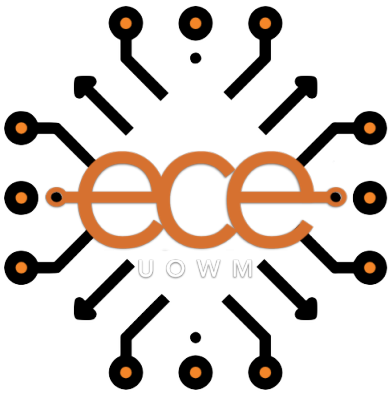


Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αλγόριθμοι συνδυαστικής βελτιστοποίησης για προβλήματα ανάθεσης εργασιών σε περιβάλλον cloud computing

---

Τρυφωνίδου Αικατερίνη (ΑΜ: 1402)

Επιβλέπων Καθηγητής: Νικόλαος Πλόσκας

Εργαστήριο Ευφυών Συστημάτων & Βελτιστοποίησης

26 Φεβρουαρίου 2024

Περίληψη

Τα προβλήματα συνδυαστικής βελτιστοποίησης ανήκουν σε έναν μεγάλο κλάδο της επιχειρησιακής έρευνας και των μαθηματικών που ασχολούνται με την εύρεση βέλτιστων λύσεων μέσα σε ένα πεπερασμένο σύνολο αντικειμένων. Το πρόβλημα που επικεντρώνεται η παρούσα διπλωματική εργασία είναι αυτό της ανάθεσης εργασιών, συγκεκριμένα σε περιβάλλον cloud. Ο καθορισμός του τρόπου ανάθεσης σε ένα περιβάλλον νέφους επηρεάζεται από το πλήθος των εργασιών προς ανάθεση και τα διαθέσιμα στον χρήστη μηχανήματα. Φυσικά, μεγάλο ρόλο έχουν τα μεγέθη των εργασιών, τα οποία διαφέρουν μεταξύ τους, και ο χώρος που διαθέτει κάθε μηχάνημα. Οι αλγόριθμοι που θα χρησιμοποιηθούν για την επίλυση του προβλήματος ανάθεσης βασίζονται στον Ουγγρικό αλγόριθμο, προσαρμόζοντας τον με σκοπό να διαχειρίζεται μη ισορροπημένες περιπτώσεις ανάθεσης. Πέρα από διάφορες εκδοχές του Ουγγρικού αλγορίθμου, στη βιβλιογραφία γίνεται μελέτη ενός στοχαστικού ευρετικού αλγόριθμου, του Dhouib-Matrix-AP2. Όλοι οι αλγόριθμοι αναλύονται και εφαρμόζονται σε ένα σύνολο δεδομένων, ενώ οι επιδόσεις τους συγκρίνονται με τη βέλτιστη λύση που παρέχει ο λύτης γραμμικού προγραμματισμού Gurobi.

Λέξεις κλειδιά: Ανάθεση εργασιών, Ουγγρικός Αλγόριθμος, Μη ισορροπημένη ανάθεση, Υπολογιστή Νέφους, Γραμμικός Προγραμματισμός

Abstract

Combinatorial optimization problems are part of a large branch of operations research and mathematics concerned with finding optimal solutions within a finite set of objects. The problem that the present work focuses on is the problem of task assignment, particularly in a cloud environment. Determining the manner of assignment in a cloud environment is strongly influenced by the number of tasks to be assigned and available to the user machines. Of course, the sizes of the tasks, which may differ from one to another, and the available storage of each machine have a major influence. The algorithms that will be used to solve the assignment problem are based on the Hungarian algorithm, adapting the latter in order to efficiently handle unbalanced assignment scenarios In addition to several variations of the Hungarian algorithm, a stochastic heuristic algorithm, Dhouib-Matrix-AP2, is also examined in the literature. All algorithms are analyzed and implemented using a given dataset, whilst their performance is benchmarked against the optimal solution provided by the Gurobi linear programming solver.

Keywords: Assignment Problem, Hungarian Algorithm, Unbalanced Assignment, Cloud Computing, Linear Programming

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Αλγόριθμοι συνδυαστικής βελτιστοποίησης για προβλήματα ανάθεσης εργασιών σε περιβάλλον cloud computing" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Νικολάου Πλόσκα αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος αλγορίθμων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Ορισμός του προβλήματος

Το πρόβλημα της ανάθεσης εργασιών είναι ένα κλασικό πρόβλημα συνδυαστικής βελτιστοποίησης που έχει στόχο την εύρεση του καλύτερου συνδυασμού στοιχείων μέσα σε ένα διακριτό σύνολο, ικανοποιώντας προκαθορισμένους περιορισμούς. Οι αλγόριθμοι συνδυαστικής βελτιστοποίησης είναι μια ειδική κατηγορία αλγορίθμων βελτιστοποίησης, οι οποίοι εφαρμόζονται σε διάφορους τομείς, όπως η πληροφορική, η επιχειρησιακή έρευνα και τα μαθηματικά. Σε αντίθεση με τον χρονοπρογραμματισμό εργασιών, στο πρόβλημα της ανάθεσης δεν χρειάζεται να καθοριστεί ή να υπάρχει γνώση της σειράς τοποθέτησης των εργασιών στους διαθέσιμους πόρους.

Το πρόβλημα, στην πιο βασική μορφή του, αφορά τον τρόπο ανάθεσης εργασιών σε ανθρώπους, μηχανές ή άλλους πόρους, προκειμένου να μειωθεί ή να μεγιστοποιηθεί ένα προκαθορισμένο κριτήριο, όπως το συνολικό κόστος, ο χρόνος ή η αποδοτικότητα. Γενικά, τα δύο κύρια στοιχεία της ανάθεσης είναι ένα σύνολο εργασιών και ένα σύνολο διαθέσιμων πόρων. Όταν θεσπίστηκε για πρώτη φορά το πρόβλημα της ανάθεσης ο βασικός περιορισμός του προβλήματος ήταν πως κάθε εργασία μπορεί να ανατεθεί μόνο σε έναν πόρο και το αντίστροφο, δηλαδή κάθε πόρος έχει τη δυνατότητα να αναλάβει μόνο μια εργασία. Η βελτιστοποίηση της ανάθεσης τις περισσότερες φορές αναπαρίσταται ως ένας πίνακας κόστους, ορίζοντας το κόστος ανάθεσης που έχει κάθε εργασία σε κάθε πόρο.

Η πτυχή του προβλήματος που θα εστιάσει η παρούσα εργασία αφορά την ανάθεση εργασιών σε περιβάλλον cloud computing. Συγκεκριμένα στο μη ισορρο-

πημένο πρόβλημα τοποθέτησης πολλαπλών εργασιών N σε ένα πλήθος εικονικών μηχανών M (Virtual Machine - VM), έτσι ώστε να ελαχιστοποιηθεί το κόστος και ο χρόνος εκτέλεσης όλων των εργασιών, με τους διαθέσιμους πόρους που του παρέχονται. Όπως έχει προαναφερθεί, το πρόβλημα μπορεί να χαρακτηριστεί είτε ως ισορροπημένο, το πλήθος δηλαδή των εργασιών να είναι ίσο με αυτό των VMs, είτε ως μη ισορροπημένο, όπου οι εργασίες είναι περισσότερες από τα διαθέσιμα VM. Το πρόβλημα της ανάθεσης σε περιβάλλον cloud είναι αρκετά περίπλοκο, καθώς όσο αυξάνεται το πλήθος των εργασιών αυξάνεται ραγδαία και το πλήθος των εφικτών λύσεων. Επιπλέον, κάθε εργασία έχει διαφορετικές απαιτήσεις σε μνήμη, χώρο αποθήκευσης, σε χρόνο που χρειάζεται για να εκτελεστεί και σε υπολογιστική ισχύ, περιορίζοντας τον τρόπο ανάθεσης της.

Μια ακόμα παράμετρος του προβλήματος που μπορεί να χαρακτηρίσει το πρόβλημα είναι η ύπαρξη ή όχι ετερογένειας ανάμεσα στα VMs, αν δηλαδή έχουν διαφορετική μνήμη, χώρο κλπ.. Λόγω του μεγάλου πλήθους λύσεων που μπορεί να έχει το πρόβλημα, κατατάσσεται στην κατηγορία NP-hard, καθώς είναι πολύ δύσκολο ως ακατόρθωτο να βρεθεί βέλτιστη λύση μέσα σε πολυωνυμικό χρόνο. Η επίλυση του προβλήματος γίνεται συνήθως με τον Ουγγρικό αλγόριθμο, μεταευρετικούς αλγόριθμους ή βασισμένους σε σμήνη, καθώς και με μηχανική μάθηση. Η επίδοση των ευρετικών αλγορίθμων εξαρτάται από το εκάστοτε πρόβλημα που εφαρμόζονται. Αν και έχουν τη δυνατότητα να προσφέρουν ακριβή λύση σε πεπερασμένο χρόνο, δεν είναι δεδομένο το ίδιο για δύσκολα προβλήματα βελτιστοποίησης.

1.2 Κίνητρα και στόχοι υλοποίησης

Η επίλυση του προβλήματος της ανάθεσης εργασιών είναι ένα απλό πρόβλημα, το οποίο εύκολα μετατρέπεται σε πολύπλοκο. Όσο αυξάνεται το μέγεθος του προβλήματος, η εύρεση της βέλτιστης λύσης καθίσταται χρονοβόρα και πολλές φορές αδύνατη, λόγω του μεγέθους που έχει ο χώρος αναζήτησης του προβλήματος. Ανά τα χρόνια πληθώρα ερευνητών έχουν ασχοληθεί είτε με τη βελτιστοποίηση του Ουγγρικού αλγορίθμου είτε προτείνοντας νέους εκσυγχρονισμένους αλγόριθμους για την επίλυση μη ισορροπημένων προβλημάτων ανάθεσης εργασιών. Στα πλαίσια της παρούσας διπλωματικής εργασίας θα γίνει ανάλυση των ήδη υπαρχόντων εκδοχών του Ουγγρικού αλγόριθμου, καθώς και του αλγόριθμου Dhouib-Matrix-

AP2. Στόχος της εργασίας είναι η σύγκριση των αλγορίθμων, αναζητώντας τον καλύτερο τρόπο επίλυσης προβλημάτων μη ισορροπημένης ανάθεσης εργασιών. Αφού μελετηθούν οι αλγόριθμοι της βιβλιογραφίας, η εργασία προτείνει μια νέα βελτιωμένη προσέγγιση για την επίλυση προβλημάτων ανάθεσης εργασιών. Η νέα προσέγγιση έχει στόχο τη μετατροπή του Ουγγρικού αλγόριθμου ώστε να έχει τη δυνατότητα να επιλύσει μη ισορροπημένα προβλήματα ανάθεσης.

1.3 Διάρθρωση κειμένου

Η παρούσα διπλωματική χωρίζεται σε 5 κεφάλαια, με το πρώτο να ορίζει το αντικείμενο ενασχόλησης της εργασίας, περιγράφοντας το πρόβλημα ανάθεσης εργασιών, τα κίνητρα και τους στόχους που θέτονται. Μετά την εισαγωγή του προβλήματος, η διάρθρωση του κειμένου συνεχίζει με τα ακόλουθα κεφάλαια.

Στο Κεφάλαιο 2 παρουσιάζεται το βασικό θεωρητικό υπόβαθρο, η σχετική έρευνα επιστημόνων που στοχεύουν στην επίλυση του προβλήματος ανάθεσης εργασιών, καθώς και οι αντίστοιχοι αλγόριθμοι συνδυαστικής βελτιστοποίησης που χρησιμοποιήθηκαν. Στη συνέχεια, για κάθε αλγόριθμο της βιβλιογραφίας γίνεται βήμα προς βήμα επίλυση ενός παραδειγματικού σεναρίου.

Στο Κεφάλαιο 3 γίνεται ανάλυση του σχετικού μοντέλου που περιγράφει το πρόβλημα της ανάθεσης και παρουσιάζεται ένας αλγόριθμος συνδυαστικής βελτιστοποίησης βασισμένος στον Ουγγρικό αλγόριθμο. Έπειτα, παρατίθεται ο αντίστοιχος ψευδοκώδικας του αλγόριθμου και εξετάζεται μέσω της εφαρμογής του σε ένα συγκεκριμένο παράδειγμα, προσφέροντας μια πρακτική εικόνα της λειτουργίας του. Το κεφάλαιο ολοκληρώνεται με τη σύγκριση των αποτελεσμάτων που προκύπτουν από την εφαρμογή όλων των αλγορίθμων του προηγούμενου κεφαλαίου και του Κεφαλαίου 3 στο σενάριο που έχουν επιλύσει.

Στο Κεφάλαιο 4 εξετάζεται εκτενώς η συμπεριφορά των προαναφερόμενων αλγορίθμων και παρουσιάζονται τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν για τη σύγκρισή τους. Ακολούθως, αφού πραγματοποιηθεί σφαιρική ανάλυση της λειτουργικότητας της εργαλειοθήκης CloudSim, προσομοιώνονται οι αλγόριθμοι σε περιβάλλον νέφους.

Τέλος, στο Κεφάλαιο 5 παρατίθενται συνοπτικά τα πορίσματα της μελέτης που διεξήχθη και γίνεται αναφορά σε πιθανές μελλοντικές βελτιώσεις και επεκτάσεις

της έρευνας που παρουσιάστηκε.

Κεφάλαιο 2

Βιβλιογραφική ανασκόπηση

Σε αυτή την ενότητα θα παρουσιαστούν βασικές έννοιες που έχουν σχέση με τον Ουγγρικό αλγόριθμο καθώς και η αντίστοιχη βιβλιογραφική ανασκόπηση. Θα γίνει ανάλυση διαφόρων εκδοχών του Ουγγρικού αλγορίθμου που στοχεύουν στην επίλυση μη ισορροπημένων προβλημάτων ανάθεσης, καθώς και ορισμένων αλγόριθμων σχετικού περιεχομένου. Έπειτα, θα γίνει εφαρμογή κάθε αλγορίθμου ξεχωριστά σε ένα ίδιο παραδειγματικό σενάριο.

2.1 Ουγγρικός αλγόριθμος

Ο πιο γνωστός αλγόριθμος που δημιουργήθηκε για την επίλυση του προβλήματος ανάθεσης σε πολυωνυμικό χρόνο με γραμμικό προγραμματισμό είναι ο Ουγγρικός αλγόριθμος (Hungarian Algorithm). Ο αλγόριθμος αναπτύχθηκε το 1955 από τον H. W. Kuhn [5] κατά τη διατύπωση του προβλήματος ανάθεσης. Ο Ουγγρικός αλγόριθμος βασίζεται σε μια επαναληπτική μέθοδο, όπου κατασκευάζεται ένας πίνακας κόστους, γίνεται έλεγχος κατά πόσο ο πίνακας δίνει τη βέλτιστη λύση και έπειτα ολοκληρώνεται με την ανάλογη ανάθεση των εργασιών. Στην περίπτωση που δεν βρεθεί βέλτιστη λύση ο πίνακας αναπροσαρμόζεται και η διαδικασία επαναλαμβάνεται μέχρι ο αλγόριθμος να οδηγηθεί στη βέλτιστη λύση. Η χρονική πολυπλοκότητα της μεθόδου ήταν αρχικά $O(n^4)$, όπου n το πλήθος των εργασιών προς ανάθεση, ωστόσο παρατηρήθηκε πως η πολυπλοκότητα μπορεί να μειωθεί σε $O(n^3)$[6].

Το μεγάλο μειονέκτημα του Ουγγρικού αλγόριθμου είναι πως μπορεί να εφαρμοστεί μόνο σε ισορροπημένα προβλήματα ανάθεσης εργασιών, δηλαδή σε προβλήματα όπου το πλήθος των εργασιών είναι ίδιο με το πλήθος των διαθέσιμων

μηχανών. Στον Πίνακα 2.1 παρουσιάζεται η βασική μορφή που έχει ένα ισορροπημένο πρόβλημα ανάθεσης, ενώ στον Πίνακα 2.2 απεικονίζεται ένα μη ισορροπημένο πρόβλημα. Στα μη ισορροπημένα προβλήματα ανάθεσης ανήκουν προβλήματα τα οποία δεν έχουν ίσο αριθμό εργασιών με μηχανές. Όταν εφαρμόζεται ο Ουγγρικός αλγόριθμος σε προβλήματα που είναι μη ισορροπημένα, προστίθενται εικονικές μηχανές (dummy) ώστε το πρόβλημα να μετατραπεί σε "ισορροπημένο". Αφού εξασφαλιστεί πως το πρόβλημα πλέον είναι ισορροπημένο, ο αλγόριθμος ξεκινάει να εφαρμόζεται κανονικά. Ωστόσο, αυτή η διαχείριση των μη ισορροπημένων προβλημάτων έχει ως αποτέλεσμα οι εργασίες που ανατεθούν στις dummy μηχανές να μην εκτελεστούν ποτέ. Επιπλέον, όσο το μέγεθος του προβλήματος αυξάνεται, η δημιουργία και χρήση εικονικών μηχανών αυξάνει δραματικά τους υπολογιστικούς πόρους και τη μνήμη που απαιτείται για την επίλυση του προβλήματος.

