\varepsilon$ *και PPA*.*

Στις median τιμές (***πίνακας 4.1.2***) ωστόσο, παρατηρείται διαφορετική συμπεριφορά μεταξύ των δύο αλγορίθμων, σε σχέση τους μέσους όρους. Παρατηρούμε πως ο χρόνος εκτέλεσης του BOA* είναι καλύτερος για την ενδιάμεση τιμή από αυτόν του αλγορίθμου PPA*, εωσότου η τιμή του παράγοντα προσέγγισης ε να γίνει μεγαλύτερη ή ίση του 0,05. Έως τότε ο BOA* αλγόριθμος εμφανίζει καλύτερη απόδοση όσον αφορά τον χρόνο.

	BOA^*_ε num_{med} sols	PPA^* num_{med} sols	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
ε = 0	89	89	0,788616	3,042729
ε = 0,01	20	17	0,407293	0,549803
ε = 0,025	10	9	0,319734	0,342581
ε = 0.05	6	5	0,231693	0,174581
ε = 0.1	3	3	0,164293	0,094403

Πίνακας 4.1.2 *Διάμεσος του αριθμού των λύσεων για τους BOA* και PPA*.*

Οπότε, προκειμένου να βγάλουμε πιο ασφαλή συμπεράσματα θα χωρίσουμε τα αποτελέσματα μας, με βάση τον αριθμό λύσεων, σε τρία σύνολα. Το ένα σύνολο θα περιέχει το *25%* των μικρότερων τιμών, το δεύτερο σύνολο το *25%* των μεγαλύτερων τιμών και το τρίτο σύνολο τις τιμές που κυμαίνονται μεταξύ των δύο συνόλων *25%-75%.* Τα αποτελέσματα αυτά περιέχονται στους πίνακες ***4.1.3***, ***4.1.5*** και ***4.1.4*** αντίστοιχα.

Ο πίνακας *4.1.4* θα πρέπει να θεωρείται ως το πιο αντιπροσωπευτικό δείγμα από τα σύνολα των λύσεων, καθώς δεν περιέχει ούτε τις ακραία μικρές, ούτε τις ακραία μεγάλες λύσεις, του ολικού συνόλου. Έτσι παρατηρούμε πως ο αλγόριθμος PPA* παρουσιάζει καλύτερη απόδοση από τον BOA* για όλες τις τιμές όπου ισχύει : *ε>0.*

Πολύ ενδιαφέρουσα περίπτωση είναι αυτή του συνόλου των ελαχίστων τιμών *(πίνακας 4.1.3)* όπου ο αλγόριθμος BOA* παρουσιάζει καλύτερο χρόνο για όλες τις τιμές του *ε.*

Τέλος στο γράφημα των μεγίστων τιμών παρατηρούμε εναλλαγές στα αποτελέσματα των δύο αλγορίθμων, καθώς αυξάνεται η τιμή του παράγοντα προσέγγισης *ε*, γεγονός που δεν μας επιτρέπει να βγάλουμε ασφαλή συμπεράσματα.

ε	BOA^*_ε num_{avg} sols	PPA^* num_{avg} sols	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} sols	PPA^* num_{med} sols	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
ε=0	23,714	23,714	0,0277	0,1347	12	12	0,0031	0,0076
ε=0.01	8	7,2857	0,2769	0,3240	7	7	0,0021	0,0056
ε=0.025	4,57	4,1429	0,1559	0,1627	5	5	0,0015	0,0031
ε=0.05	2,571	2,5714	0,092	0,0820	3	3	0,0010	0,0020
ε=0.1	1,857	1,7142	0,0715	0,1199	2	2	0,0009	0,0015

Πίνακας4.1.3 *Τα αποτελέσματα του minimum 25% συνόλου.*

ε	BOA^*_ε num_{avg} sols	PPA^* num_{avg} sols	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} sols	PPA^* num_{med} sols	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
$\varepsilon=0$	96,769	96,769	2,4937	8,8156	89	89	0,7886	3,0427
$\varepsilon=0.01$	19,769	17,769	2,8506	2,62	20	17	0,5807	0,5498
$\varepsilon=0.025$	10,308	9,538	2,3764	1,6986	10	9	0,9467	0,4939
$\varepsilon=0.05$	5,846	5,4615	1,3265	0,8545	6	5	0,5768	0,2657
$\varepsilon=0.1$	3,461	3,3846	0,759	0,4736	3	3	0,8163	0,1459

