

**Μάθημα «Μέθοδοι Βελτιστοποίησης**

**στη Διοικητική Επιστήμη»**

**Ε’ Εξάμηνο**

Report διαγωνιστικής εργασίας

Ομάδα Εργασίας :

Μητροπούλου Μαρία (8180070)

Μπαντουβά Βασιλική (8180073)

Ποταμιάνου Ευσεβία Βασιλική (8180108)

Σακέτου Κωνσταντίνα (8180111)

Χατζηγρηγορίου Μαρία Ελένη (8180150)

<22/01/2021>

Για την επίλυση του προβλήματος της εργασίας χρησιμοποιήσαμε αρχικά τροποποιημένο τον πλεονεκτικό αλγόριθμο ApplyNearestNeighborMethod(), στην συνέχεια εφαρμόσαμε στην λύση –μέσω της Local Search– τροποποιημένους τους αλγόριθμους Swap και TwoOpt, χρησιμοποιώντας τους ελέγχους MoveIsTabu και SetTabu, με αποτέλεσμα να φτάσουμε στην λύση 4.8476190.

Για την κατασκευή του πλεονεκτικού αλγορίθμου, σκεφτήκαμε να χρησιμοποιήσουμε την βασική λογική της ApplyNearestNeighborMethod() τροποποιώντας τα κριτήρια επιλογής της διαδρομής στην οποία κάθε φορά προστίθεται νέος κόμβος. Συνοπτικά: εντοπίζουμε (IdentifyBestInsertion) την καλύτερη εισαγωγή για κάθε μία από τις 25 διαδρομές -χρησιμοποιώντας την λογική της λίστας rcl-, πραγματοποιούμε μια εικονική εφαρμογή της συγκεκριμένης κίνησης (bestInsertion) και αποθηκεύουμε τα χαρακτηριστικά της κίνησης και το «ανανεωμένο» κόστος των δύο διαδρομών που έχουν το μικρότερο «ανανεωμένο» κόστος. Αφού έχουμε κάνει τη παραπάνω διαδικασία και για τα 25 routes, και έχουν προκύψει τα δύο καλύτερα ως προς το κόστος routes, επιλέγουμε στοχαστικά ποια από τις δύο αυτές κινήσεις θα εφαρμοστεί τελικά (ApplyCustomerInsertion). Η διαδικασία επαναλαμβάνεται μέχρι να τοποθετήσουμε όλους τους κόμβους στις 25 διαδρομές.

Στην λύση που προκύπτει από τον παραπάνω αλγόριθμο, εφαρμόζουμε την λογικής της SwapMove, όπου η μόνη τροποποίηση που έχει γίνει (στην μέθοδο FindBestSwapMove) είναι ο έλεγχος πριν την αποθήκευση της κίνησης (StoreBestSwapMove), σχετικά με το εάν τα κόστη των διαδρομών που θα επηρεαστούν από την εφαρμογή της κίνησης υπερβαίνουν το ήδη υπάρχον μέγιστο κόστος της τρέχουσας λύσης. Η διαδικασία τελειώνει όταν δεν βρεθεί moveCost < 0.

Τέλος, στην λύση που παράγεται εφαρμόζουμε τον TwoOpt αλγόριθμο. Στην μέθοδο FindBestTwoOptMove, ακολουθούμε και πάλι την λογική της εικονικής εφαρμογής κινήσεων (ApplyCheck), αποθηκεύουμε τα αναμενόμενα κόστη των διαδρομών που θα επηρεαζόντουσαν αν εφαρμοζόταν η συγκεκριμένη κίνηση, επιλέγουμε την διαδρομή με το μεγαλύτερο αναμενόμενο κόστος (από τις δύο που επιλέγονται σε κάθε for της μεθόδου -rt1,rt2-) –καθώς αυτή θα επηρεάσει (αν επηρεάσει) το τρέχον μέγιστο κόστος της λύσης–, ελέγχουμε αν η κίνηση αποτελεί tabu (MoveIsTabu & MoveIsTabuList) και αν δεν είναι, συνεχίζουμε –βάση καινούριων ελέγχων- στην αποθήκευση της κίνησης. Στη περίπτωση που επηρεάζεται η διαδρομή με το max κόστος της τρέχουσας λύσης, δημιουργούμε μία μεταβλητή (diff) που αποθηκεύει το άθροισμα του cost difference (ανανεωμένο εικονικό κόστος διαδρομής – τρέχον κόστος διαδρομής) και του moveCost σταθμισμένο με συγκεκριμένα βάρη, η οποία πρέπει να είναι μικρότερη του μηδενός και η ελάχιστη μεταξύ των επαναλήψεων της FindBestTwoOptMove, για να αποθηκευτεί η κίνηση ως η καλύτερη μέχρι στιγμής (StoreBestTwoOptMove) (αλλάζει το top.diff ενώ το top.moveCost δεν μεταβάλλεται). Σε κάθε άλλη περίπτωση, όπου η διαδρομή που επηρεάζεται δεν αποτελεί την διαδρομή με το max κόστος της τρέχουσας λύσης, αποθηκεύουμε την κίνηση με έλεγχο μόνο στην τιμή του moveCost (αλλάζει το top.moveCost, το top.diff δεν μεταβάλλεται). Έτσι προκύπτει η καλύτερη κίνηση και εφαρμόζεται η ApplyTwoOptMove μέθοδος. Οι επαναλήψεις τελειώνουν όταν δεν βρεθεί diff και moveCost συγχρόνως μικρότερα του μηδενός.