	J_1	J_2		J_N
M_1	C_{11}	C_{12}	...	C_{1N}
M_2	C_{21}	C_{22}	...	C_{2N}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
M_N	C_{N1}	C_{N2}	...	C_{NN}

Πίνακας 2.1: Πρόβλημα ισορροπημένης ανάθεσης

	J_1	J_2		J_N
M_1	C_{11}	C_{12}	...	C_{1N}
M_2	C_{21}	C_{22}	...	C_{2N}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
M_M	C_{M1}	C_{M2}	...	C_{MN}

Πίνακας 2.2: Πρόβλημα μη ισορροπημένης ανάθεσης

2.2 Παραλλαγή Ουγγρικού αλγόριθμου από Kumar [1]

Το 2006 προτάθηκε από τον A. Kumar [1] μια βελτιωμένη εκδοχή του Ουγγρικού αλγόριθμου, η οποία επικεντρώνεται στην τροποποίηση ενός μη ισορροπημένου προβλήματος σε ισορροπημένο. Αυτή ήταν η πρώτη εκδοχή του Ουγγρικού αλγόριθμου που στόχευε να επιλύσει το πρόβλημα της μη ισορροπημένης ανάθεσης εργασιών χωρίς τη χρήση dummy μηχανών. Στη βασική του μορφή, ο αλγόριθμος χωρίζει το πρόβλημα σε τετραγωνικούς πίνακες, όπου κάθε ένας νέος πίνακας επι-

λύεται εφαρμόζοντας τον Ουγγρικό αλγόριθμο. Στο τέλος, αφού λυθούν διαδοχικά όλα τα επιμέρους προβλήματα, προσθέτονται όλα τα κόστη για να σχηματίσουν το τελικό κόστος ανάθεσης του αρχικού προβλήματος.

Έστω ότι κάθε πρόβλημα έχει τη μορφή πίνακα, όπου οι γραμμές N αναπαριστούν τις εργασίες προς ανάθεση και κάθε στήλη M το κόστος ανάθεσης σε κάθε VM. Ο αλγόριθμος ξεκινάει υπολογίζοντας το άθροισμα κάθε στήλης (RowSum()) και το άθροισμα κάθε γραμμής (ColumnSum()). Αν για το πρόβλημα ισχύει N>M, τότε επιλέγονται m εργασίες σύμφωνα με το ελάχιστο άθροισμα σειράς, κατασκευάζοντας έτσι το πρώτο επιμέρους πρόβλημα, μεγέθους mxm. Η διαδικασία επαναλαμβάνεται για τις υπόλοιπες $(n - m)$ εργασίες, εφόσον συνεχίζουν να είναι περισσότερες από τις διαθέσιμες μηχανές. Αν οι περισσευούμενες εργασίες είναι λιγότερες από το πλήθος των VM, τότε επιλέγονται τόσες στήλες όσες και οι εργασίες, κατασκευάζοντας πάλι έναν τετραγωνικό πίνακα. Η επιλογή των VM γίνεται με βάση το μικρότερο άθροισμα κάθε στήλης ColumnSum().

Αφού γίνει ανακατασκευή του προβλήματος, χωρίζοντας το αρχικό πρόβλημα σε μικρότερα υποπροβλήματα, ξεκινάει η διαδοχική επίλυση τους. Τα προβλήματα λύνονται με την εφαρμογή του κλασσικού Ουγγρικού αλγορίθμου. Γίνεται εύρεση του ελάχιστου μη μηδενικού κόστους κάθε σειράς και αφαιρείται από τα υπόλοιπα κόστη στην αντίστοιχη σειρά. Το ίδιο επαναλαμβάνεται και για κάθε στήλη, δημιουργώντας έναν πίνακα που σε κάθε σειρά υπάρχει ένα τουλάχιστον μηδενικό. Έπειτα, ελέγχονται και μαρκάρονται διαδοχικά τα μηδενικά κάθε γραμμής του μεταεπεξεργασμένου πίνακα κόστους. Είναι απαραίτητο να κατασκευαστεί ένας νέος πίνακας ”markings”, στον οποίο καταγράφονται όλα τα μηδενικά του πίνακα με την ανάλογη σήμανση. Για να δηλωθεί η επίτευξη ανάθεσης μια εργασίας δηλώνεται με τον χαρακτήρα “Δ” στην αντίστοιχη θέση του πίνακα ”markings”. Με τον χαρακτήρα “Χ” δηλώνεται η ύπαρξη μηδενικού στο αντίστοιχο σημείο στον πίνακα κόστους, ενώ τα υπόλοιπα στοιχεία δηλώνονται με τον χαρακτήρα “Ο”.

Αν μια γραμμή έχει ένα μόνο μηδενικό, αυτό σημαίνεται ως “Δ”, ενώ αν στη στήλη που έγινε η ανάθεση υπάρχουν άλλα μηδενικά, εκείνα σημειώνονται με τον χαρακτήρα “Χ”. Η διαδικασία επαναλαμβάνεται ως ότου δεν υπάρχουν στον πίνακα μη μαρκαρισμένα μηδενικά ή αν σε μια σειρά βρίσκονται περισσότερα από δύο μη μαρκαρισμένα μηδενικά. Ομοίως, εξετάζονται και στήλες του πίνακα, ακολουθώντας

την προαναφερόμενη διαδικασία. Η αναζήτηση και η καταγραφή των μηδενικών επαναλαμβάνεται έως ότου πραγματοποιηθεί καταγραφή όλων των μηδενικών του πίνακα. Στην περίπτωση που υπάρχουν περισσότερα από δύο μηδενικά σε μια τουλάχιστον στήλη και τουλάχιστον δύο μηδενικά σε μια σειρά, τα οποία αλληλοκαλύπτονται όπως φαίνεται στον Πίνακα 2.3, δεν είναι εφικτή η καταγραφή τους. Ωστόσο, για να προχωρήσει ο αλγόριθμος στο επόμενο στάδιο είναι αναγκαία η επίλυση αυτού του προβλήματος ώστε να ολοκληρωθεί η διαδικασία καταγραφής των μηδενικών. Επειδή η συγκεκριμένη περίπτωση δεν καλύπτεται από τον Kumar, ακολουθείται μια προτεινόμενη προσέγγιση. Με την νέα προσέγγιση επιλέγεται τυχαία ένα μη καταγεγραμμένο μηδενικό μιας σειράς, στο οποίο, εφόσον δεν υπάρχει ανάθεση στην ανάλογη σειρά και στήλη, πραγματοποιείται ανάθεση. Τα υπόλοιπα μηδενικά της σειράς σημαίνονται με τον χαρακτήρα "Χ", όπως και τα αντίστοιχα μηδενικά της στήλης που έγινε μόλις η ανάθεση. Η μέθοδος αυτή εφαρμόζεται διαδοχικά για κάθε σειρά του πίνακα, έως ότου να έχουν καλυφθεί όλα τα μηδενικά του πίνακα.

	J_1	J_2		J_N
M_1	C_{11}	C_{12}	...	C_{1N}
M_2	0	0	...	C_{2N}
⋮	⋮	⋮		⋮
⋮	0	0		⋮
M_{N-1}	⋮	⋮		⋮
M_N	C_{N1}	C_{N2}	...	C_{NN}

Πίνακας 2.3: Μη δυνατή καταγραφή μηδενικών

Μόλις ολοκληρωθεί το προαναφερόμενο τμήμα του αλγόριθμου, διασφαλίζοντας πως έχουν καταγραφεί όλα τα μηδενικά του υποπροβλήματος, γίνεται έλεγχος αν είναι δυνατή η βέλτιστη ανάθεση. Ο αλγόριθμος προχωράει στο τελικό στάδιο της ανάθεσης όταν σε κάθε στήλη του πίνακα υπάρχει το σύμβολο "Δ", υποδεικνύοντας την επίτευξη ανάθεσης σε κάθε εργασία του υποπροβλήματος. Αλλιώς, αν δεν έχει βρεθεί ολοκληρωμένο πλάνο ανάθεσης είναι απαραίτητο να πραγματοποιηθεί περαιτέρω επεξεργασία του πίνακα. Γίνεται διαδοχικός έλεγχος πρώτα στις γραμμές που δεν υπάρχει ανάθεση, δεύτερον στις στήλες που δεν έχουν ήδη ελεγχθεί αλλά έχουν μηδέν στις ελεγχθείσες γραμμές, και τέλος στις γραμμές που δεν έχουν ελεγχθεί αλλά υπάρχει ανάθεση "Δ". Μετέπειτα ο αλγόριθμος ακολουθεί

τον Ουγγρικό αλγόριθμο, χαράζοντας τον ελάχιστο αριθμό γραμμών στον πίνακα, με σκοπό την κάλυψη όλων των μηδενικών, ενώ αναζητεί τη μικρότερη τιμή που δεν διαπερνάται από κάποια γραμμή. Η τιμή αυτή αφαιρείται από κάθε ακάλυπτο κόστος στον πίνακα, ενώ προστίθεται σε κόστη που βρίσκονται σε σημεία τομής των γραμμών. Όταν ολοκληρωθεί η επεξεργασία του υποπροβλήματος, επαναλαμβάνεται από την αρχή η διαδικασία αναζήτησης και καταγραφής των μηδενικών του πίνακα με τα σύμβολα "Δ" και "Χ". Αφού ολοκληρωθεί η ανάθεση σε κάθε ένα από τα υποπροβλήματα, υπολογίζονται τα κόστη ανάθεσης κάθε υποπροβλήματος και προσθέτονται μεταξύ τους για να βρεθεί το συνολικό κόστος της ανάθεσης του αρχικού προβλήματος.

Στο τελικό στάδιο του Ουγγρικού αλγόριθμου πραγματοποιείται η ανάθεση εργασίας σε κάποιο διαθέσιμο μηχάνημα. Για τον αλγόριθμο του Kumar, η διαδικασία αυτή είναι μερικώς ολοκληρωμένη, αρκεί στον πίνακα συμβόλων να υπάρχουν τόσα σύμβολα "Δ" όσες και εργασίες. Για να ολοκληρωθεί η ανάθεση, γίνεται αναζήτηση στον πίνακα συμβόλων για τους χαρακτήρες "Δ" που δηλώνουν την ανάθεση μιας εργασίας σε μηχανή στην αντίστοιχη θέση x_{ij} που εντοπίστηκε το σύμβολο. Το κόστος της ανάθεσης αναγράφεται στον αρχικό πίνακα κόστους C στην αντίστοιχη θέση που έγινε η ανάθεση. Για προβλήματα μεγάλου μεγέθους, ο αλγόριθμος μπορεί να εισέλθει στο τελικό στάδιο ανάθεσης έχοντας στον πίνακα συμβόλων N-1 σύμβολα "Δ". Αφού γίνει ανάθεση σε αυτά τα σύμβολα, η τελική ανάθεση γίνεται στην τομή της σειράς και της στήλης που δεν έχει γίνει καμία ανάθεση.

Μολονότι ο προαναφερόμενος αλγόριθμος ήταν ο πρώτος που αντιμετώπισε το πρόβλημα των μη ισορροπημένων αναθέσεων με τη χρήση του Ουγγρικού αλγορίθμου, έχει ένα βασικό ελάττωμα: την αδυναμία εύρεσης της συνολικά βέλτιστης λύσης. Το μειονέκτημα αυτό οφείλεται στη διάσπαση του κύριου προβλήματος σε μικρότερα υποπροβλήματα. Παρόλο που η βέλτιστη λύση βρίσκεται σε κάθε επιμέρους πρόβλημα, δεν εξασφαλίζεται το ίδιο στο κύριο πρόβλημα. Ένα μεγάλο ποσοστό των αναθέσεων που πραγματοποιούνται συνδέονται είτε άμεσα είτε έμμεσα με προηγούμενες επιλογές ανάθεσης, καθιστώντας εξαιρετικά δύσκολη την επιτυχή εύρεση βέλτιστης λύσης διασπάζοντας το πρόβλημα. Επομένως, όσο αυξάνεται το μέγεθος του προβλήματος, το χάσμα ανάμεσα στη λύση που προκύπτει από τον αλγόριθμο και στο βέλτιστο αποτέλεσμα θα συνεχίσει να αυξάνεται ση-

μαντικά.

Αλγόριθμος 1: Παραλλαγή Ουγγρικού αλγόριθμου από Kumar [1] - Part 1

```
Input: N = Number of tasks, M = number of VMm, Cost matrix C
Output: Optimal or near Optimal Assignment Plan and total Assignment
        Cost
for Each Column do
    Calculate the sum of each column: Sum_Column
for Each Row do
    Calculate the sum of each row: Sum_Row
Initialize k to 0
Initialize remainingRows to total number of rows in matrix
while There are Remaining Tasks do
    if Tasks are more than the available VMs then
        while Size of sub problem is smaller than M do
            Find the index i with the minimum value in Sum_Row and add
             index i in subproblem_pos list
        Sort subproblem_pos in ascending order
        Create a subMatrix by selecting rows from matrix using indexes in
         subproblem_pos
        Update remainingRows by subtracting column
    else if Tasks are less than the available VMs then
        while The size of sub_vm_pos is smaller than remainingRows do
            Find the index j with the minimum value in Sum_Column and
             add index j to sub_vm_pos
        Sort sub_vm_pos in ascending order
        for Each task not used in other subproblems do
            for Each VM in sub_vm_pos do
                Add row to subMatrix
            Write index i in "jobPosition_k.txt"
        Create a subMatrix by selecting rows from matrix using indexes in
         subproblem_pos
        Update remainingRows by subtracting column
```

Αλγόριθμος 2: Παραλλαγή Ουγγρικού αλγόριθμου από Kumar [1] - Part 2

```
for Each subproblem k created do
    Read file "file_k.txt" and write to subMatrix matrix
    for each column of subMatrix matrix do
        Find the minimum cost c_ij in column and subtract c_ij from each cost
         in the column.
    for each row of subMatrix matrix do
        Find the minimum cost c_ij in row and subtract c_ij from each cost in
         the row.
    while Assignment uncompleted do
        while true do
            Initialize markings matrix as "0"
            while There are no unmarked zeros or more than 2 in a row do
                for Each row in the subMatrix do
                    Count unmarked zeros in row
                    if There is one unmarked zero in Row then
                        Mark zero in markings matrix as "D"
                        for Each unmarked zero in the column do
                            Mark zero in markings matrix as "X"
                Count unmarked zeros
            Count marked "D", "X" and unmarked zeros
            if Unmarked zeros = 0 then
                Exit While
            if There is no change in markings then
                for Each row do
                    Randomly assign zeros as "D" in markings matrix
                    Assign as "X" unmarked zeros in the column and rest of
                     row
                Count marked "D", "X" and unmarked zeros
                Exit While
```

Αλγόριθμος 3: Παραλλαγή Ουγγρικού αλγόριθμου από Kumar [1] - Part 3

```
for Each subproblem k created do
                                          ▷ Continue previous page
    while Assignment uncompleted do
        if No unmarked zeros but not complete assignment then
            while Checking continues do
                Check rows for which assignment "D" has not been made
                Check columns which have a zero in checked rows
                Check rows which have assignments in the checked column
            Draw lines through all unchecked rows and through all checked
             columns
            Select the smallest uncovered element and subtract it from every
             other uncovered element
            Add the same element to every element that is at intersection of
             lines
        if Marked "D" = Subproblem size then
            for Each row in the subMatrix do
                Assign task where "D" is found in markings matrix. Use file
                 "jobPosition_k.txt" to determine task based on index
                if Subproblem size is smaller than M then
                    Use file "VMUsed_k.txt" to determine which VM is picked
            Calculate total assignment cost of subproblem.
Add all sub-problems total costs.
Print final Assignment Plan.
```

2.2.1 Εφαρμογή παραλλαγής Ουγγρικού αλγόριθμου από Kumar[1]

Σε αυτό το σημείο θα εξεταστεί η εφαρμογή του Ουγγρικού αλγορίθμου που προτείνει ο Kumar στο παραδειγματικό σενάριο που απεικονίζεται στον Πίνακα 2.4. Όπως φαίνεται, το παράδειγμα παρουσιάζει ένα πρόβλημα μη ισορροπημένης ανάθεσης, όπου 8 εργασίες χρειάζεται να ανατεθούν σε 5 διαθέσιμα μηχανήματα. Κάθε σειρά περιγράφει μια εργασία και κάθε στήλη το αντίστοιχο κόστος ανάθεσης που θα έχει αυτή η εργασία σε κάθε μηχάνημα. Σύμφωνα με τον Kumar [1], το πρόβλημα πρέπει να χωριστεί σε δύο τετραγωνικά υποπροβλήματα, με σκοπό να

αντιμετωπιστούν ως ισορροπημένα προβλήματα ανάθεσης.