Πίνακας 4.1.4 *Τα αποτελέσματα του 25-75% συνόλου τιμών.*

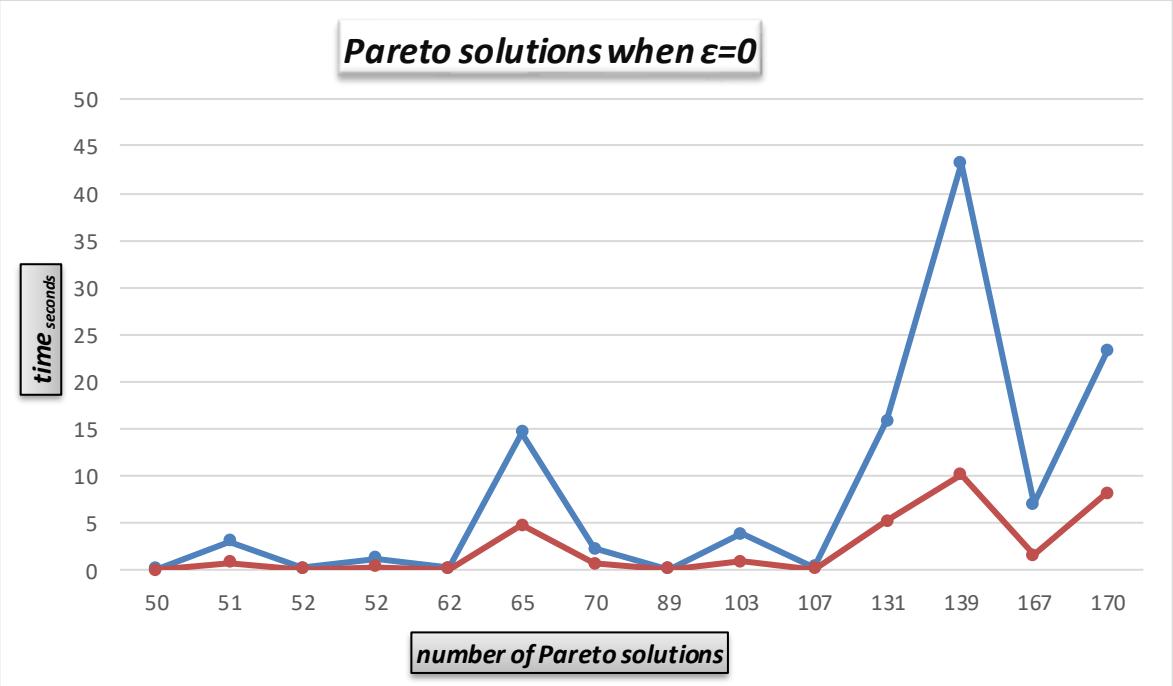
ε	BOA^*_ε num_{avg} sols	PPA^* num_{avg} sols	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} sols	PPA^* num_{med} sols	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
$\varepsilon=0$	246,667	246,667	10,0375	46,8853	209	209	11,335	38,317
$\varepsilon=0.01$	29,5	27,1667	5,0695	4,8239	29	27	2,9767	1,9325
$\varepsilon=0.025$	14,5	13,6667	2,4415	1,5648	14	14	0,5482	0,7506
$\varepsilon=0.05$	8,5	7,8333	1,4129	0,8116	9	5	0,4491	0,3635
$\varepsilon=0.1$	4,8334	4,8334	0,4424	0,4920	5	3	0,0196	0,0451