	M_1	M_2	M_3	M_4	M_5
J_1	300	290	280	290	210
J_2	250	310	290	300	200
J_3	180	190	300	190	180
J_4	320	180	190	240	170
J_5	270	210	190	250	160
J_6	190	200	220	190	140
J_7	220	300	230	180	160
J_8	260	190	260	210	180

Πίνακας 2.4: Παράδειγμα προβλήματος μη ισορροπημένης ανάθεσης

Για να γίνει ο επιμέρους διαχωρισμός του προβλήματος χρειάζεται αρχικά να υπολογιστεί το άθροισμα των τιμών κάθε στήλης και το άθροισμα των τιμών κάθε σειράς, όπως φαίνεται παρακάτω στους Πίνακες 2.6 και 2.5 αντίστοιχα. Το πρώτο υποπρόβλημα θα έχει μέγεθος 5x5, αξιοποιώντας όλα τα διαθέσιμα μηχανήματα και επιλέγοντας εργασίες σύμφωνα με το μικρότερο άθροισμα σειρών. Οι εργασίες που θα επιλεχθούν θα είναι η J_3, J_4, J_5, J_6 και J_7, αφού σε αυτές τις σειρές υπάρχουν τα πέντε μικρότερα αθροίσματα σειρών. Το πρώτο πρόβλημα παρουσιάζεται παρακάτω, έχοντας ως πίνακα κόστους τον Πίνακα 2.7.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
SumRow()	1,370	1,350	1,040	1,100	1,080	940	1,090	1,100

Πίνακας 2.5: Άθροισμα σειρών

	M_1	M_2	M_3	M_4	M_5
SumColumn()	1,990	1,870	1,960	1,850	1,400

Πίνακας 2.6: Άθροισμα στηλών

Σύμφωνα με τον Ουγγρικό αλγόριθμο, το επόμενο βήμα είναι η επεξεργασία του πίνακα κόστους, και συγκεκριμένα η δημιουργία μηδενικών σε αυτών. Σε κάθε σειρά αναζητείται το μικρότερο κόστος που υπάρχει, το οποίο αφαιρείται από όλα τα στοιχεία της σειράς, δημιουργώντας τουλάχιστον ένα μηδενικό σε κάθε σειρά του πίνακα. Έπειτα, επαναλαμβάνεται η αντίστοιχη διαδικασία για κάθε στήλη, ωστόσο σε στήλες που το μικρότερο κόστος είναι μηδενικό δεν μεταβάλλονται τα στοιχεία του πίνακα. Ο νέος πίνακας που προκύπτει από την επεξεργασία του αρχικού πίνακα κόστους φαίνεται στον Πίνακα 2.8.

	M_1	M_2	M_3	M_4	M_5
J_3	180	190	300	190	180
J_4	320	180	190	240	170
J_5	270	210	190	250	160
J_6	190	200	220	190	140
J_7	220	300	230	180	160

Πίνακας 2.7: Πρώτο υποπρόβλημα

	M_1	M_2	M_3	M_4	M_5
J_3	0	0	0	0	0
J_4	150	0	0	60	0
J_5	110	40	10	80	0
J_6	50	50	70	40	0
J_7	60	130	60	10	0

Πίνακας 2.8: Δημιουργία μηδενικών

Ακολουθώντας τον αλγόριθμο που περιγράφει ο Kumar, σε αυτό το σημείο θεωρείται αναγκαία η κατασκευή του πίνακα συμβόλων για το πρώτο υποπρόβλημα. Αφού δημιουργηθεί ένας πίνακας μεγέθους 5x5, όπου κάθε στοιχείο s_{ij} είναι "Ο", εξετάζονται διαδοχικά οι σειρές του Πίνακα 2.8. Η πρώτη ανάθεση "Δ" θα γίνει στο στοιχείο s_{35}, αφού η τρίτη σειρά είναι η πρώτη φόρα που εντοπίζεται ένα μόνο μηδενικό, ενώ όλα τα υπόλοιπα μηδενικά της στήλης 5 θα καταγραφούν με το σύμβολο "Χ". Στη συνέχεια, θα εξεταστούν διαδοχικά οι στήλες, καθώς δεν υπάρχει άλλη σειρά που να πληρεί τις προϋποθέσεις ανάθεσης. Συγκεκριμένα, με "Δ" θα σημανθεί το στοιχείο s_{11} και s_{22}, ενώ παράλληλα καλύπτονται όλα τα μηδενικά του πίνακα. Όπως φαίνεται στον Πίνακα 2.9, έχουν γίνει μόνο τρεις αναθέσεις "Δ" στο υποπρόβλημα, ενώ το ελάχιστο πλήθος γραμμών που χρειάστηκε να χαραχθούν στον πίνακα για να καλυφθούν όλα τα μηδενικά είναι επίσης τρία.

	M_1	M_2	M_3	M_4	M_5
J_3	~~Δ~~	~~Χ~~	~~Χ~~	~~Χ~~	~~Χ~~
J_4	~~0~~	~~Δ~~	~~Χ~~	~~0~~	~~Χ~~
J_5	0	0	0	0	Δ
J_6	0	0	0	0	Χ
J_7	0	0	0	0	Χ

Πίνακας 2.9: Πίνακας συμβόλων

Επειδή το πλήθος των αναθέσεων "Δ" είναι μικρότερο από το πλήθος των εργασιών, το υποπρόβλημα πρέπει να υποβληθεί σε περαιτέρω επεξεργασία. Έπειτα

από αναζήτηση στον πίνακα, το μικρότερο μη καλυπτόμενο κόστος βρίσκεται στη θέση x_{33} και έχει τιμή 10, η οποία αφαιρείται από κάθε μη καλυπτόμενο στοιχείο, ενώ προστίθεται στα στοιχεία x_{15} και x_{25} που βρίσκονται σε τομές χαραγμένων γραμμών. Ο νέος ανανεωμένος πίνακας απεικονίζεται με τον Πίνακα 2.10, ωστόσο αναγκαία είναι και η ανανέωση του πίνακα συμβόλων. Ακολουθώντας για μια ακόμα φορά τα βήματα που περιγράφονται στον αλγόριθμο που προτείνει ο Kumar, η πρώτη σήμανση "Δ" θα γίνει στη θέση s_{45}, ενώ η επόμενη θα είναι στην τελευταία σειρά και συγκεκριμένα στη θέση x_{54}. Επαναλαμβάνοντας από την αρχή τον έλεγχο στις σειρές του προβλήματος, διαπιστώνεται πως οι προηγούμενες σημάνσεις που πραγματοποιήθηκαν οδήγησαν στη δυνατότητα νέας ανάθεσης στην τρίτη σειρά, καθώς από τα δύο μηδενικά που έχει η σειρά, το ένα έχει ήδη μαρκαριστεί με το σύμβολο "Χ", οπότε το δεύτερο μηδενικό στη θέση 3 είναι ελεύθερο για να γίνει ανάθεση. Το ίδιο συμβαίνει και για τις σειρές 2 και 1, όπου μαρκάρονται με τον χαρακτήρα "Δ" τα στοιχεία s_{21} και s_{11}.

	M_1	M_2	M_3	M_4	M_5
J_3	0	0	0	0	10
J_4	150	0	0	60	10
J_5	100	30	0	70	0
J_6	40	40	60	30	0
J_7	50	120	50	0	0

Πίνακας 2.10: Επεξεργασμένος πίνακας προβλήματος

	M_1	M_2	M_3	M_4	M_5
J_3	~~Δ~~	~~Χ~~	~~0~~	~~Χ~~	~~0~~
J_4	~~0~~	~~Δ~~	~~Χ~~	~~0~~	~~Χ~~
J_5	~~0~~	~~0~~	~~Δ~~	~~0~~	~~Χ~~
J_6	~~0~~	~~0~~	~~0~~	~~0~~	~~Δ~~
J_7	~~0~~	~~0~~	~~0~~	~~Δ~~	~~Χ~~

Πίνακας 2.11: Πίνακας συμβόλων νέου πίνακα

Πλέον, το πρόβλημα μπορεί να προχωρήσει στο τελικό στάδιο καθώς έχει επιτευχθεί ανάθεση για όλες τις εργασίες του προβλήματος. Όλες οι αναθέσεις παρουσιάζονται αναλυτικά στον Πίνακα 2.12 μαζί με το κόστος κάθε ανάθεσης. Το τελικό κόστος ανάθεσης για το υποπρόβλημα είναι 870, το οποίο θα προστεθεί στο τέλος με το κόστος που θα προκύψει από το δεύτερο υποπρόβλημα, σχηματίζοντας το συνολικό κόστος ανάθεσης για το αρχικό πρόβλημα.

Machine		Job	Cost
M_1	$\rightarrow$	J_3	180
M_2	$\rightarrow$	J_4	180
M_3	$\rightarrow$	J_5	190
M_4	$\rightarrow$	J_7	180
M_5	$\rightarrow$	J_6	140
Total Cost			870

Πίνακας 2.12: Αναθέσεις πρώτου υποπροβλήματος

Το δεύτερο υποπρόβλημα που θα σχηματιστεί θα είναι και το τελευταίο, αφού οι εργασίες που υπολείπονται είναι λιγότερες από τα μηχανήματα. Συγκεκριμένα, οι εργασίες J_1, J_2 και J_8 είναι οι αυτές που δεν έχουν ανατεθεί, οπότε για να είναι το υποπρόβλημα ισορροπημένο επιλέγονται οι μηχανές M_2, M_4 και M_5. Η επιλογή τους έγινε σύμφωνα με το άθροισμα στήλης, επιλέγοντας τα μικρότερα από αυτά. Το δεύτερο υποπρόβλημα περιγράφεται με τον Πίνακα 2.13. Ακολουθώντας την ίδια διαδικασία με πριν, το πρόβλημα μετά τη δημιουργία μηδενικών απεικονίζεται με τον Πίνακα 2.14.

	M_1	M_4	M_5
J_1	300	290	210
J_2	250	300	200
J_8	260	210	180

Πίνακας 2.13: Δεύτερο υποπρόβλημα

	M_1	M_4	M_5
J_1	70	50	0
J_2	100	70	0
J_8	0	0	0

Πίνακας 2.14: Δημιουργία μηδενικών

Κατόπιν εξετάζεται ο πίνακας που δημιουργήθηκε, ελέγχοντας κατά πόσο είναι εφικτή η εύρεση της βέλτιστης ανάθεσης. Ξεκινώντας από την πρώτη σειρά, στην οποία υπάρχει ένα μοναδικό μηδενικό, γίνεται καταγραφή αυτού στον πίνακα συμβόλων με τον χαρακτήρα "Δ". Όπως περιγράφηκε και στο προηγούμενο υπό πρόβλημα, τα υπόλοιπα μηδενικά που βρίσκονται σε αυτή τη στήλη μαρκάρονται ως "Χ". Ο αλγόριθμος, συνεχίζει ελέγχοντας με τον ίδιο τρόπο της στήλες του προβλήματος, καθώς δεν υπάρχουν πλέον σειρές με ένα μηδενικό. Η τελευταία ανάθεση γίνεται στην πρώτη στήλη, στη θέση $s11$, με την οποία ολοκληρώνεται ο έλεγχος του πίνακα.

Όπως φαίνεται στον Πίνακα 2.15 δεν έχει επιτευχθεί ολοκληρωμένη ανάθεση στο πρόβλημα, καθώς έχουν γίνει δύο από τις τρεις αναθέσεις, οπότε συνεχίζεται η επεξεργασία του προβλήματος. Σαρώνοντας τον πίνακα, το μικρότερο μη καλυπτόμενο στοιχείο είναι το 50, το οποίο εντοπίζεται στη θέση x_{12}. Αφαιρώντας

	M_1	M_4	M_5
J_1	0	0	Δ
J_2	0	0	X
J_8	Δ	X	X

Πίνακας 2.15: Πίνακας συμβόλων

το στοιχείο από τα μη καλυπτόμενα στοιχεία και προσθέτοντας το στο στοιχείο που βρίσκεται στην τομή των χαραγμένων γραμμών, προκύπτει ο Πίνακας 2.16. Σε αυτόν τον πίνακα, ακολουθώντας την ίδια μέθοδο για τον σχηματισμό του πίνακα συμβόλων, μαρκάρεται σε κάθε σειρά το σύμβολο "Δ", ολοκληρώνοντας έτσι τη διαδικασία ελέγχου και επεξεργασίας του πίνακα. Πλέον, το πρόβλημα είναι έτοιμο να επιλυθεί, αναθέτοντας διαδοχικά τις εργασίες στο αντίστοιχο μηχάνημα σύμφωνα με τον πίνακα συμβόλων. Το τελικό κόστος ανάθεσης του υποπροβλήματος είναι 680, ενώ στον Πίνακα 2.18 παρουσιάζονται αναλυτικά όλες οι αναθέσεις με το ανάλογο κόστος ανάθεσης.

	M_1	M_4	M_5
J_1	20	0	0
J_2	50	20	0
J_8	0	0	50

Πίνακας 2.16: Επεξεργασμένος πίνακας προβλήματος

	M_1	M_4	M_5
J_1	0	Δ	X
J_2	0	0	Δ
J_8	Δ	X	0

Πίνακας 2.17: Πίνακας συμβόλων νέου πίνακα

Το τελικό κόστος ανάθεσης για το πρόβλημα είναι 1,550, ενώ όλες οι αναθέσεις που πραγματοποιήθηκαν φαίνονται αναλυτικά στον Πίνακα 2.19. Γενικά, ο Ουγγρικός αλγόριθμος που προτείνει ο Kumar καταφέρνει να βρει σχετικά εύκολα και γρήγορα λύση στο πρόβλημα, χωρίζοντας σε μικρότερα υποπροβλήματα μεγέθους ίσο με το πλήθος των διαθέσιμων μηχανών. Αν και βρίσκει τη βέλτιστη λύση σε κάθε ένα από τα υποπροβλήματα, ο αλγόριθμος δεν εγγυάται πως η συνολική λύση είναι και η βέλτιστη. Σε παρακάτω ενότητες θα αποδειχθεί πως η προσέγγιση του Kumar όχι μόνο δεν βρίσκει τη βέλτιστη λύση στο παράδειγμα που μόλις

Machine		Job	Cost
M_2	$\rightarrow$	J_8	290
M_4	$\rightarrow$	J_1	200
M_5	$\rightarrow$	J_2	190
Total Cost			680

Πίνακας 2.18: Αναθέσεις δεύτερου υποπροβλήματος

Machine		Job	Cost
M_1	$\rightarrow$	J_3	180
M_2	$\rightarrow$	J_4, J_8	180, 290
M_3	$\rightarrow$	J_5	190
M_4	$\rightarrow$	J_1, J_7	200, 180
M_5	$\rightarrow$	J_2, J_6	190, 140
Total Cost			1,550

Πίνακας 2.19: Πίνακας αναθέσεων παραλλαγής Ουγγρικού αλγόριθμου από Kumar

αναλύθηκε, αλλά απέχει αρκετά από αυτή.

2.3 Παραλλαγή Ουγγρικού αλγόριθμου από Rabbani [2]

Μια καλύτερη αντιμετώπιση του προβλήματος επίλυσης μη ισορροπημένης ανάθεσης πρότειναν οι Rabbani, Khan και Quddoos, όπου επίσης βασίζονται στον Ουγγρικό αλγόριθμο [2]. Σε αντίθεση με την προσέγγιση του Kumar, το πρόβλημα λύνεται συνολικά χωρίς να διαιρεθεί σε ισορροπημένα υποπροβλήματα. Γενικά, σε κάθε πρόβλημα θεωρείται πως μια εργασία μπορεί να ανατεθεί αποκλειστικά σε ένα VM, ενώ κάθε μηχανή μπορεί να αναλάβει περισσότερες από μια εργασίες. Η συγκεκριμένη εκδοχή του Ουγγρικού αλγόριθμου ήταν η πρώτη φορά που τα μη ισορροπημένα προβλήματα λύνονται συνολικά, χωρίς τη χρήση dummy μηχανών.

Έστω ότι ο πίνακας κόστους απεικονίζει το πρόβλημα με N εργασίες και M διαθέσιμες μηχανές, όπου κάθε στήλη αναπαριστά κάθε εργασία και κάθε γραμμή το αντίστοιχο κόστος ανάθεσης της εργασίας σε εκείνη μηχανή. Η επίλυση του προβλήματος ξεκινάει με την επεξεργασία του αρχικού πίνακα κόστους, βρίσκοντας αρχικά το μικρότερο κόστος c_{ij} κάθε στήλης και αφαιρώντας το από κάθε άλλο στοιχείο της αντίστοιχης στήλης. Η διαδικασία αυτή επαναλαμβάνεται και για κάθε σειρά, διαμορφώνοντας έναν πίνακα με μηδενικά σε κάθε στήλη και γραμμή. Αφού ολοκληρωθεί η διαμόρφωση του πίνακα, αρχίζει μια επαναληπτική διαδικασία με την οποία υπολογίζεται κατά πόσο είναι δυνατή η επίτευξη βέλτιστης λύσης για το

πρόβλημα. Ο έλεγχος γίνεται με τη χρήση γραμμών, με στόχο τη χάραξη ελάχιστου αριθμού γραμμών που διαπερνούν όλα τα μηδενικά του πίνακα. Εφόσον το πλήθος των γραμμών που χαράχτηκαν είναι ίσο με το πλήθος των μηχανών, είναι εφικτή η βέλτιστη επίλυση του προβλήματος, αλλιώς το πρόβλημα είναι απαραίτητο να μεταβεί σε περεταίρω επεξεργασία.

Όπως προαναφέρθηκε παραπάνω, σκοπός είναι η κάλυψη των μηδενικών του πίνακα. Στην περίπτωση που δεν είναι εφικτή η εύρεση της βέλτιστης λύσης στο πρόβλημα, χρειάζεται να δημιουργηθούν μηδενικά σε νέες θέσεις. Με αναζήτηση σε όλο τον πίνακα βρίσκεται το ελάχιστο, μη καλυπτόμενο από γραμμές κόστος c_{ij}. Το κόστος αυτό αφαιρείται από τα υπόλοιπα μη καλυπτόμενα κόστη, ενώ σε κόστη που βρίσκονται σε τομές των χαραγμένων γραμμών, η τιμή c_{ij} προστίθεται. Η διαδικασία αυτή επαναλαμβάνεται έως ότου να σχεδιαστεί ο ελάχιστος αριθμός χαραγμένων γραμμών, με το πλήθος τους να είναι ίσο ή μικρότερο από αυτό των μηχανών, μέχρι δηλαδή να είναι δυνατή η εύρεση της βέλτιστης λύσης. Όταν ολοκληρωθεί αυτή η διαδικασία το πρόβλημα είναι έτοιμο να προχωρήσει στην τελική φάση της ανάθεσης.

Η ανάθεση των εργασιών γίνεται σύμφωνα με το πλήθος των μηδενικών που βρίσκονται σε κάθε σειρά. Είναι μια επαναλαμβανόμενη διαδικασία, στην οποία αναζητούνται αρχικά σειρές στις οποίες υπάρχει αποκλειστικά μόνο ένα μηδενικό. Σε σειρές που συμβαίνει αυτό, σημαίνει πως είναι δυνατή η επίτευξη της βέλτιστης ανάθεσης της εργασίας j στο μηχάνημα i. Επιλέγονται, λοιπόν, διαδοχικά σειρές που έχουν ένα μηδενικό και γίνεται ανάθεση της εργασίας στην αντίστοιχη θέση που βρίσκεται το μηδενικό. Αν βρίσκονται άλλα μηδενικά σε εκείνη τη στήλη που έγινε η ανάθεση διαγράφονται, καθώς κάθε εργασία δρομολογείται μόνο σε ένα VM. Σε σειρές με περισσότερα από ένα μηδενικά, η ανάθεση γίνεται σύμφωνα με τον αρχικό πίνακα κόστους. Συγκεκριμένα σε εκείνες τις σειρές, η εργασία που ανατίθεται είναι αυτή με το μικρότερο κόστος c_{ij} του αρχικού πίνακα κόστους C. Οι δύο προαναφερόμενοι τρόποι επιλογής ανάθεσης επαναλαμβάνονται έως ότου να πραγματοποιηθεί ανάθεση όλων των εργασιών στα VM. Το τελικό κόστος ανάθεσης προκύπτει από την πρόσθεση όλων των τιμών που αντιστοιχούν στις θέσεις ανάθεσης του αρχικού πίνακα.

Η προσέγγιση του προβλήματος ανάθεσης εργασιών με τη χρήση του προτεινό-

μενου βελτιωμένου Ουγγρικού αλγορίθμου είναι σίγουρα καλύτερη επιλογή από την πρόταση του Kumar. Η επίλυση του προβλήματος στοχεύει στη συνολική αντιμετώπιση του προβλήματος, αποφεύγοντας τα μειονεκτήματα που προκύπτουν όταν γίνεται υποδιαίρεση του αρχικού προβλήματος σε μικρότερα. Πλέον, με τον Ουγγρικό αλγόριθμο είναι δυνατή η επίλυση μη ισορροπημένων προβλημάτων ανάθεσης εργασιών, έχοντας μικρή ως και μηδαμινή απόκλιση από τη βέλτιστη λύση. Φυσικά, όσο μεγαλώνει το μέγεθος του προβλήματος, αυξάνεται η πολυπλοκότητα του αλγόριθμου και η απόκλιση, ωστόσο σε σύγκριση με το μέγεθος του προβλήματος δεν είναι ιδιαίτερα αισθητή. Παρόλα αυτά, είναι αναγκαίο να σημειωθεί πως η επίτευξη μιας τόσο μικρής απόκλισης γίνεται πάντα σε βάρος του χρόνου. Όσο αυξάνεται ο χώρος αναζήτησης του προβλήματος είναι λογικό πως είναι απαραίτητο να διατεθεί ο ανάλογος χρόνος για να καταφέρει ο αλγόριθμος να βρει τη βέλτιστη ή σχεδόν βέλτιστη λύση.

Αλγόριθμος 4: Παραλλαγή Ουγγρικού αλγόριθμου από Rabbani και Khan [2]

Input: n = Number of tasks, m = number of VMs, Cost matrix C

Output: Optimal or near Optimal assignment plan

for each column of C matrix do
    Find the minimum cost c_{ij} in column and subtract c_{ij} from each cost in the column.
end

for each row of Cost matrix do
    Find the minimum cost c_{ij} in row and subtract c_{ij} from each cost in the row.
end

while Optimal assignment not found do
    Draw the minimum number of lines to cover every zero in C matrix.
    Find the smallest uncovered cost c_{ij}.
    Add c_{ij} to the costs located at each intersection of lines.
    Subtract c_{ij} from each uncovered costs in the C matrix.
end

while Assignment not complete do
    for Each Row do
        if Tasked is not assigned then
            Find if there is only one 0 in the column and assign that task to VM.
        end
        for Each task do
            Count number of zeros in each row.
            if There are multiple Zeros in the row then
                Assign task, choosing the smallest cost c_{ij} according to original cost matrix.
            end
        end
    end
end

Print Assignment Plan.

Calculate total cost of assignment plan.

2.3.1 Εφαρμογή παραλλαγής Ουγγρικού αλγόριθμου από Rabbani [2]

Σε αυτό το σημείο θα μελετηθεί η συμπεριφορά του τροποποιημένου Ουγγρικού αλγορίθμου εφαρμόζοντας τον στο παραδειγματικό σενάριο που απεικονίζεται στον Πίνακα 2.20. Όπως φαίνεται, το παράδειγμα παρουσιάζει ένα πρόβλημα μη ισορροπημένης ανάθεσης, όπου 8 εργασίες χρειάζεται να ανατεθούν σε 5 διαθέσιμα μηχανήματα. Αρχικά θα γίνει η προετοιμασία του πίνακα κόστους, δημιουργώντας μηδενικά στις στήλες και στις σειρές. Αφού βρεθεί το μικρότερο κόστος κάθε στήλης, αφαιρείται από τα υπόλοιπα στοιχεία της. Η ίδια διαδικασία θα επαναληφθεί και για τις σειρές, δημιουργώντας τον Πίνακα 2.21.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	300	250	180	320	270	190	220	260
M_2	290	310	190	180	210	200	300	190
M_3	280	290	300	190	190	200	230	260
M_4	290	300	190	240	250	190	180	210
M_5	210	200	180	170	160	140	160	180

Πίνακας 2.20: Παράδειγμα προβλήματος μη ισορροπημένης ανάθεσης

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	90	50	0	150	110	50	60	80
M_2	70	100	0	0	40	50	130	0
M_3	50	70	100	0	10	60	50	60
M_4	70	90	0	60	80	40	10	20
M_5	0	0	0	0	0	0	0	0

Πίνακας 2.21: Πίνακας μετά τη δημιουργία μηδενικών

Μετά την κατασκευή του παραπάνω πίνακα ακολουθεί η αναζήτηση της βέλτιστης λύσης σε αυτόν. Χαράζοντας γραμμές στον Πίνακα 2.21, το ελάχιστο πλήθος γραμμών που χρειάζεται για να καλύψει όλα τα μηδενικά είναι τέσσερα, όποτε ο πίνακας πρέπει να μεταβληθεί περαιτέρω. Αναζητώντας στον Πίνακα 2.22, που παρουσιάζει τις χαραγμένες γραμμές, το μικρότερο μη καλυπτόμενο από γραμμές κόστος είναι το 10 που βρίσκεται στη θέση c_{47}. Το κόστος αυτό αφαιρείται από κάθε μη καλυπτόμενο κόστος, ενώ στις θέσεις c_{53}, c_{54} και c_{58}, που βρίσκονται σε τομές γραμμών, το κόστος προστίθεται. Στον νέο Πίνακα 2.23 που προκύπτει, επαναλαμβάνεται η διαδικασία χάραξης γραμμών. Πλέον οι ελάχιστες δυνατές γραμμές που χρειάζονται για να καλυφθούν όλα τα μηδενικά του Πίνακα 2.24 είναι πέντε,

ίσος αριθμός δηλαδή με τις σειρές του πίνακα, οπότε το πρόβλημα είναι έτοιμο προς επίλυση.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	90	50	0	150	110	50	60	80
M_2	70	100	0	0	40	50	130	0
M_3	50	70	100	0	10	60	50	60
M_4	70	90	0	60	80	40	10	20
M_5	0	0	0	0	0	0	0	0

Πίνακας 2.22: Χάραγμα γραμμών στον Πίνακα 2.21

Για τη διαδικασία της ανάθεσης, γίνεται αρχικά σάρωση του πίνακα, αναζητώντας σειρές οι οποίες έχουν μόνο ένα μηδενικό. Σε τέτοιου είδους σειρές υπάρχει δυνατότητα επίτευξης βέλτιστης ανάθεσης για μια συγκεκριμένη εργασία. Η πρώτη σειρά έχει ένα μηδενικό στη θέση 3, όποτε η εργασία J_3 δρομολογείται στη μηχανή M_1 με κόστος ανάθεσης 180, διαγράφοντας μετά την ανάθεση τη στήλη 3. Η επόμενη ανάθεση θα γίνει με κόστος 190 για την εργασία J_7 στη μηχανή M_4, αφού η σειρά 4 είναι η μόνη που έχει ένα μόνο μηδενικό. Η δεύτερη και τρίτη σειρά έχουν το ίδιο πλήθος μηδενικών, οπότε η ανάθεση της εργασίας J_4 γίνεται σύμφωνα με το μικρότερο κόστος 180 του αρχικού πίνακα κόστους, που φαίνεται στον Πίνακα 3.1, και θα ανατεθεί στο μηχάνημα M_2. Η επόμενη εργασία, J_5, που θα ανατεθεί, θα είναι στο μηχάνημα M_3 με κόστος 190, ενώ η αμέσως επόμενη ανάθεση είναι και αυτή με ίδιο κόστος ανάθεσης για την εργασία J_8, η οποία δρομολογείται στη μηχανή M_2. Τέλος, οι εργασίες J_1, J_2 και J_6 δρομολογούνται όλες στη μηχανή M_5 με κόστος ανάθεσης 210, 200 και 140 αντίστοιχα. Συνολικά, το τελικό κόστος ανάθεσης εργασιών για τον Πίνακα 3.1 είναι 1,470, ενώ όλες οι αναθέσεις που πραγματοποιήθηκαν παρουσιάζονται συνοπτικά στον Πίνακα 2.25.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	80	40	0	150	100	40	50	80
M_2	60	90	0	0	30	40	120	0
M_3	40	60	100	0	0	50	40	60
M_4	60	80	0	60	70	30	0	20
M_5	0	0	10	10	0	0	0	10

Πίνακας 2.23: Αναδιαμορφωμένος πίνακας κόστους

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	80	40	0	150	100	40	50	80
M_2	60	90	0	0	30	40	120	0
M_3	40	60	100	0	0	50	40	60
M_4	60	80	0	60	70	30	0	20
M_5	0	0	10	10	0	0	0	10

Πίνακας 2.24: Χάραγμα γραμμών στον πίνακα

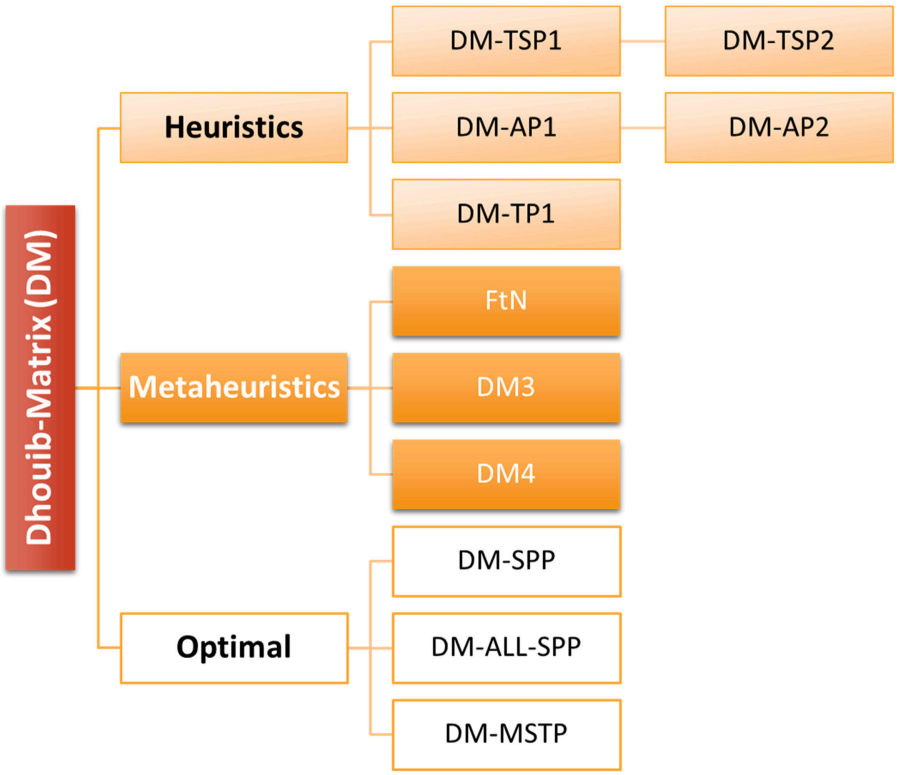
Machine		Job	Cost
M_1	$\rightarrow$	J_3	180
M_2	$\rightarrow$	J_4, J_8	180, 190
M_3	$\rightarrow$	J_5	190
M_4	$\rightarrow$	J_7	180
M_5	$\rightarrow$	J_1, J_2, J_6	210, 200, 140
Total Cost			1,460

Πίνακας 2.25: Πίνακας αναθέσεων τροποποιημένου Ουγγρικού αλγόριθμου

2.4 Dhouib-Matrix-AP2

Το κλασικό πρόβλημα της ανάθεσης έχει τη δυνατότητα να επιλυθεί με πληθώρα διαφορετικών μεθόδων. Προηγουμένως, εξετάστηκαν και αναλύθηκαν αλγόριθμοι βασισμένοι στον Ουγγρικό αλγόριθμο. Ωστόσο, τα περισσότερα προβλήματα βελτιστοποίησης επιλύονται με τη χρήση ευρετικών και μεθευρετικών αλγορίθμων. Ο Dhouib έχει αναπτύξει ένα πλήθος διαφορετικών προσεγγίσεων στοχευμένους στην επίλυση προβλημάτων βελτιστοποίησης. Στο πρόβλημα της ανάθεσης εργασιών αναφέρθηκε για πρώτη φορά το 2022, προτείνοντας τον Dhouib-Matrix-AP1, έναν ευρετικό αλγόριθμο για την επίλυση ισορροπημένων προβλημάτων ανάθεσης [7]. Ωστόσο, το 2023, προτείνει μια καινούργια προσέγγιση, τον αλγόριθμο Dhouib-Matrix-AP2 (DM-AP2), ο οποίος είναι εμπνευσμένος από την προηγούμενη έρευνα [8]. Ο DM-AP2 είναι μια στοχαστική εκδοχή του DM-AP1, που έχει σκοπό την επίλυση μη ισορροπημένων προβλημάτων ανάθεσης.

Όπως συμβαίνει και σε προαναφερόμενο αλγόριθμο της βιβλιογραφίας, ο DM-AP2 πριν ξεκινήσει τη διαδικασία επίλυσης του προβλήματος, μετατρέπει τον πίνακα κόστους σε τετραγωνικό πίνακα μεγέθους NxN. Η μετατροπή πραγματοποιείται προσθέτοντας πλασματικές σειρές που απεικονίζουν κόστη ανάθεσης από ήδη υπάρχοντα VM. Πιο συγκεκριμένα, δημιουργείται μια πλασματική σειρά



Σχήμα 2.1: Κατηγορίες αλγοριθμών του Dhouib [3]

από το μικρότερο κόστος ανάθεσης c_{ij} κάθε εργασίας και προστίθεται στο τέλος του πίνακα $n-m$ φορές, δηλαδή όσες γραμμές χρειαστεί μέχρι ο πίνακας κόστους γίνει τετραγωνικός. Πέρα από τον πίνακα κόστους, ο αλγόριθμος χρησιμοποιεί δύο επιπλέον λίστες αποτελούμενες από τα διαθέσιμα VMs. Στην πρώτη λίστα, ονομαζόμενη Real-Agents, περιέχονται όλα τα επιπλέον VM που προστέθηκαν, ενώ στη δεύτερη λίστα Fictional-Agents αναγράφονται τα VM που χρησιμοποιήθηκαν για να δημιουργηθούν οι πλασματικές σειρές στον πίνακα κόστους.

Σε αντίθεση με τον Ουγγρικό αλγόριθμο, ο DM-AP2 δεν χρειάζεται να επεξεργαστεί περαιτέρω τον πίνακα κόστους. Επομένως, μόλις ο πίνακας C γίνει τετραγωνικός μπορεί να ξεκινήσει η διαδικασία ανάθεσης. Από τον πίνακα κόστους υπολογίζεται το άθροισμα κάθε σειράς και κάθε στήλης, και επιλέγεται το μεγαλύτερο από αυτά. Στην επιλεγμένη σειρά ή στήλη γίνεται αναζήτηση του μικρότερου στοιχείου. Μόλις εντοπιστεί το στοιχείο, στη θέση d_{ij} που βρίσκεται πραγματοποιείται ανάθεση της εργασίας J_i στο VM_j. Η αντίστοιχη σειρά και στήλη που βρισκόταν το κόστος που επιλέχθηκε διαγράφονται, ενώ τα αθροίσματα υπολογίζονται εκ νέου ακολουθούμενα από την επόμενη δρομολόγηση εργασίας. Η επαναλαμβανόμενη διαδικασία τερματίζεται όταν ολοκληρωθούν όλες οι αναθέσεις

εργασιών. Στην περίπτωση που υπάρχει ισότητα μεταξύ αθροισμάτων ή κοστών d_{ij} επιλέγεται τυχαία κάποιο από αυτά.