Πίνακας 4.1.5 *Τα αποτελέσματα του 25% maximum συνόλου.*

Το παρακάτω *γράφημα (4.1.1)* μας δείχνει την διαφορά στο χρόνο εκτέλεσης μεταξύ των αλγορίθμων BOA* και PPA* , όταν ο παράγοντας προσέγγισης *ε* έχει την τιμή μηδέν *(ε=0)*. Το γράφημα αντιστοιχίζει επίσης και τον αριθμό των λύσεων που ανήκουν στο *κατά Pareto βέλτιστο μέτωπο* με τον χρόνο εκτέλεσης. Δηλαδή, όσο πιο δεξιά είναι μια κουκίδα στο γράφημα τόσες περισσότερες είναι οι λύσεις που ανήκουν στο κατά Pareto μέτωπο μέσα σε συγκεκριμένο χρόνο. Κάνουμε έτσι την απλή διαπίστωση πως οι δύο αλγόριθμοι παρήγαν τις ίδιες λύσεις σε κάθε εκτέλεση τους, αλλά με τον BOA* να είναι σαφώς πιο αποδοτικός. Το σύνολο των τιμών προέκυψε μετά από 26 εκτελέσεις των αλγορίθμων με τυχαίες τιμές και οι τιμές που αναγράφονται στο γράφημα είναι το σύνολο των τιμών που ανήκουν στο σύνολο 25%-75% του κατά Pareto βέλτιστου

συνόλου, το οποίο αφορά τον αριθμό των λύσεων που ανήκουν στο κατά Pareto βέλτιστο μέτωπο. Δηλαδή στο γράφημα αυτό δεν περιέχεται το 25% των μικρότερων αριθμών των λύσεων καθώς και το 25% των μεγαλύτερων αριθμών των λύσεων.

Ο κύριος λόγος που παραθέτουμε αυτό το γράφημα είναι για να δείξουμε πως ο αριθμός των λύσεων δεν είναι σε αναλογία με τον χρόνο εκτέλεσης των αλγορίθμων, καθώς το γράφημα παρουσιάζει πολλές καμπύλες. Άρα η αύξηση του χρόνου εκτέλεσης δεν οφείλεται στην αύξηση του μεγέθους του κατά Pareto βέλτιστου μετώπου, αλλά στην αύξηση των επαναλήψεων των αλγορίθμων, που οφείλονται στην αύξηση των αλμάτων προκειμένου να φτάσουμε από την κορυφή εκκίνησης στην κορυφή προορισμού.



Γράφημα 4.1.1 *Σύγκριση χρόνου εκτέλεσης για BOA (κόκκινη γραμμή)* και PPA*(μπλε γραμμή) για ε =0.*

4.2 San Francisco Bay experiment

Στο πείραμα αυτό, δίνουμε ως είσοδο στους αλγορίθμους μας ένα μεγαλύτερο σύνολο δεδομένων. Ο χάρτης του οδικού δικτύου του San Francisco BAY περιέχει περίπου 50.000 περισσότερους κόμβους και 70.000 περισσότερες ακμές από αυτόν της Νέας Υόρκης. Περιέχει λοιπόν ακριβώς 321.270 κόμβους και 800.172 ακμές. Οπότε θα ήταν ενδιαφέρον να δούμε την συμπεριφορά των αλγορίθμων σε ένα μεγαλύτερο δίκτυο.

Προκειμένου να λάβουμε τα αποτελέσματα που παρουσιάζονται στους πίνακες έγιναν 32 εκτελέσεις των αλγορίθμων με τυχαίες τιμές για τις κορυφές εκκίνησης και στόχου.

Παρακάτω παραθέτουμε ακριβώς τους ίδιους πίνακες όπως και στο πείραμα της Νέας Υόρκης. Έτσι λοιπόν οι πίνακες 4.2.1 και 4.2.2 αφορούν ολόκληρο το σύνολο των αποτελεσμάτων ενώ, οι πίνακες 4.2.3 , 4.2.4 και 4.2.5 αφορούν τα ίδια υποσύνολα που παραθέσαμε στο παραπάνω πείραμα.

Παρατηρούμε πως οι δύο αλγόριθμοι παρουσιάζουν κοινές συμπεριφορές και στα δύο πειράματα. Όσον αφορά τους μέσους όρους των χρόνων εκτελέσεων και στο ολικό σύνολο, αλλά και στο σύνολο που περιέχει το 25%-75% των αποτελεσμάτων, το οποίο είναι το πιο ασφαλές για την εξαγωγή συμπερασμάτων, καθώς δεν περιέχει τις ακραίες τιμές, ο BOA* είναι πιο αποδοτικός από τον PPA* για όλες τις περιπτώσεις όπου ο παράγοντας προσέγγισης ε λαμβάνει τιμές μεγαλύτερες του μηδενός. Ωστόσο για τις συγκεκριμένες περιπτώσεις ο PPA* παρουσιάζει καλύτερη απόδοση χρόνου και για τις ενδιάμεσες τιμές. Επίσης αισθητά καλύτερη απόδοση χρόνου του PPA* παρουσιάζεται και στις ακραίες τιμές των συνόλων.