Όταν πραγματοποιηθεί μια ανάθεση εργασίας είναι απαραίτητο να αξιοποιηθούν οι δυο λίστες που δημιουργήθηκαν στην αρχή για να γίνει σωστή η δρομολόγηση της. Για κάθε ανάθεση όπου ο δείκτης i του VM που επιλέχθηκε είναι μεγαλύτερος του M, δηλαδή του πλήθους των διαθέσιμων μηχανών, τότε το VM ανήκει στις πλασματικές σειρές που προστέθηκαν στην αρχή του αλγορίθμου. Για αυτό, ο δείκτης i ανατρέχεται στην λίστα Fictional-Agents με σκοπό να βρεθεί το αντίστοιχο πραγματικό αναγνωριστικό id του VM. Ωστόσο στην απλή περίπτωση που ο δείκτης i της ανάθεσης δεν έχει ξεπεράσει το πλήθος M, η αντιστοίχιση γίνεται κανονικά.

Με τον αλγόριθμο DM-AP2 είναι βέβαιο πως σε κάθε μηχάνημα θα δρομολογηθεί τουλάχιστον μια εργασία λόγω της τετραγωνικής ιδιότητας που δημιουργείται στην αρχή κάθε προβλήματος. Σε μικρά προβλήματα ο DM-AP2 βρίσκει εύκολα και σύντομα τη βέλτιστη λύση, σε μόλις N επαναλήψεις, όσες είναι και οι εργασίες προς ανάθεση. Ωστόσο σε προβλήματα που εμφανίζεται ισότητα μεταξύ αθροισμάτων, λόγω της τυχαίας επιλογής που πραγματοποιεί ανάμεσα στις σειρές και τις στήλες, αυξάνεται ο χρόνος και η δυσκολία εύρεσης της βέλτιστης λύσης. Κάθε επιλογή ανάθεσης επηρεάζει σημαντικά τις επερχόμενες αναθέσεις, καθώς διαμορφώνονται ανάλογα τα επόμενα αθροίσματα. Για να αντισταθμιστεί η υποβάθμιση της ποιότητας της λύσης, ο αλγόριθμος επαναλαμβάνει το στάδιο της ανάθεσης. Όταν η λύση που προσφέρει ο αλγόριθμος δεν έχει βελτιωθεί σε N επαναλήψεις, ολοκληρώνεται. Ωστόσο, ακολουθώντας τη συγκεκριμένη προσέγγιση αυξάνεται ο τελικός χρόνος εκτέλεσης του. Είναι φανερό πως όσο αυξάνεται το πλήθος των εργασιών, ο χρόνος εκτέλεσης αυξάνεται ανάλογα, οπότε είναι αναγκαίος ο περαιτέρω περιορισμός του.

2.4.1 Εφαρμογή Dhouib-Matrix-AP2 Αλγορίθμου

Στην υποενότητα αυτή θα παρουσιαστεί η αναλυτική εφαρμογή του αλγόριθμου Dhouib-Matrix-AP2 στο παραδειγματικό σενάριο που παρουσιάζει ο Πίνακας 2.20. Το παρόν πρόβλημα που καλείται να επιλύσει ο DM-AP2 είναι ένα μη ισορροπημένο πρόβλημα με 5 μηχανές και 8 εργασίες. Όπως και ο αλγόριθμος του Kumar [1],

Αλγόριθμος 5: Dhouib-Matrix-AP2 Αλγόριθμος - Part 1

```
Input: n = Number of tasks, m = number of VMs, Cost matrix C
Output: Optimal or near Optimal Assignment Plan and total Assignment
        Cost
Create Fictional Agent vector.
Create Fictional Agent position vector.
for Each Column do
    Find minimum cost x_ij of column.
    Push x_ij to Fictional Agent and i to Fictional Agent Position.
for Rows until equal to Number of Columns do
    Copy Fictional Agent to create a square cost matrix.
while N iterations without improvement do
    for Each Row do
        for Each Column do
            Calculate each Column Sum.
            Calculate each Row Sum.
    while There are assignments to be made do
        for Each Column do
            if Task is unmatched then
                Find smallest Column Sum. if Multiple entries with smallest
                 Column Sum exist then
                    Push Task positions j into a highestElements vector.
                    Mark that the smallest Sum derives from Columns.
            if VM is unmatched then
                Find smallest Row Sum. if Multiple entries with smallest Row
                 Sum exist then
                    Push VM positions j into a highestElements vector.
                    Mark that the smallest Sum derives from Rows.
        if highestElements vector is not Empty then
            Select randomly one element from highestElements vector.
        else if If smallest Sum derives from Columns then
            for Each unmatched Column do
                Find smallest element and mark the position.
        for Each unmatched Row do
            Find smallest element and mark the position.
        else if Assignment row is in Fictional Agent then
            Assign Task to VM in Fictional Agent
        Assign Task to VM in Real Agent  for Each Row do
            for Each Column do
                Recalculate each Column Sum.
                Recalculate each Row Sum.
// Continues below
```

Αλγόριθμος 6: Dhouib-Matrix-AP2 Αλγόριθμος - Part 2

```
while N iterations without improvement do
    Calculate total cost of latest Assignment Plan.
    else if The new plan S_new is better than previous plan S then
        S = S_new
    n++
Print final Assignment Plan.
Calculate total Assignment Cost.
```

έτσι και ο DM-AP2 χρειάζεται, πριν προχωρήσει στην επίλυση του προβλήματος, να το μετατρέψει από μη ισορροπημένο σε ισορροπημένο πρόβλημα. Σε αντίθεση όμως με τον Kumar, αντί να χωρίσει το πρόβλημα σε μικρότερα υποπροβλήματα, θα τετραγωνίσει τον πίνακα προσθέτοντας τρεις σειρές.

Για να δημιουργηθούν αυτές οι σειρές σαρώνεται κάθε στήλη του πίνακα διαδοχικά, επιλέγοντας το μικρότερο κόστος κάθε στήλης. Όλα τα κόστη που επιλέχθηκαν, τοποθετούνται διαδοχικά σε μια νέα σειρά του πίνακα, η οποία θα προστεθεί στο κάτω μέρος του ήδη υπάρχοντος πίνακα κόστους. Η νέα σειρά θα προστεθεί τόσες φορές μέχρι ο πίνακας να γίνει τετραγωνικός. Στο συγκεκριμένο πρόβλημα, η νέα σειρά θα επαναληφθεί τρεις φορές. Όσο δημιουργούνται οι νέες σειρές του πίνακα, δημιουργείται παράλληλα μια λίστα Real Agents, που περιέχει τις πραγματικές θέσεις κάθε κόστους, και μια λίστα Fictional Agents, στην οποία αναγράφονται ποιες σειρές δημιουργήθηκαν βοηθητικά για την επίλυση του προβλήματος. Στο παράδειγμα που θα επιλυθεί τυχαίνει οι νέες σειρές που θα προστεθούν στον πίνακα να είναι όμοιες της σειράς που αντιπροσωπεύει το μηχάνημα 5. Οπότε, η λίστα Real Agents περιέχει τις τιμές 5, 5, 5, 5, 5, 5, όπου κάθε στήλη υποδεικνύει σε ποια μηχανή αντιστοιχούν, ενώ στη λίστα Fictional Agents αναγράφονται οι σειρές 6, 7, 8 που είναι αυτές που προστέθηκαν στον πίνακα. Ο νέος πίνακας κόστους, σύμφωνα με τον DM-AP2 είναι πλέον ο Πίνακας 2.26.

Για να ξεκινήσει η διαδικασία ανάθεσης εργασίας, υπολογίζεται πρώτα το άθροισμα κάθε στήλης και κάθε σειράς, τα οποία παρουσιάζονται στους Πίνακες 2.28 και 2.27 αντίστοιχα. Σύμφωνα με τον DM-AP2, επιλέγεται η στήλη ή σειρά με το μεγαλύτερο κόστος, οπότε πρώτα θα επιλεχθεί η πρώτη στήλη με κόστος 2,000. Σαρώντας τη στήλη 1, το μικρότερο στοιχείο εντοπίζεται στην πέμπτη σειρά, συνεπώς η πρώτη ανάθεση θα γίνει με τη δρομολόγηση της εργασίας J_1 στη μηχανή M_5, με κόστος ανάθεσης 210. Επειδή η μηχανή M_5 είναι πραγματική δεν

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	300	250	180	320	270	190	220	260
M_2	290	310	190	180	210	200	300	190
M_3	280	290	300	190	190	200	230	260
M_4	290	300	190	240	250	190	180	210
M_5	210	200	180	170	160	140	160	180
M_6	210	200	180	170	160	140	160	180
M_7	210	200	180	170	160	140	160	180
M_8	210	200	180	170	160	140	160	180

Πίνακας 2.26: Τετραγωνισμένος πίνακας κόστους

χρησιμοποιείται η λίστα Fictional Agents για να γίνει η αντιστοίχιση.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
Row Sum:	1,990	1,870	1,960	1,850	1,400	1,400	1,400	1,400

Πίνακας 2.27: Άθροισμα σειρών

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
Column Sum:	2,000	1,950	1,580	1,610	1,560	1,360	1,570	1,640

Πίνακας 2.28: Άθροισμα στηλών

Αφού διαγραφεί η ανάλογη σειρά και στήλη που έγινε η ανάθεση, υπολογίζονται ξανά τα αθροίσματα κάθε σειράς και στήλης, όπως παρουσιάζονται παρακάτω στον Πίνακα 2.29. Με τον ίδιο τρόπο επιλέγεται η δεύτερη στήλη, διότι έχει το μεγαλύτερο κόστος, 1,750, από όλα τα αθροίσματα, ενώ στη στήλη αυτή το μικρότερο στοιχείο ανήκει στην έκτη γραμμή. Ωστόσο, επειδή η σειρά που γίνεται η ανάθεση ανήκει στη λίστα Fictional Agents, πρέπει να γίνει αντιστοίχιση στο πραγματικό μηχάνημα. Σύμφωνα με τη λίστα Real Agents, η μηχανή M_6 είναι στην πραγματικότητα η μηχανή M_5, οπότε η ανάθεση της εργασίας J_2 θα γίνει και αυτή στο μηχάνημα M_5 με κόστος 200.

	J_2	J_3	J_4	J_5	J_6	J_7	J_8
Row Sum:	1,690	1,580	1,680	1,560	1,190	1,190	1,190
	M_1	M_2	M_3	M_4	M_6	M_7	M_8
Column Sum:	1,750	1,400	1,440	1,400	1,220	1,410	1,460

Πίνακας 2.29: Άθροισμα σειρών και στηλών μετά την ανάθεση του J_1

Επαναλαμβάνοντας τη διαδικασία, η νέα αναζήτηση θα γίνει στην πρώτη σειρά, αφού σύμφωνα με τον Πίνακα 2.30, το άθροισμα της είναι μεγαλύτερο από τα υπό-

	J_3	J_4	J_5	J_6	J_7	J_8
Row Sum:	1,440	1,270	1,390	1,260	990	990
	M_1	M_2	M_3	M_4	M_7	M_8
Column Sum:	1,220	1,270	1,240	1,080	1,250	1,280

Πίνακας 2.30: Άθροισμα σειρών και στηλών μετά την ανάθεση του J_2

λοιπα. Το μικρότερο στοιχείο με κόστος 180, εντοπίζεται στη στήλη που αντιπροσωπεύει την εργασία J_3, δρομολογώντας την στο μηχάνημα M_1. Διαγράφοντας την αντίστοιχη στήλη και σειρά υπολογίζονται πάλι τα αθροίσματα όπως φαίνονται στον Πίνακα 2.31. Είναι ξεκάθαρο πως από την τρίτη σειρά το μικρότερο στοιχείο αντιστοιχίζεται στην εργασία J_4 με κόστος 190, όποτε θα γίνει η ανάθεση της στο μηχάνημα M_3.

	J_4	J_5	J_6	J_7	J_8
Row Sum:	1,080	1,090	1,070	810	810
	M_2	M_3	M_4	M_7	M_8
Column Sum:	950	970	890	1,030	1,020

Πίνακας 2.31: Άθροισμα σειρών και στηλών μετά την ανάθεση του J_3

	J_5	J_6	J_7	J_8
Row Sum:	900	830	640	640
	M_2	M_4	M_7	M_8
Column Sum:	780	670	800	760

Πίνακας 2.32: Άθροισμα σειρών και στηλών μετά την ανάθεση του J_4

Σε αυτό το σημείο έχουν δρομολογηθεί οι μισές εργασίες, αφήνοντας τις εργασίες J_5, J_6, J_7 και J_8 προς δρομολόγηση. Οι πρώτες τέσσερις εργασίες έτυχε να δρομολογηθούν διαδοχικά, ωστόσο δεν είναι ο κανόνας. Όπως φαίνεται από τον νέο πίνακα αθροισμάτων, η πέμπτη ανάθεση θα γίνει για το μηχάνημα M_2. Σε αυτό θα δρομολογηθεί η εργασία J_8, έχοντας το μικρότερο κόστος στη σειρά με τιμή 190. Ακολούθως, θα δρομολογηθεί η εργασία J_7 στο μηχάνημα M_4 με κόστος ανάθεσης 180, αφού, όπως και προηγουμένως, έχει το μικρότερο κόστος στην τέταρτη σειρά.

	J_5	J_6	J_7
Row Sum:	620	460	460
	M_4	M_7	M_8
Column Sum:	570	470	500

Πίνακας 2.33: Άθροισμα σειρών και στηλών μετά την ανάθεση του J_8

	J_5	J_6
Row Sum:	300	300
	M_7	M_8
Column Sum:	320	280

Πίνακας 2.34: Άθροισμα σειρών και στηλών μετά την ανάθεση του J_7

	J_6
Row Sum:	140
	M_8
Column Sum:	140

Πίνακας 2.35: Άθροισμα σειρας και στήλης μετά την ανάθεση του J_5

Χρειάζεται να σημειωθεί πως έχουν απομείνει μόνο δύο μηχανές, οι οποίες σύμφωνα με τη λίστα Fictional Agents ανήκουν και οι δύο σε πλασματικές μηχανές. Επιπλέον στην αρχή επισημάνθηκε πως όλες οι πλασματικές μηχανές προέρχονται από την ίδια μηχανή M_5, οπότε οι τελευταίες δύο εργασίες J_5 και J_6 θα δρομολογηθούν στη μηχανή M_5. Συγκεκριμένα, ακολουθώντας τα βήματα του DM-AP2, πρώτα θα ανατεθεί η εργασία J_5 στη μηχανή M_7, που όπως αναφέρθηκε προηγουμένως αντιστοιχίζεται στη μηχανή M_5, και τέλος θα γίνει ανάθεση στη μηχανή M_8 η εργασία J_6 με κόστος 140. Συνοπτικά, ο αλγόριθμος Dhouib-Matrix-AP2 επιλύει το πρόβλημα με 5 μηχανές και 8 εργασίες σε οκτώ επαναλήψεις, με συνολικό κόστος ανάθεσης 1,450, που είναι και η βέλτιστη λύση. Όλες οι αναθέσεις και το αντίστοιχο κόστος κάθε ανάθεσης παρουσιάζονται αναλυτικά στον Πίνακα 2.36. Οι περισσότερες αναθέσεις δρομολογήθηκαν για το μηχάνημα M_5, καθώς ήταν αυτό που είχε σε μεγαλύτερο βαθμό τα μικρότερα κόστη ανάθεσης.

Machine		Job	Cost
M_1	$\rightarrow$	J_3	180
M_2	$\rightarrow$	J_8	190
M_3	$\rightarrow$	J_4	190
M_4	$\rightarrow$	J_7	180
M_5	$\rightarrow$	J_1, J_2, J_5, J_6	210, 200, 160, 140
Total Cost			1,450

Πίνακας 2.36: Πίνακας αναθέσεων DM-AP2

Κεφάλαιο 3

Υλοποίηση

Σε αυτό το κεφάλαιο θα παρουσιαστεί μια νέα εκδοχή του Ουγγρικού αλγόριθμου για επίλυση μη ισορροπημένων προβλημάτων ανάθεσης εργασιών. Μαζί με τον αλγόριθμο θα γίνει μοντελοποίηση του προβλήματος, χρησιμοποιώντας τον λύτη γραμμικού προγραμματισμού Gurobi για την εύρεση της βέλτιστης λύσης ενός προβλήματος ανάθεσης. Έπειτα, θα γίνει μελέτη της νέας προτεινόμενης εκδοχής του Ουγγρικού αλγορίθμου, εφαρμόζοντας τον σε ένα συγκεκριμένο πρόβλημα ανάθεσης. Τέλος, θα πραγματοποιηθεί σύγκριση του αλγόριθμου μαζί με τους αλγόριθμους που αναλύθηκαν στο προηγούμενο κεφάλαιο πάνω στα αποτελέσματα τεσσάρων προβλημάτων της βιβλιογραφίας.

3.1 Λύτης Gurobi

Το Gurobi είναι ένας υψηλής τεχνολογίας μαθηματικός επιλυτής μαθηματικών προβλημάτων βελτιστοποίησης καθώς και μοντελοποίησης. Ειδικότερα, πρόκειται για έναν λύτη σχεδιασμένο να επιλύει προβλήματα γραμμικού προγραμματισμού (LP), προβλήματα μικτού ακέραιου προγραμματισμού (MIP) και γενικά προβλήματα βελτιστοποίησης, όπως είναι και το πρόβλημα ανάθεσης. Το Gurobi χρησιμοποιείται σε δεκάδες βιομηχανίες, όπως στις τηλεπικοινωνίες, στην υγειονομική περίθαλψη, στην αυτοματοποίηση και φυσικά στην ακαδημαϊκή έρευνα. Η συνεχή υποστήριξη που παρέχεται, η ταχεία ικανότητα επίλυσης προβλημάτων και η επεκτασιμότητα του Gurobi, έχουν καταστήσει τον λύτη ως ένα αξιόπιστο εργαλείο μοντελοποίησης πολύπλοκων προβλημάτων. Το Gurobi διαθέτει αποτελεσματικούς αλγόριθμους και προσεγγίσεις για προβλήματα βελτιστοποίησης μεγάλης κλίμακας, καθιστώντας το κατάλληλο για σύνθετα προβλήματα, αφού είναι ικανό να διαχειριστεί μεγάλα

σύνολα δεδομένων.

Ένας λύτης όπως το Gurobi, λειτουργεί χρησιμοποιώντας τις γενικές προδιαγραφές ενός προβλήματος για την μοντελοποίηση του. Με την εισαγωγή αρχικών δεδομένων έχει τη δυνατότητα εξασφάλισης της βέλτιστης λύσης στο εν λόγω πρόβλημα, είτε το πρόβλημα είναι μεγιστοποίησης είτε ελαχιστοποίησης. Με τη χρήση της βελτιστοποίησης του Gurobi έγινε δυνατή η εύρεση όλων των βέλτιστων λύσεων στα προβλήματα που λύθηκαν από τους υπόλοιπους αλγόριθμους συνδυαστικής βελτιστοποίησης. Το λογισμικό διαθέτει εύχρηστο περιβάλλον εργασίας, υποστηρίζοντας πολλαπλές γλώσσες προγραμματισμού, συμπεριλαμβανομένων της C++, Python, Java και MATLAB. Στην προκειμένη περίπτωση επιλέχθηκε η ανάπτυξη του μοντέλου σε C++.

3.2 Μοντέλο γραμμικού προγραμματισμού

Όπως έχει αναφερθεί σε προηγούμενες ενότητες, το πρόβλημα της ανάθεσης έχει τρεις βασικές μεταβλητές εισόδου: το πλήθος των εργασιών N, το πλήθος των διαθέσιμων μηχανών M και τον πίνακα κόστους C που παρουσιάζει το κόστος ανάθεσης κάθε εργασίας σε κάθε ένα μηχάνημα. Κάθε μια από αυτές τις σταθερές μεταβλητές είναι απαραίτητη για τη σωστή μοντελοποίηση του προβλήματος. Σημαντική είναι επίσης η δημιουργία μιας δυαδικής μεταβλητής απόφασης X_{ij}, με την οποία απεικονίζεται αν μια εργασία J_i έχει ανατεθεί σε ένα VM_j. Αν υπάρχει ανάθεση, η μεταβλητή X_{ij} ισούται με 1, αλλιώς αν δεν υπάρχει ανάθεση παραμένει 0.

Για την κατασκευή του μοντέλου του προβλήματος είναι αναγκαίος ο ορισμός του στόχου και η καταγραφή των περιορισμών και παραδοχών του προβλήματος. Στόχος του προβλήματος ανάθεσης είναι η εύρεση του μικρότερου δυνατού κόστους, είναι δηλαδή ένα πρόβλημα ελαχιστοποίησης, όπως φαίνεται και από την Εξίσωση 3.1. Όπως έχει προαναφερθεί, ο κλασικός Ουγγρικός αλγόριθμος επιτρέπει την αποκλειστική ανάθεση μιας εργασίας σε ένα μόνο VM και το ανάποδο, δηλαδή σε κάθε VM μπορεί να δρομολογηθεί μια μόνο εργασία. Ωστόσο, για την αντιμετώπιση προβλημάτων μη ισορροπημένης ανάθεσης είναι λογικό πως οι προαναφερόμενοι περιορισμοί χρειάζεται να τροποποιηθούν για να ταιριάζουν στα νέα δεδομένα. Με την αρχική υπόθεση να δηλώνει πως το πλήθος N των εργασιών

είναι άνισο και συγκεκριμένα μεγαλύτερο από τον πλήθος M των διαθέσιμων VM, μια νέα παραδοχή θα αφορά τη δρομολόγηση των εργασιών στα VMs. Πλέον σε κάθε VM πρέπει να δρομολογηθεί τουλάχιστον μια εργασία, ώστε ανατεθούν όλες οι εργασίες, χωρίς να υπάρχει δρομολόγηση εργασιών σε κάποιο "dummy" VM. Ο περιορισμός αυτός απεικονίζεται με την Εξίσωση 3.3, ενώ με την Εξίσωση 3.4 παρουσιάζεται ο περιορισμός ανάθεσης μιας εργασίας αποκλειστικά σε ένα VM.

3.3 Υλοποίηση του μοντέλου

Ξεκινώντας τη διαδικασία μοντελοποίησης, χρειάζεται να δημιουργηθεί ένα περιβάλλον Gurobi GRBenv* env, καθώς επίσης και ένα κενό μοντέλο GRBmodel. Ένα μοντέλο μπορεί να περιγράψει ένα μόνο πρόβλημα βελτιστοποίησης και απαρτίζεται από ένα σύνολο μεταβλητών απόφασης, ένα σύνολο περιορισμών και τα γενικά χαρακτηριστικά που περιγράφονται με μαθηματικούς τύπους. Στο συγκεκριμένο πρόβλημα ανάθεσης, η κύρια μεταβλητή του μοντέλου είναι η X_{ij}, που όπως προαναφέρθηκε είναι δυαδική, και ορίζεται στο μοντέλο ως GRBVar**, εκφράζοντας έναν δείκτη σε έναν δισδιάστατο πίνακα μεταβλητών Gurobi. Με αυτόν τον τρόπο η μεταβλητή μιμείται τον πίνακα κόστους C σε μέγεθος, απεικονίζοντας την κατάσταση ανάθεσης μιας εργασίας. Με την συνάρτηση model.addVar() προστίθεται στο μοντέλο κάθε μεταβλητή x_{ij}, ορίζοντας τον τύπο της μεταβλητής ως GRB_BINARY και θέτοντας το κάτω και άνω όριο αντίστοιχα.

Για να προστεθεί ένας περιορισμός στο μοντέλο Gurobi είναι απαιτούμενη η δημιουργία και χρήση μεταβλητών γραμμικών εκφράσεων, τύπου GRBLinExpr. Κάθε περιορισμός που εφαρμόζεται στο μοντέλο διατυπώνεται με τη μορφή μιας μεταβλητής και προστίθεται έπειτα ο περιορισμός. Συγκεκριμένα, η πρώτη μεταβλητή γραμμικής έκφρασης για το μοντέλο αναγράφεται ως το άθροισμα των μεταβλητών απόφασης X_{ij} κάθε στήλης, θέλοντας να υπολογίσει το πλήθος VM_j που ανατίθεται μια εργασία J_i. Η δεύτερη μεταβλητή του μοντέλου παρουσιάζεται με το άθροισμα των μεταβλητών απόφασης X_{ij} κάθε σειράς, που όπως απεικονίζεται και στην Εξίσωση 3.3, καταγράφοντας πόσες εργασίες έχει αναλάβει ένα VM_j. Οι μεταβλητές εφαρμόζονται για κάθε στήλη και γραμμή αντίστοιχα, και έπειτα προσθέτονται στο μοντέλο του προβλήματος με την συνάρτηση model.addConstr(). Σε αυτή τη συνάρτηση ορίζονται οι περιορισμοί κάθε γραμμικής μεταβλητής. Συ-

γκεκριμένα, ορίζεται ο περιορισμός της ισότητας με το 1 για τον πρώτο τύπο μεταβλητών, όπως φαίνεται στην Εξίσωση 3.4, ενώ για τον δεύτερο τύπο μεταβλητής ορίζεται ο περιορισμός μεγαλύτερο ή ίσος του 1 ακολουθώντας την Εξίσωση 3.3.

Ελαχιστοποίηση:

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} X_{ij} \tag{3.1}$$

$$X_{ij} = \begin{cases} 1 & \text{, if the machine i is assigned to task j} \\ 0 & \text{, if the machine i is not assigned to task j} \end{cases} \tag{3.2}$$

Περιορισμοί:

$$\sum_{j=1}^{n} X_{ij} \geq 1, i = 1, 2, ..., m \tag{3.3}$$

$$\sum_{i=1}^{m} X_{ij} = 1, j = 1, 2, ..., n \tag{3.4}$$

Το επόμενο βήμα για τη κατασκευή του μοντέλου ανάθεσης είναι ο προσδιορισμός του στόχου βελτιστοποίησης, ο οποίος είναι η ελαχιστοποίηση του τελικού κόστους ανάθεσης. Το τελικό κόστος του προβλήματος υπολογίζεται από τη Συνάρτηση 3.1, όπου αθροίζεται κάθε κόστος c_{ij} του αρχικού πίνακα κόστους πολλαπλασιασμένο με το αντιστοιχό x_{ij}. Το γινόμενο $c_{ij} * x_{ij}$ θα είναι είτε 0 είτε ίσο με το c_{ij}, αφού το x_{ij} μπορεί να έχει τιμές 0 και 1. Το άθροισμα του γινομένου ορίζεται και αυτό ως μια μεταβλητή γραμμικής έκφρασης GRBLinExpr. Στο μοντέλο ορίζεται ως αντικειμενική συνάρτηση με τη συνάρτηση model.setObjective(), με παραμέτρους το προαναφερόμενο GRBLinExpr, και με τη δεύτερη παράμετρο να GRB_MINIMIZE που υποδηλώνει το στόχο της ελαχιστοποίησης.

Όταν έχουν προστεθεί όλοι οι περιορισμοί, ο στόχος, καθώς και οι μεταβλητές απόφασης του προβλήματος, το μοντέλο είναι έτοιμο για βελτιστοποίηση με τη χρήση της συνάρτησης model.optimize(). Για συγκεκριμένη είσοδο πίνακα κόστους C και μέγεθος προβλήματος, το Gurobi καταφέρνει να υπολογίσει τη βέλτιστη λύση του προβλήματος για το μοντέλο που κατασκευάστηκε. Για το πρόβλημα της ανάθεσης, πέρα από την εμφάνιση του τελικού βέλτιστου κόστους, είναι δυνατή η εμφάνισης μιας αναλυτικής λίστας ανάθεσης, που παρουσιάζει σε ποιο VM ανατέθηκε κάθε εργασία και με τι κόστος πραγματοποιήθηκε αυτή η ανάθεση.

Αλγόριθμος 7: Δημιουργία και επίλυση μοντέλου γραμμικού προγραμματισμού με τον Gurobi

Input: n = Number of tasks, m = number of VMs, Cost matrix C
Output: Optimal or near Optimal Assignment Plan and total Assignment Cost
Create a GRB Environment variable env.
Create a GRB Model model .
Initialize size of problem: Number of Rows (VMs) and Number of Columns (Tasks).
Create a GRB pointer to pointer variable X in order to use as a dynamic array, with size row x column.
Add a binary decision X_{ij} to Gurobi model.
Create a GRB linear expression Machine.
for Each Row do
    machine = 0
    for Each Column do
        $machine+ = x_{ij}$
    Add model constrain machine >= 1
for Each Column do
    Create a GRB linear expression Task and initialize as 0.
    for Each Row do
        $Task+ = x_{ij}$
    Add model constrain task = 1
Update model.
Open data file.
Create a GRB linear expression Total.
for Each Row do
    for Each Column do
        Retrieve from file each respective cost.
        $Total+ = Cost_{ij} * X_{ij}$
Close file.
Set Model Objective to minimize total.
Optimize Model.
if Model is optimized then
    Get Assignment Plan.
    Get Assignment Total Cost.

3.4 Προτεινόμενος Ούγγρικος αλγόριθμος

Η παρούσα διπλωματική προτείνει μια μετεξέλιξη του Ουγγρικό αλγόριθμο, έχοντας ως θεμέλια τον βελτιωμένο Ουγγρικό αλγόριθμο των Rabbani, Khan και Quddoos [2]. Ο προτεινόμενος αλγόριθμος έχει σκοπό να μειώσει τον χρόνο εκτέλεσης, ελαττώνοντας τις επαναλαμβανόμενες αναζητήσεις που πραγματοποιεί ο αλγόριθμος στον πίνακα κόστους κατά τη διάρκεια της δρομολόγησης κάθε ερ-

γασίας. Σε γενικές γραμμές, η δομή του προτεινόμενου αλγορίθμου είναι όμοια με των Rabbani και Khan, με τη μόνη διαφοροποίηση να αφορά τον τρόπο ανάθεσης των εργασιών.

Για να πραγματοποιηθεί η διαδικασία ανάθεσης είναι απαραίτητη η επεξεργασία του πίνακα κόστους, ώστε να γίνει εφικτή η βέλτιστη ή σχεδόν βέλτιστη επίλυση του προβλήματος. Η ακόλουθη επεξεργασία είναι πανομοιότυπη με τον προαναφερόμενο Ουγγρικό αλγόριθμο. Ο αλγόριθμος ξεκινάει αναζητώντας στον πίνακα C το μικρότερο κόστος c_{ij} κάθε σειράς και αφαιρώντας το από την εκάστοτε σειρά, δημιουργώντας σε κάθε μια ένα τουλάχιστον μηδενικό. Για να εγγυηθεί ο αλγόριθμος την ύπαρξη ενός τουλάχιστον μηδενικού και για τις στήλες, η παραπάνω διαδικασία επαναλαμβάνεται για κάθε στήλη. Ακολουθώντας την ολοκλήρωση της δημιουργίας μηδενικών στον νέο πίνακα κόστους C_{new}, ο αλγόριθμος συνεχίζει με έλεγχο του επεξεργασμένου αυτού πίνακα, προκειμένου να διαπιστωθεί αν είναι δυνατή η επίτευξη ανάθεσης για το πρόβλημα.

Όπως έγινε σε όλους τους αλγόριθμους της βιβλιογραφίας που βασίζονται στον Ουγγρικό Αλγόριθμο, ο έλεγχος εύρεσης λύσης πραγματοποιείται με τη χάραξη γραμμών καλύπτοντας όλα τα μηδενικά του πίνακα. Η βέλτιστη επίλυση του προβλήματος επιτυγχάνεται με τη χρήση ελάχιστων γραμμών κάλυψης στον επεξεργασμένο πίνακα. Στην περίπτωση που δεν είναι εφικτή η εύρεση λύσης, ο αλγόριθμος θα προχωρήσει σε περαιτέρω επεξεργασία του πίνακα C_{new}. Αυτή τη φορά λαμβάνονται υπόψη οι γραμμές που καλύπτουν το πρόβλημα για την μετατροπή του πίνακα C_{new}. Για μια ακόμα φορά γίνεται αναζήτηση στον πίνακα για το μικρότερο κόστος, ακάλυπτο από τις χαραγμένες γραμμές, το οποίο προστίθεται στα κόστη που βρίσκονται σε τομές δύο γραμμών. Το ίδιο κόστος αφαιρείται από τα υπολειπόμενα μη καλυπτόμενα κόστη, δημιουργώντας τον νέο πίνακα κόστους. Μετά την ολοκλήρωση του προηγούμενου βήματος, ελέγχεται και πάλι ο ανανεωμένος πίνακας. Η διαδικασία αυτή επαναλαμβάνεται έως ότου το πλήθος των γραμμών που καλύπτουν τα μηδενικά του πίνακα να είναι ίσο με το πλήθος των VM.

Όταν σημανθεί πως η εύρεση της βέλτιστης λύσης στο πρόβλημα είναι εφικτή, ο αλγόριθμος θα προχωράει στο τελικό στάδιο ανάθεσης, όπου κάθε εργασία δρομολογείται σε ένα VM. Όπως έχει αναφερθεί, κάθε εργασία μπορεί να ανατεθεί σε ένα μόνο VM, ωστόσο κάθε VM μπορεί να διαχειριστεί τουλάχιστον μια εργα-

σία. Η ανάθεση ξεκινάει αναζητώντας τις μη χρησιμοποιημένες σειρές που έχουν αποκλειστικά ένα μόνο μηδενικό. Για αυτές τις σειρές η ανάθεση εργασίας γίνεται στη θέση του μηδενικού, με κόστος σύμφωνα με τον αρχικό πίνακα κόστους, και έπειτα η σειρά μαρκάρεται ως χρησιμοποιημένη. Η αναζήτηση σειρών με ένα μηδενικό τερματίζεται όταν δεν υπάρχουν άλλες σειρές που καλύπτουν αυτή την προϋπόθεση.

Αν σε αυτό το σημείο δεν έχουν ανατεθεί όλες οι εργασίες του προβλήματος, τότε η διαδικασία της ανάθεσης συνεχίζεται. Κάθε μη ανατεθειμένη εργασία δρομολογείται σε ένα μη μαρκαρισμένο VM, που έχει το μικρότερο κόστος στον αρχικό πίνακα. Αφού γίνει η ανάθεση, η στήλη της εργασίας διαγράφεται από τον πίνακα C_{new} και η σειρά του VM που χρησιμοποιήθηκε μαρκάρεται ως χρησιμοποιημένη. Μέχρι να δρομολογηθεί από μια εργασία σε κάθε VM, αυτή η διαδικασία ανάθεσης επαναλαμβάνεται. Άμα υπάρχουν μη ανατιθέμενες εργασίες όταν όλες οι σειρές επισημανθούν ως χρησιμοποιημένες, ο αλγόριθμος συνεχίζει στο τελικό στάδιο ανάθεσης. Σε αυτό το στάδιο, η ανάθεση κάθε εργασίας εξαρτάται αποκλειστικά και μόνο από το μικρότερο κόστος που έχει στον αρχικό πίνακα κόστους C, χωρίς να ληφθεί υπόψη πόσες εργασίες έχουν δρομολογηθεί στο αντίστοιχο VM. Όπως και με τους άλλους αλγορίθμους έτσι και με αυτόν, όταν ολοκληρωθεί η ανάθεση όλων των εργασιών υπολογίζεται το τελικό κόστος σύμφωνα με τον αρχικό πίνακα κόστους C.