	$BOA*_{\varepsilon}\ num_{avg}\ sols$	$PPA*\ num_{avg}\ sols$	$BOA*_{\varepsilon}\ time_{avg}$	$PPA*\ time_{avg}$
ε = 0	102,96875	102,96875	1,43526447	4,74486609
ε = 0,01	16,03125	14,65625	0,94130628	0,85819318
ε = 0,025	8,53125	7,96875	0,65780625	0,40588663
ε = 0.05	5,0625	4,8125	0,39942363	0,18664719
ε = 0.1	3,28125	3,09375	0,23536547	0,11266763

Πίνακας 4.2.1 *Μέσοι όροι αριθμού λύσεων και χρόνου εκτέλεσης για BOA*ε και PPA*.*

Πίνακας 4.2.2 *Διάμεσος του αριθμού των λύσεων για τους BOA* και PPA*.*

	$BOA*_{\varepsilon}\ num_{med}\ sols$	$PPA*\ num_{med}\ sols$	$BOA*_{\varepsilon}\ time_{med}$	$PPA*\ time_{med}$
ε = 0	85	85	0,2029775	1,052069
ε = 0,01	16	15	0,129637	0,127844
ε = 0,025	9	8	0,090124	0,071713
ε = 0.05	5	5	0,0453845	0,037781
ε = 0.1	3	3	0,029711	0,0217945

ε	BOA^*_ε num_{avg} $sols$	PPA^* num_{avg} $sols$	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} $sols$	PPA^* num_{med} $sols$	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
$\varepsilon=0$	21,75	21,75	0,0238	0,0944	20	20	0,0084	0,0256
$\varepsilon=0.01$	7,25	6,375	0,9729	0,0154	7	7	0,3257	0,0093
$\varepsilon=0.025$	5	4,75	0,01879	0,0174	6	5	0,0046	0,0148
$\varepsilon=0.05$	3,25	3,125	0,058	0,0683	4	3	0,0044	0,0147
$\varepsilon=0.1$	2,125	1,875	0,1064	0,0952	2	2	0,0087	0,0081

Πίνακας4.2.3 *Τα αποτελέσματα του minimum 25% συνόλου.*

ε	BOA^*_ε num_{avg} $sols$	PPA^* num_{avg} $sols$	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} $sols$	PPA^* num_{med} $sols$	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
$\varepsilon=0$	79,75	79,75	1,1762	3,7933	85	85	0,3744	1,66
$\varepsilon=0.01$	16,125	14,625	0,8025	0,6650	16	15	0,1067	0,1922
$\varepsilon=0.025$	8,4375	7,75	0,5047	0,3604	9	8	0,1435	0,0856
$\varepsilon=0.05$	5	4,625	0,28	0,1512	5	5	0,0702	0,0378
$\varepsilon=0.1$	3,125	2,9375	0,1448	0,0405	3	3	0,0099	0,0186

Πίνακας 4.2.4 *Τα αποτελέσματα του 25-75% συνόλου τιμών.*

ε	BOA^*_ε num_{avg} $sols$	PPA^* num_{avg} $sols$	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} $sols$	PPA^* num_{med} $sols$	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
$\varepsilon=0$	230,625	230,625	3,365	3,7402	228	228	0,9694	3,7402
$\varepsilon=0.01$	24,625	23	1,1873	2,0871	24	23	0,0349	0,3861
$\varepsilon=0.025$	12,25	11,625	1,603	0,8851	13	11	0,316	0,2322
$\varepsilon=0.05$	7	6,875	0,9797	0,3758	7	7	0,2441	0,0727
$\varepsilon=0.1$	4,75	4,625	0,5454	0,2745	5	5	0,1121	0,0517

Πίνακας 4.2.5 *Τα αποτελέσματα του 25% maximum συνόλου.*

4.3 Colorado experiment

Αυτό το πείραμα έλαβε χώρα σε ένα ακόμη μεγαλύτερο dataset, αντιπροσωπευτικό του χάρτη οδικού δικτύου της πολιτείας του Colorado. Το σύνολο δεδομένων αυτό αποτελείται από 435.666 κόμβους και 1.057.066 ακμές, που απαρτίζουν το συνδεδεμένο, κατευθυνόμενο γράφημα. Σε αυτό το πείραμα έγινε εκτέλεση των δύο αλγορίθμων 32 φορές και τα αποτελέσματα των εκτελέσεων αναγράφονται στους παρακάτω πίνακες. Όπως και στα δύο προηγούμενα πειράματα υπάρχουν δύο πίνακες που αφορούν το ολικό σύνολο των αποτελεσμάτων. Αυτοί είναι ο πίνακας 4.3.1 που αφορά τους μέσους όρους των αποτελεσμάτων και ο 4.3.2 που αφορά της ενδιάμεσες τιμές. Οι επόμενοι τρεις πίνακες 4.3.3, 4.3.4, 4.3.5, αφορούν τα σύνολα min-25%, 25%-75% και max-25% υποσύνολα αντίστοιχα.

Πίνακας 4.3.1 *Μέσοι όροι αριθμού λύσεων και χρόνου εκτέλεσης για BOA*ε και PPA*.*

	$BOA^*_\varepsilon\ num_{avg}\ sols$	$PPA^*\ num_{avg}\ sols$	$BOA^*_\varepsilon\ time_{avg}$	$PPA^*\ time_{avg}$
ε = 0	156,78125	156,78125	2,47080513	6,65275869
ε = 0,01	12,5625	11,59375	1,78094569	0,5678005
ε = 0,025	6,4375	6,09375	1,49749791	0,22653209
ε = 0.05	4	3,6875	1,34023959	0,12884669
ε = 0.1	2,46875	2,375	0,65508278	0,04999541

Πίνακας 4.3.2 *Διάμεσος του αριθμού των λύσεων για τους BOA* και PPA*.*

	$BOA^*_\varepsilon\ num_{med}\ sols$	$PPA^*\ num_{med}\ sols$	$BOA^*_\varepsilon\ time_{med}$	$PPA^*\ time_{med}$
ε = 0	92	92	0,1576375	0,582484
ε = 0,01	12	11	0,085623	0,052111
ε = 0,025	6	6	0,059316	0,024886
ε = 0.05	4	4	0,0296135	0,0173705
ε = 0.1	2	2	0,0232195	0,0072895

ε	BOA^*_ε num_{avg} sols	PPA^* num_{avg} sols	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} sols	PPA^* num_{med} sols	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
ε=0	12,125	12,125	0,00217	0,0071	14	14	0,00179	0,0031
ε=0.01	3,625	3,5	0,00163	0,00228	2	2	0,0014	0,00188
ε=0.025	2,375	2,25	0,00145	0,00178	2	2	0,00122	0,00138
ε=0.05	1,5	1,5	0,00117	0,00139	2	2	0,00108	0,00127
ε=0.1	1,25	1,25	0,00108	0,00124	1	1	0,00106	0,00118

Πίνακας4.3.3 *Τα αποτελέσματα του minimum 25% συνόλου.*

ε	BOA^*_ε num_{avg} sols	PPA^* num_{avg} sols	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} sols	PPA^* num_{med} sols	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
ε=0	108,562	108,5625	0,498	1,4158	92	92	0,1576	0,5824
ε=0.01	12	11,3125	0,259	0,1376	12	11	0,0856	0,0521
ε=0.025	6,1875	5,875	0,177	0,0681	6	6	0,0593	0,0249
ε=0.05	4,0625	3,6875	0,1063	0,0381	4	4	0,0296	0,0174
ε=0.1	2,25	2,25	0,035	0,0179	2	2	0,0232	0,0073