Σε γενικές γραμμές ο προτεινόμενος αλγόριθμος είναι ίδιος με τον αλγόριθμο στο [2], με εξαίρεση τον τρόπο που πραγματοποιείται η τελική ανάθεση. Σε προβλήματα μικρού μεγέθους, τα αποτελέσματα και των δύο αλγορίθμων είναι πανομοιότυπα, με πολύ μικρές αποκλίσεις μεταξύ τους. Ωστόσο όσο αυξάνεται το μέγεθος των προβλημάτων γίνεται εμφανής η διαφορά μεταξύ των δύο αλγορίθμων.

3.5 Εφαρμογή προτεινόμενου Ουγγρικού αλγόριθμου

Σε αυτή την ενότητα θα γίνει εφαρμογή του προτεινόμενου Ουγγρικού αλγόριθμου στο παραδειγματικό σενάριο που απεικονίζεται στον Πίνακα 3.1. Στα πρώτα στάδια επεξεργασίας του πίνακα κόστους, ο προτεινόμενος αλγόριθμος είναι πανομοιότυπος με τον τροποποιημένο αλγόριθμο του Rabbani [2]. Η κύρια διαφορά τους, όπως θα παρουσιαστεί στη συνέχεια είναι στον τρόπο επιλογής ανάθεσης

Αλγόριθμος 8: Προτεινόμενος Ούγγρικος αλγόριθμος

```
Input: n = Number of tasks, m = number of VMs, Cost matrix C
Output: Optimal or near Optimal assignment plan
for each column of C matrix do
    Find the minimum cost c_ij in column and subtract c_ij from each cost in
     the column.
end
for each row of Cost matrix do
    Find the minimum cost c_ij in row and subtract c_ij from each cost in the
     row.
end
while The number of rows is not equals the number of lines do
    Draw the minimum number of lines to cover every zero in C matrix.
    Find the smallest uncovered cost c_ij.
    Add c_ij to the costs located at each intersection of lines.
    Subtract c_ij from each uncovered costs in the C matrix.
end
while There is not change do
    for Each unmached task do
        Count the number of Zeros in each row
        if There is only one Zero then
            Assign Task to VM
            Mark the VM as used.
        end
    end
end
while Assignment is not Completed do
    for Each unassigned Task do
        if VM is not used then
            Find minimum cost c_ij in column.
            Assign the task to the VM where the minimum cost is found.
        end
    end
    if All VMs are marked as used then
        for Each unassigned Task do
            Find minimum cost c_ij in column.
            Assign the task to the VM where the minimum cost is found.
        end
    end
end
Print Assignment Plan.
Calculate total cost of assignment plan.
```

των εργασιών. Ο αλγόριθμος, λοιπόν, ξεκινάει με τον ίδιο τρόπο, τροποποιώντας τον αρχικό πίνακα κόστους με απώτερο σκοπό τη δημιουργία μηδενικών. Αρχικά θα δημιουργηθούν μηδενικά κατά μήκος των στηλών, αφαιρώντας το μικρότερο κόστος c_{ij} κάθε στήλης από τα υπόλοιπα της ίδιας στήλης. Με αυτόν τον τρόπο,

πλέον κάθε στήλη έχει ένα τουλάχιστον μηδενικό. Στη συνέχεια, όπως έγινε και στον τροποποιημένο Ουγγρικό αλγόριθμο, επαναλαμβάνεται η ίδια διαδικασία για κάθε σειρά αντίστοιχα, με το αποτέλεσμα να παρουσιάζεται στον Πίνακα 3.2.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	300	250	180	320	270	190	220	260
M_2	290	310	190	180	210	200	300	190
M_3	280	290	300	190	190	200	230	260
M_4	290	300	190	240	250	190	180	210
M_5	210	200	180	170	160	140	160	180

Πίνακας 3.1: Παράδειγμα προβλήματος μη ισορροπημένης ανάθεσης

Ακολουθώντας την επεξεργασία του πίνακα, πραγματοποιείται ο έλεγχος επίτευξης βέλτιστης λύσης με τη χρήση γραμμών. Στον πίνακα που δημιουργήθηκε χαράζονται γραμμές σκοπεύοντας να καλύψουν όλα τα μηδενικά του πίνακα. Για να μπορέσει ο αλγόριθμος να προχωρήσει στο επόμενο στάδιο ανάθεσης χρειάζεται οι γραμμές που θα χαραχθούν στον πίνακα να είναι όσες και οι σειρές του προβλήματος. Όπως φαίνεται στον Πίνακα 3.3, διότι οι χαραγμένες γραμμές είναι τέσσερις είναι απαραίτητο να γίνει περαιτέρω επεξεργασία του πίνακα κόστους. Το μικρότερο μη καλυπτόμενο κόστος στον πίνακα βρίσκεται στη θέση c_{35} και έχει την τιμή 10, η οποία αφαιρείται από τα υπόλοιπα μη καλυπτόμενα κόστη. Στις θέσεις c_{53}, c_{54} και c_{58} που βρίσκονται σε τομές δύο γραμμών, προστίθεται η τιμή αυτή, διαμορφώνοντας τον νέο Πίνακα 3.3.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	90	50	0	150	110	50	60	80
M_2	70	100	0	0	40	50	130	0
M_3	50	70	100	0	10	60	50	60
M_4	70	90	0	60	80	40	10	20
M_5	0	0	0	0	0	0	0	0

Πίνακας 3.2: Πίνακας μετά τη δημιουργία μηδενικών

Όπως φαίνεται στον νέο πίνακα, το πλήθος των μηδενικών είναι μικρότερο σε σχέση με την προηγούμενη μορφή του πίνακα, με την κατανομή τους όμως να είναι επίσης διαφορετική. Πλέον, χαράζοντας γραμμές στον πίνακα, παρατηρείται πως ο ελάχιστος αριθμός γραμμών είναι ίσως με το πλήθος των μηχανών του προβλήματος. Συνεπώς, το πρόβλημα είναι έτοιμο να μεταβεί στο τελικό στάδιο του αλγορίθμου, που είναι η ανάθεση. Όπως έχει ήδη επισημανθεί, το στάδιο της ανάθεσης είναι

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	90	50	0	150	110	50	60	80
M_2	70	100	0	0	40	50	130	0
M_3	50	70	100	0	10	60	50	60
M_4	70	90	0	60	80	40	10	20
M_5	0	0	0	0	0	0	0	0

Πίνακας 3.3: Χάραγμα γραμμών στον Πίνακα 3.2

το τμήμα του αλγορίθμου που διαφοροποιεί τον προτεινόμενο αλγόριθμο από τον αλγόριθμο που περιγράφεται από τον Rabbani [2].

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	80	40	0	150	100	40	50	80
M_2	60	90	0	0	30	40	120	0
M_3	40	60	100	0	0	50	40	60
M_4	60	80	0	60	70	30	0	20
M_5	0	0	10	10	0	0	0	10

Πίνακας 3.4: Δέυτερο χάραγμα γραμμών στον πίνακα

Αρχικά, ο αλγόριθμος ξεκινάει με την ανάθεση της εργασίας J_3 στο μηχάνημα M_1, καθώς είναι η πρώτη σειρά που έχει ένα μοναδικό μηδενικό. Διαγράφοντας τη στήλη 3 του πίνακα, συνεχίζεται ο διαδοχικός έλεγχος των σειρών. Η επόμενη ανάθεση θα γίνει στο μηχάνημα M_4, στο οποίο θα δρομολογηθεί η εργασία J_7, με κόστος ανάθεσης 190. Στην επόμενη σάρωση του πίνακα διαπιστώνεται ότι δεν υπάρχει πλέον άλλη σειρά στην οποία να βρίσκεται ένα μόνο μηδενικό. Για αυτό αρχίζουν να εξετάζονται οι στήλες, παραβλέποντας τις σειρές που έχει γίνει ανάθεση μέχρι να έχουν δρομολογηθεί εργασίες σε όλες τις μηχανές. Με αυτό τον τρόπο βεβαιώνεται ο αλγόριθμος πως κάθε μηχανή έχει αναλάβει τουλάχιστον μια εργασία. Η πρώτη στήλη που δεν έχει γίνει ακόμα ανάθεση είναι για την εργασία J_1. Η ανάθεση της εργασίας θα γίνει σύμφωνα με το μικρότερο κόστος ανάθεσης του αρχικού πίνακα κόστους C, που στην προκειμένη περίπτωση έχει κόστος 210 για το μηχάνημα M_5.

Από τις προηγούμενες αναθέσεις δεν ελέγχονται πλέον οι σειρές που αντιστοιχούν στις μηχανές M_1, M_4 και M_5 μέχρι να γίνει ανάθεση και στις υπόλοιπες δύο σειρές. Από τις μη ανατεθειμένες εργασίες, το μικρότερο κόστος στον αρχικό πίνακα κόστους βρίσκεται στη θέση c_{24} με τιμή 180, οπότε η εργασία J_4 δρομο-

λογείται στο M_2. Πλέον, έχει μείνει μια μόνο μηχανή, η M_3, που δεν έχει αναλάβει κάποια εργασία. Επιλέγεται η εργασία J_5, η οποία, όπως έγινε προηγουμένως, έχει το μικρότερο κόστος στον πίνακα, και δρομολογείται στη μηχανή. Η ανάθεση της εργασίας J_5 γίνεται με κόστος 190 το οποίο προστίθεται στο συνολικό κόστος του προβλήματος. Σε αυτό το σημείο ο αλγόριθμος έχει εγγυηθεί πως κάθε μηχάνημα έχει αναλάβει από μια εργασία, όπως εξάλλου προϋποθέτει ο Ουγγρικός αλγόριθμος. Οι περισσευούμενες εργασίες αναθέτονται και αυτές διαδοχικά, σύμφωνα με το μικρότερο κόστος σε κάθε στήλη, χωρίς να υφίσταται πλέον ο περιορισμός επιλογής μηχανής. Οι εργασίες J_2, J_6 και J_8 δρομολογούνται όλες στο μηχάνημα M_5 με κόστη 200, 140 και 180 αντίστοιχα. Προσθέτοντας όλα τα κόστη ανάθεσης κάθε εργασίας οδηγούν το συνολικό κόστος επίλυσης του προβλήματος να έχει τιμή 1,460. Όλες οι αναθέσεις και τα αντίστοιχα κόστη ανάθεσης παρουσιάζονται αναλυτικά στον Πίνακα 3.5.

Machine		Job	Cost
M_1	$\rightarrow$	J_3	180
M_2	$\rightarrow$	J_4	180
M_3	$\rightarrow$	J_5	190
M_4	$\rightarrow$	J_7	180
M_5	$\rightarrow$	$J_1,\ J_2,\ J_6,\ J_8$	210, 200, 140, 180
Total Cost			1,460

Πίνακας 3.5: Πίνακας αναθέσεων

3.5.1 Σύγκριση αποτελεσμάτων

Μέχρι στιγμής, όλοι οι αλγόριθμοι της βιβλιογραφικής ανασκόπησης και του Κεφαλαίου 3 έχουν μελετηθεί υλοποιώντας το πρόβλημα που παρουσιάζεται στον Πίνακα 3.1. Σε αυτό το σημείο θα γίνει σύγκριση των μεταξύ τους αποτελεσμάτων, καθώς και με τη βέλτιστη λύση που δόθηκε εισάγοντας τον πίνακα κόστους στο μαθηματικό μοντέλο. Το τελικό κόστος ανάθεσης που έχει κάθε αλγόριθμος για το συγκεκριμένο παράδειγμα παρουσιάζεται συνοπτικά στον Πίνακα 3.6.

Δίνοντας ως είσοδο το πρόβλημα στο μοντέλο που σχεδιάστηκε με τον λύτη Gurobi, η βέλτιστη λύση του προβλήματος λαμβάνει την τιμή 1,450. Ο μόνος αλγόριθμος που κατάφερε να βρει τη βέλτιστη ανάθεση είναι ο αλγόριθμος DM-AP2, ο οποίος, όπως φαίνεται στον Πίνακα 3.7, πραγματοποιεί τις βέλτιστες αναθέ-

σεις που υποδεικνύει ο λύτης. Αξίζει να σημειωθεί, πως ο αλγόριθμος DM-AP2 κατέληξε σε λύση, σε οκτώ μόλις επαναλήψεις, όσες δηλαδή είναι οι εργασίες προς δρομολόγηση. Την αμέσως επόμενη βέλτιστη λύση προσφέρει ο προτεινόμενος Ουγγρικός αλγόριθμος που αναλύθηκε προηγουμένως, με απόκλιση κόστους 10. Έπειτα ακολουθεί ο Τροποποιημένος Ουγγρικός αλγόριθμος, ενώ τη χειρότερη ποιότητα λύσης φέρει η εκδοχή του Ουγγρικού αλγόριθμου που προτείνει ο Kumar. Από τις τρεις εκδοχές του Ουγγρικού αλγόριθμου για επίλυση μη ισορροπημένων προβλημάτων, ο προτεινόμενος αλγόριθμος εμφανίζει μέχρι στιγμής τις καλύτερες επιδόσεις.

Παραλλαγή Ουγγρικού αλγόριθμου από Kumar	1,550
Τροποποιημένος Ουγγρικός αλγόριθμος	1,470
Προτεινόμενος Ουγγρικός Αλγόρθμος	1,460
Dhouib-Matrix-AP2	1,450

Πίνακας 3.6: Πίνακας αναθέσεων για το πρόβλημα 3.1

Machine		Job	Cost
M_1	$\rightarrow$	J_3	180
M_2	$\rightarrow$	J_8	190
M_3	$\rightarrow$	J_4	190
M_4	$\rightarrow$	J_7	180
M_5	$\rightarrow$	$J_1,\ J_2,\ J_5,\ J_6$	210, 200, 160, 140
Total Cost			1,450

Πίνακας 3.7: Βέλτιστη ανάθεση

Με εξαίρεση τον προτεινόμενο Ουγγρικό αλγόριθμο που παρουσιάζεται εδώ, οι υπόλοιποι αλγόριθμου συγκρίνονται συχνά στη βιβλιογραφία. Συγκεκριμένα, στο [9] οι αλγόριθμοι του Kumar και Rabbani συγκρίνονται με βάση το ίδιο πρόβλημα με έναν αλγόριθμο βασισμένο σε Βελτιστοποίησης Μυρμηγκοφωλιάς (ACO). Περαιτέρω μελέτη της συμπεριφοράς των αλγορίθμων, πραγματοποιείται στη δημοσίευση [10], η οποία παρέχει έναν κατάλογο αποτελούμενο από τρία επιπλέον δοκιμαστικά προβλήματα πέρα από το πρόβλημα 3.1 που μόλις αναλύθηκε. Τα προβλήματα αυτά απεικονίζονται στους Πίνακες 3.8, 3.10 και 3.12αντίστοιχα.

Οι περισσότεροι από τους αλγόριθμους που έχουν αναφερθεί μέχρι στιγμής, έχουν μελετηθεί με είσοδο τα τρία αυτά προβλήματα. Παρακάτω θα ακολουθήσει εφαρμογή και σύγκριση του προτεινόμενου Ουγγρικού αλγόριθμου για τα παραπάνω προβλήματα, εξετάζοντας διεξοδικότερα τον αλγόριθμο. Για το πρόβλημα

	J_1	J_2	J_3	J_4	J_5	J_6	J_7
M_1	30	25	18	32	27	19	22
M_2	29	31	19	18	21	20	30
M_3	28	29	30	19	19	22	23
M_4	29	30	19	24	25	19	18
M_5	21	20	18	17	16	14	16

Πίνακας 3.8: Δοκιμαστικό πρόβλημα με 7 εργασίες και 5 μηχανές

Παραλλαγή Ουγγρικού αλγόριθμου από Kumar	136
Τροποποιημένος Ουγγρικός αλγόριθμος	128
Προτεινόμενος Ουγγρικός Αλγόρθμος	128
Dhouib-Matrix-AP2	128
Βέλτιστη Λύση Προβλήματος	128

Πίνακας 3.9: Πίνακας αναθέσεων για το πρόβλημα 3.8

με πίνακα κόστους τον Πίνακα 3.8, το βέλτιστο κόστος ανάθεσης σύμφωνα με το γραμμικό μοντέλο είναι 128, ενώ όπως φαίνεται στον Πίνακα 3.9 όλοι οι αλγόριθμοι πέρα από την εκδοχή του Kumar καταφέρνουν να επιλύσουν το πρόβλημα με το ελάχιστο δυνατό κόστος ανάθεσης. Ωστόσο, δεν συμβαίνει το ίδιο για τα άλλα δυο δοκιμαστικά παραδείγματα. Το πρόβλημα ανάθεσης δέκα εργασιών σε έξι μηχανές μπορούν να το επιλύσουν βέλτιστα μόνο οι αλγόριθμοι DM-AP2 και ο αλγόριθμος ACO. Με τη βέλτιστη λύση του προβλήματος να λαμβάνει την τιμή 65, ο προτεινόμενος Ουγγρικός και ο τροποποιημένος Ουγγρικός αλγόριθμος έχουν απόκλιση της τάξης 8 και 7 αντίστοιχα, με τα αποτελέσματα τους να φαίνονται στον Πίνακα 3.11. Τέλος, για το πρόβλημα με δέκα εργασίες και επτά μηχανές του Πίνακα 3.13, την καλύτερη ποιότητα λύσεων έχει ο προτεινόμενος αλγόριθμος μαζί με τον γενετικά τροποποιημένο αλγόριθμο, που παρουσιάζεται στη δημοσίευση [10], με τελικό κόστος ανάθεσης 69, ενώ ακολουθεί ο DM-AP2 με κόστος ανάθεσης 70.