Πίνακας 4.3.4 *Τα αποτελέσματα του 25-75% συνόλου τιμών.*

ε	BOA^*_ε num_{avg} sols	PPA^* num_{avg} sols	BOA^*_ε $time_{avg}$	PPA^* $time_{avg}$	BOA^*_ε num_{med} sols	PPA^* num_{med} sols	BOA^*_ε $time_{med}$	PPA^* $time_{med}$
ε=0	397,875	397,875	8,885	23,7724	344	144	7,239	18,321
ε=0.01	22,625	20,25	6,6043	1,9938	22	19	4,900	1,3003
ε=0.025	11	10,375	5,6346	0,76821	12	10	4,5821	0,4899
ε=0.05	6,375	5,875	5,1471	0,4378	6	6	3,8813	0,2163
ε=0.1	4,125	3,75	2,5493	0,16357	4	4	1,5182	0,1205

Πίνακας 4.3.5 *Τα αποτελέσματα του 25% maximum συνόλου.*

Όπως διαπιστώνουμε από τους παραπάνω πίνακες και σε αυτή την περίπτωση ο αλγόριθμος BOA* παρουσιάζει καλύτερη απόδοση από τον PPA* όσον αφορά την εύρεση ολόκληρου του κατά Pareto βέλτιστου μετώπου. Ο PPA* όμως και σε αυτή την περίπτωση παρουσιάζει καλύτερη απόδοση στο ολικό σύνολο, αλλά και στο σύνολο των ενδιάμεσων (25%-75%) αποτελεσμάτων, το οποίο μας οδηγεί στα πιο ασφαλή συμπεράσματα. Στην περίπτωση αυτή ωστόσο παρατηρούμε πολύ καλύτερη απόδοση του PPA* στις ακραίες μεγάλες τιμές των αλγορίθμων. Με εξαίρεση την περίπτωση όπου ο παράγοντας προσέγγισης *ε* έχει την τιμή μηδέν, ο PPA* παρουσιάζει συντριπτικά καλύτερα αποτελέσματα σε σχέση με τον BOA*. Σε αντίθεση με τις μικρότερες τιμές όπου ο BOA* εμφανίζει ελαφρώς καλύτερη απόδοση .

4.4 Συμπεράσματα

Στο κεφάλαιο αυτό παραθέτουμε τα συμπεράσματα που εξήγαμε κατά την εκτέλεση των τριών πειραμάτων της Νέας Υόρκης, του Σαν Φρανσίσκο και του Κολοράντο. Τα πειράματα ονομάστηκαν έτσι με βάση τα σύνολα των δεδομένων που χρησιμοποιήθηκαν σε αυτά.

Το πρώτο αδιαμφισβήτητο με βάση τα πειράματα συμπέρασμα, είναι πως αν επιθυμούμε την εύρεση ολόκληρου του κατά Pareto βέλτιστου μετώπου, δηλαδή ο παράγοντας προσέγγισης *ε* λαμβάνει την τιμή μηδέν, ο αλγόριθμος BOA* εμφανίζει ισχυρά καλύτερη απόδοση σε σχέση με τον PPA*. Αυτό συμβαίνει επειδή για την εύρεση ολόκληρου του κατά Pareto βέλτιστου μετώπου οι δύο αλγόριθμοι επεκτείνουν τους κόμβους με πολύ όμοιο τρόπο. Η βασική διαφορά είναι πως ο PPA* περιέχει στους κόμβους του δύο μονοπάτια και όχι ένα όπως ο BOA*, αλλά περιέχει και πιο χρονοβόρες λειτουργίες, όπως για παράδειγμα η συγχώνευση.

Ένα δεύτερο συμπέρασμα είναι πως στην γενική περίπτωση για την εύρεση του κατά προσέγγιση Pareto μετώπου, δηλαδή ο παράγοντας προσέγγισης *ε* λαμβάνει τιμές μεγαλύτερες του μηδενός, ο αλγόριθμος PPA* παρουσιάζει καλύτερη απόδοση από τον BOA*. Εξήγαμε αυτό το συμπέρασμα, καθώς στο σύνολο που δεν περιέχει τις ακραίες περιπτώσεις, στο 25%-75% σύνολο, ο αλγόριθμος PPA* παρουσιάζει καλύτερη απόδοση από τον BOA*. Αυτό μας δείχνει πως οι λειτουργίες του PPA* ***(ε₁,ε₂-φράγμα, συγχώνευση)*** έχουν την ικανότητα να κλαδεύουν μονοπάτια με πιο αποδοτικό τρόπο, απ' τον απλό έλεγχο κυριαρχίας που συμβαίνει κατά την εκτέλεση του BOA*.

Στις ακραία μικρές τιμές βλέπουμε πως οι επιδόσεις τον δύο αλγορίθμων είναι πολύ ανταγωνιστικές με τον BOA* να υπερισχύει έναντι του PPA* αλγορίθμου. Αυτό ίσως συμβαίνει καθώς αν ο BOA* βρει γρήγορα μια αποδοτική λύση αυτή θα κυριαρχήσει στις υπάρχουσες ήδη λύσεις, επομένως οι κόμβοι δεν θα επεκταθούν και δεν θα εκτελεστούν περεταίρω λειτουργίες. Ενώ στην περίπτωση αυτή ο PPA* θα εκτελέσει τις λειτουργίες συγχώνευσης και οριοθέτησης πριν απορριφθούν οι κόμβοι.

Αντίθετα στις ακραίες μεγάλες τιμές παρατηρούμε πως ο PPA* κυριαρχεί έναντι του BOA* αλγορίθμου. Επομένως μπορούμε να θεωρήσουμε πως όσο τα σύνολα δεδομένων μεγαλώνουν ο PPA* θα παρουσιάζει ακόμα καλύτερη απόδοση από τον BOA*, όσον αφορά την εύρεση του *κατά προσέγγιση Pareto μετώπου*, καθώς το μέγεθος των κατά προσέγγιση Pareto συνόλων τείνει να αυξάνεται όσο αυξάνεται το μέγεθος του συνόλου δεδομένων που οι αλγόριθμοι εύρεσης δικριτηριακά βέλτιστων διαδρομών δέχονται ως είσοδο. Αυτό συμβαίνει καθώς όπως προ-είπαμε ο αλγόριθμος BOA* απορρίπτει τους κόμβους που υπάρχουν στην λίστα προτεραιότητας μόνο με έλεγχο κυριαρχίας. Επομένως κατά την εκτέλεση των αλγορίθμων όταν δέχονται ως είσοδο μεγάλα dataset ο PPA* αλγόριθμος θα επεκτείνει λιγότερους κόμβους σε σχέση με τον BOA*, εξαετίας της λειτουργίας της συγχώνευσης των μονοπατιών. Χαρακτηριστικό παράδειγμα που επιβεβαιώνει τον ισχυρισμό μας αποτελεί το max25%-σύνολο αποτελεσμάτων του πειράματος του Colorado, όπου ο PPA* αλγόριθμος παρουσιάζει συντριπτικά καλύτερη απόδοση από τον BOA*.

Βιβλιογραφία

[1] Gallo G; Pallotino S.: Shortest path algorithms. In *Annals of Operations Research*, 13, pp.1–79 (1988)

[2] Ulloa, C.H; Yeoh, W., Baier, J.A., Zhang, H., Suazo, L. Koenig, S.: A Simple and Fast Bi-Objective Search Algorithm. In *International Conference on Automated Planning and Scheduling* (ICAPS), *30*(1): 143-151 (2020).

[3] Goldin, B.; Salzman, O.: Approximate Bi-Criteria Search by Efficient Representation of Subsets of the Pareto-Optimal Frontier. In *International Conference on Automated Planning and Scheduling* (ICAPS), *31*(1): 149-158 (2021).

[4] Anders J.V Skriver. Vassiliadis. A classification of Bicriterion Shortest Path (BSP) algorithms. In In *Asia-Pacific Journal of Operational Research*, 17. pp. 199-212 (2000).

[5] Hamza Ben Ticha, Nabil Absi, Dominique Feillet, Alain Quilliot: The Steiner bi-objective shortest path problem. In *EURO Journal on Computational Optimization*, 9 (2021).

[6] J.C.N. Climaco, E.Q.V. Martins: A bicriterion shortest path algorithm. In *European Journal of Operational Research*, 11 (4) pp. 399-404 (1982),

[7] E.Q.V. Martins: An algorithm for ranking paths that may contain cycles. In *European Journal of Operational Research*, 18 (1), pp. 123-130 (1984).

[8] F. Huarng, P. Pulat and L. Shih: A computational comparison of some bicriterion shortest path algorithms. In *J. Chin. Inst. Ind. Eng.*, 13 (2) pp. 121-125. (1996)

[9] Madhumita Panda, Abinash Mishra: A Survey of Shortest-Path Algorithms. In *International Journal of Applied Engineering Research*, ISSN 0973-4562 Volume 13, Number 9 pp. 6817-6820 (2018).

[10] Guerriero, F. and Musmanno, R.: Label correcting methods to solve multicriteria shortest path problems. In *Journal of Optimization Theory and Applications*, 111:589–613 (2001).

[11] Andrea Raith, Matthias Ehrgott: A comparison of solution strategies for biobjective shortest path problems. In *Computers & Operations Research*, 36(4):1299-1331 (2009).

[12] C.T. Tung and K.L. Chew: A bicriterion Pareto-optimal path algorithm. In *Asia-Pacific Journal of Operational Research*, 5:166–172, (1988)

[13] J.M. Paixao and J.L. Santos: Labelling methods for the general case of the multi-objective shortest path problem - a computational study. Universidade de Coimbra, Technical Report. pp. 07–42 (2007).

[14] A.J.V. Skriver and K.A. Andersen: A label correcting approach for solving bicriterion shortest-path problems. In *Computers & Operations Research*, 27: 507–524 (2000).

[15] F. Guerriero and R. Musmanno: Label correcting methods to solve multicriteria shortest path problems. In *Journal of Optimization Theory and Applications*, 111(3):589–613, (2001).

[16] W.M. Carlyle and R.K. Wood: Near-shortest and k-shortest simple paths. In *Networks*, 46(2):98–109 (2005).

[17] J. Mote, I. Murthy and D.L. Olson: A parametric approach to solving bicriterion shortest path problems. In *European Journal of Operational Research*, 53 (1), pp. 81-92 (1991).

[18] Zadeh L.A.: Optimality and non-scalar-valued performance criteria. In *IEEE Transactions on Automated Control*, AC-8:59–60 (1963).

[19] R. Timothy Marler, Jasbir S. Arora: The weighted sum method for multi-objectiveoptimization: new insights. In *Structural and Multidisciplinary Optimization*, 41(6):853-862 (2010).

[20] Duque, D., Lozano, L., and Medaglia, A.: An exact method for the biobjective shorstest path problem for large-scale road networks. In *European Journal of Operational Research*, (242):788–797 (2015).

[21] Bang Ye Wu, Kun-Mao Chao: Shortest-Paths Trees. Excerpt from the book "*Spanning Trees and Optimization Problems*". Chapman & Hall/CRC Press, USA (2004).

[22] M.Sinthiya, Dr.M. Chidambaram: A Study On Best First Search. In *International Research Journal of Engineering and Technology* (IRJET), Volume 3(6): 588-597 (2016).

[23] Serafini, P.: Some considerations about computational complexity for multi objective combinatorial problems. In *Recent advances and historical development of vector optimization*. Springer, pp. 222–232 (1987).

[24] Kairanbay Magzhan, Hajar Mat Jani: A Review and Evaluations of Shortest Path Algorithms. In *International Journal of Scientific & Technology Research*, 2(6): 99-104 (2013).

[25] Chinchuluun, A., and Pardalos, P. M.: A survey of recent developments in multiobjective optimization. In *Annals of Operations Research*, 154(1):29–50 (2007).

[26] Bachmann, D.; Bokler, F.; Kopec, J.; Popp, K.; Schwarze, B.; and Weichert, F.: Multi-objective optimization-based planning of power-line grid expansions. In ISPRS Int. J. Geo-Inf. 7(7), 258 (2018).

[27] Bronfman, A.; Marianov, V.; Paredes-Belmar, G.; and Luer-Villagra, A.: The maximin HAZMAT routing problem. In *European Journal of Operational Research*, 241(1):15–27. (2015)

[28] Xiang Liu, Daoxiong Gong: A Comparative Study of A-star Algorithms for Search and rescue in Perfect Maze. In *International Conference on Electric Information and Control Engineering*, IEEE (2011).

[29] Hansen P.: Bicriterion path problems. In Fandel G, Gal T, editors, *Multiple criteria decisions making, theory and application* , Lecture Notes in Economics and Mathematical Systems, vol. 177, Springer, p. 109-27 (1980).

[30] Muhammad Adeel Javaid: Understanding Dijkstra's Algorithm. In SSRN Electronic Journal (2013).