Μελετώντας τα αποτελέσματα από τα παραπάνω δοκιμαστικά δεδομένα, πα-

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
M_1	10	2	14	9	6	7	21	32	18	11
M_2	7	12	9	3	5	6	9	16	54	12
M_3	4	8	6	12	21	9	21	14	45	13
M_4	21	9	12	9	32	10	19	25	16	10
M_5	10	12	30	15	12	17	30	12	12	9
M_6	15	7	34	17	7	16	14	17	9	5

Πίνακας 3.10: Δοκιμαστικό πρόβλημα με 10 εργασίες και 6 μηχανές

Παραλλαγή Ουγγρικού αλγόριθμου από Kumar	81
Τροποποιημένος Ουγγρικός αλγόριθμος	72
Προτεινόμενος Ουγγρικός Αλγόρθμος	73
Dhouib-Matrix-AP2	65
Βέλτιστη Λύση Προβλήματος	65

Πίνακας 3.11: Πίνακας αναθέσεων για το πρόβλημα 3.10

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
M_1	21	11	16	9	15	10	12	32	26	16
M_2	14	15	20	10	16	3	6	9	21	14
M_3	9	17	11	31	21	16	7	9	10	11
M_4	16	23	8	15	10	3	6	3	20	23
M_5	12	40	14	36	9	21	14	19	4	13
M_6	8	18	9	42	8	11	19	9	32	20
M_7	21	9	12	9	32	10	19	25	16	10

Πίνακας 3.12: Δοκιμαστικό πρόβλημα με 10 εργασίες και 7 μηχανές

ρατηρείται πως σε γενικές γραμμές ο πιο αποδοτικός αλγόριθμος είναι ο DM-AP2, καθώς στις περισσότερες περιπτώσεις επιτυγχάνει την εύρεση του βέλτιστου κόστους ανάθεσης. Όσον αφορά τους υπόλοιπους αλγόριθμους, δεν αποτελεί έκπληξη, λόγω των ομοιοτήτων που έχουν, η παρόμοια συμπεριφορά που έχει ο αλγόριθμος που προτείνεται με τον τροποποιημένο Ουγγρικό αλγόριθμο. Οι δύο αυτοί αλγόριθμοι, και στα τέσσερα δοκιμαστικά παραδείγματα, παρέχουν μικρές σε απόκλιση τελικές λύσεις. Από την άλλη, η εκδοχή του Kumar, από τη φύση του αλγόριθμου, δεν παρέχει σε καμία περίπτωση επιθυμητά αποτελέσματα, με τα τελικά κόστη ανάθεσης που προσφέρει να έχουν την χειρότερη ποιότητα λύσης από όλους τους άλλους αλγόριθμους.

Παραλλαγή Ουγγρικού αλγόριθμου από Kumar	95
Τροποποιημένος Ουγγρικός αλγόριθμος	82
Προτεινόμενος Ουγγρικός Αλγόρθμος	69
Dhouib-Matrix-AP2	70
Βέλτιστη Λύση Προβλήματος	69

Πίνακας 3.13: Πίνακας αναθέσεων για το πρόβλημα 3.12

Κεφάλαιο 4

Υπολογιστική μελέτη

Σε αυτό το κεφάλαιο θα παρουσιαστεί η υπολογιστική μελέτη όλων των αλγορίθμων που αναλύθηκαν στα Κεφάλαια 2 και 3. Θα παρουσιαστούν δυο διαφορετικοί τρόποι εφαρμογής και σύγκρισης των αλγορίθμων. Αρχικά, θα γίνει ανάλυση της εκτέλεσης των αλγορίθμων που πραγματοποιήθηκε τοπικά. Ο δεύτερος τρόπος εφαρμογής των αλγορίθμων έγινε προσομοιώνοντας τους σε ένα περιβάλλον νέφους, με τη βοήθεια της εργαλειοθήκης CloudSim. Κατόπιν, θα εξεταστούν όλα τα αποτελέσματα εκτέλεσης των αλγορίθμων για πολλαπλά σύνολα δεδομένων.

4.1 Εφαρμογή αλγορίθμων

Στην πρώτη ενότητα του κεφαλαίου θα πραγματοποιηθεί εφαρμογή των αλγορίθμων που αναλύθηκαν στην βιβλιογραφική ανασκόπηση του Κεφαλαίου 2, καθώς και ο προτεινόμενος Ουγγρικός αλγόριθμος που παρουσιάζεται στο Κεφάλαιο 3. Παρακάτω θα ακολουθήσει αναλυτική παρουσίαση της εκτέλεσης των προαναφερόμενων αλγορίθμων για συγκεκριμένα σύνολα δεδομένων. Ωστόσο, πριν ξεκινήσει η ανάλυση των αποτελεσμάτων των αλγορίθμων της βιβλιογραφίας και του προτεινόμενου αλγορίθμου, είναι απαραίτητο να δημιουργηθεί μια βάση για τη σύγκριση τους. Όπως προαναφέρθηκε, όλοι οι αλγόριθμοι εκτέλεσαν τα ίδια σύνολα δεδομένων, παρόλα αυτά η σύγκριση θα υφίσταται λανθασμένη αν δεν ληφθεί υπόψη η βέλτιστη λύση για κάθε σενάριο που εκτελέστηκε. Για κάθε πρόβλημα που επιλύσαν οι αλγόριθμοι, έλυσε και το μοντέλο γραμμικού προγραμματισμού στον λύτη Gurobi, καθορίζοντας το σημείο αναφοράς για τους υπόλοιπους αλγόριθμους.

Συγκεκριμένα οι αλγόριθμοι που θα συγκριθούν παρουσιάζονται στην παρακάτω λίστα.

- Παραλλαγή Ουγγρικού αλγόριθμου από Kumar
- Τροποποιημένος Ουγγρικός αλγόριθμος από Rabbani
- Προτεινόμενος Ουγγρικός Αλγόριθμος
- Dhouib-Matrix-AP2

4.1.1 Σύνολα δεδομένων

Για να πραγματοποιηθεί σωστά μια σύγκριση αλγορίθμων συνδυαστικής βελτιστοποίησης είναι θεμιτό να χρησιμοποιηθεί το ίδιο σύνολο δεδομένων για όλους τους αλγόριθμους. Στα πλαίσια της διπλωματικής εργασίας κατασκευάστηκαν ορισμένα σύνολα δεδομένων, σκοπεύοντας να προσομοιώσουν τους πίνακες κόστους κάθε προβλήματος. Ο αλγόριθμος κατασκευής δεδομένων επιλέγει τυχαία κάθε κόστος διαδοχικά, ακολουθώντας τον Αλγόριθμο 9. Η τιμή που επιλέγεται κυμαίνεται μεταξύ των τιμών 50 και 200, με σκοπό ο πίνακας κόστους που θα δημιουργηθεί να περιέχει ευρεία γκάμα τιμών ανάθεσης. Η τιμή που επιλέγεται για κάθε εργασία απεικονίζει το κόστος ανάθεσης της σε ένα συγκεκριμένο VM. Η ίδια εργασία έχει διαφορετικό κόστος ανάθεσης σε κάθε VM, λαμβάνοντας υπόψη τις διαφορετικές προδιαγραφές που έχουν τα VM μεταξύ τους. Παράλληλα με την επιλογή κόστους, γίνεται η αποθήκευση του πίνακα σε ένα αρχείο της μορφής "dataset_XX.txt", το οποίο στη συνέχεια παρέχεται ως αρχείο εισόδου σε όλους τους αλγόριθμους που υλοποιήθηκαν.

Αλγόριθμος 9: Αλγόριθμος παραγωγής πίνακα κόστους

```
Input: N = Number of tasks, M = number of VMs, Output name file
Output: File dataset_XX.txt containing the Cost Matrix
fileReader ← openFile(dataset_XX.txt)
Initiate random seed
for i = 0, i < M do
    for j = 0, j < N do
        c_ij = (rand() % (upper - lower + 1)) + lower
        fileReader ← write(c_ij)
        j++
    i++
fileReader ← closeFile(dataset_XX.txt)
```

Εκτελώντας τον Αλγόριθμο 9 με διαφορετικές εισόδους εργασιών και μηχανών, παράγονται 80 δοκιμαστικά αρχεία. Κατηγοριοποιώντας τα δοκιμαστικά σετ σε

διαφορετικές κατηγορίες, προκύπτουν τρεις κατηγορίες δοκιμαστικών δεδομένων, χωρισμένες σύμφωνα με το μέγεθος κάθε προβλήματος. Στην πρώτη κατηγορία περιέχονται προβλήματα όπου το πλήθος των εργασιών δεν ξεπερνάει τις 110 εργασίες, στη δεύτερη κατηγορία τα μεγέθη των προβλημάτων κυμαίνονται μεταξύ 250 και 6,000 εργασιών. Στην τελευταία κατηγορία, οι αλγόριθμοι καλούνται να επιλύσουν προβλήματα της τάξης των 20,000 εργασιών. Ο διαχωρισμός των προβλημάτων σε κατηγορίες μεγέθους έχει ως απώτερο σκοπό την μελέτη της συμπεριφοράς κάθε αλγόριθμου, και κατά πόσο αυτή επηρεάζεται όσο το πρόβλημα προς επίλυση μεγαλώνει. Όντως, κατά την μεταβολή του μεγέθους των προβλημάτων παρατηρείται αλλαγή στη ποιότητα λύσης που προσφέρουν ορισμένοι από τους υλοποιημένους αλγόριθμους.

Αναλυτικότερα, στους Πίνακες 4.1, 4.2 και 4.3 απεικονίζονται τα διαφορετικά μεγέθη προβλημάτων, χωρισμένα στις αντίστοιχες κατηγορίες, που κλήθηκαν να επιλύσουν οι αλγόριθμοι. Κάθε πίνακας αντιστοιχίζεται σε μια κατηγορία προβλημάτων, όπως αναφέρθηκαν παραπάνω, στον οποίο αναγράφεται το πλήθος των εργασιών και διαθέσιμων μηχανών, καθώς και το συνολικό πλήθος ακέραιων μεταβλητών που έχει το εκάστοτε πρόβλημα. Κάθε πρόβλημα είναι ένας πίνακας NxM που αναπαριστά το κόστος ανάθεσης C μιας εργασίας N στο μηχάνημα M.

Όπως φαίνεται στους πίνακες, στην πρώτη και στη δεύτερη κατηγορία περιέχονται 30 διαφορετικά προβλήματα, ενώ για στην τελευταία έχουν επιλυθεί 20 προβλήματα μεγάλου μεγέθους.

4.1.2 Σύγκριση αλγορίθμων

Στο πρόβλημα της ανάθεσης έχει ιδιαίτερη σημασία ένας αλγόριθμος να επιφέρει λύση όσο το δυνατόν πιο κοντά στη βέλτιστη μέσα σε συγκεκριμένα χρονικά πλαίσια. Για την εύρεση της βέλτιστης λύσης σε όλα τα σύνολα δεδομένων χρησιμοποιήθηκε το μοντέλο ελαχιστοποίησης με τη χρήση της βιβλιοθήκης Gurobi, που συζητήθηκε στο Κεφάλαιο 3. Εφαρμόζοντας το γραμμικό μοντέλο με είσοδο το αρχείο που βρίσκεται ο πίνακας κόστους, καθώς και το αντίστοιχο πλήθος μηχανών και εργασιών προς δρομολόγηση, κατέστη δυνατή η εύρεση της βέλτιστης λύσης για όλα τα δοκιμαστικά σύνολα δεδομένων και των τριών κατηγοριών. Κάθε βέλτιστη λύση ορίστηκε ως το σημείο αναφοράς για τα αποτελέσματα που

Cost Matrix Size	Integers	Cost Matrix Size	Integers
10 x 15	150	27 x 30	810
10 x 20	200	20 x 50	1,000
12 x 20	240	30 x 35	1,050
10 x 25	250	35 x 40	1,400
15 x 20	300	30 x 55	1,650
18 x 20	360	30 x 65	1,950
15 x 25	375	30 x 65	1,950
14 x 32	448	40 x 75	3,000
15 x 30	450	50 x 75	3,750
20 x 25	500	50 x 75	3,750
15 x 35	525	55 x 80	4,400
15 x 35	525	60 x 85	5,100
22 x 27	594	60 x 100	6,000
20 x 30	600	85 x 100	8,500
25 x 30	750	85 x 110	9,350

Πίνακας 4.1: Μικρά προβλήματα ανάθεσης

επιφέρουν οι υπόλοιποι αλγόριθμοι.

Όλοι οι αλγόριθμοι υλοποιήθηκαν και εκτελέστηκαν σε περιβάλλον Windows 10 με τη χρήση του προγράμματος Visual Studio Code. Ο υπολογιστής που χρησιμοποιήθηκε για την υπολογιστική μελέτη διαθέτει τον επεξεργαστή Intel i5 11400F, με 6 πυρήνες και συχνότητα επεξεργαστή 2.6 GHz. Η διαθέσιμη μνήμη RAM του είναι 16 GB. Όλοι οι αλγόριθμοι υλοποιήθηκαν σε C++, η επιλογή της γλώσσας έγινε για λόγους αποδοτικότητας, ευελιξίας και ταχύτητας.

Στην πρώτη κατηγορία δοκιμαστικών δεδομένων, όπως αναλύθηκαν προηγουμένως, το κριτήριο με το οποίο θα συγκριθούν οι αλγόριθμοι είναι η λύση που προσφέρουν. Λόγω του μικρού μεγέθους που έχουν τα προβλήματα αυτά, ο χρόνος εκτέλεσης των αλγόριθμων βρίσκεται στην κλίμακα χιλιοστών δευτερολέπτου, όπως απεικονίζεται στο Σχήμα 4.2, οπότε δεν λαμβάνεται υπόψη στη σύγκριση. Στο Σχήμα 4.1 απεικονίζονται όλα τα τελικά κόστη ανάθεσης των τεσσάρων αλγόριθμων και για στις 30 περιπτώσεις. Εκτελώντας το πρώτο σύνολο προβλημάτων, παρατηρείται πως παρόλο που κανένας αλγόριθμος δεν εντοπίζει τη βέλτιστη λύση, ο προτεινόμενος Ουγγρικός αλγόριθμος αποφέρει λύσεις πιο κοντά στη βέλτιστη. Με σχετικά μικρή διαφορά ακολουθεί ο τροποποιημένος Ουγγρικός αλγόριθμος [2], όπου όσο μεγαλώνει το μέγεθος του προβλήματος γίνεται ο αλγόριθμος που φέρει την καλύτερη λύση.

Από την άλλη, ο Ουγγρικός αλγόριθμος που προτείνει ο Kumar δεν παρεκ-

Cost Matrix Size	Integers	Cost Matrix Size	Integers
100 x 250	25,000	950 x 2,000	1,900,000
150 x 500	75,000	1,300 x 1,500	1,950,000
300 x 400	120,000	1,200 x 2,000	2,400,000
350 x 550	192,500	1,000 x 2,500	2,500,000
250 x 850	212,500	1,500 x 2,000	3,000,000
450 x 650	292,500	1,500 x 2,700	4,050,000
500 x 750	292,500	1,000 x 4,500	4,500,000
500 x 900	450,000	2,000 x 4,500	9,000,000
600 x 950	570,000	2,500 x 3,600	9,000,000
750 x 1,000	750,000	2,800 x 3,400	9,520,000
600 x 1,500	900,000	2,000 x 5,000	10,000,000
800 x 1,400	1,120,000	2,500 x 4,000	10,000,000
900 x 1,300	1,170,000	3,000 x 3,500	10,500,000
900 x 1,700	1,530,000	3,000 x 6,000	18,000,000
1,000 x 1,800	1,800,000	4,500 x 5,000	22,500,000

Πίνακας 4.2: Μεσαία προβλήματα ανάθεσης

κλίνει από τη θεωρητική συμπεριφορά που παρατηρήθηκε στην ανάλυση του. Όπως είχε σωστά αναφερθεί, η τμηματοποίηση του προβλήματος επηρεάζει αρνητικά τον εντοπισμό της βέλτιστης λύσης. Η εν λόγω συμπεριφορά του αλγόριθμου επιβεβαιώνει την υπόθεση πως όλες οι αναθέσεις στο πρόβλημα της ανάθεσης εργασιών είναι αλληλοσυνδεόμενες. Αν και υπάρχει αισθητή απόκλιση από την ιδανική λύση, σε ορισμένες περιπτώσεις εφαρμογής του μπορεί να μην θεωρείται ακόμα προβληματική η χρήση του αλγόριθμου.

Σε αντίθεση με τους Ουγγρικούς αλγόριθμους, μελετώντας τον αλγόριθμο Dhouib-Matrix-AP2 παρατηρείται η μεγαλύτερη απόκλιση από την θεωρητική ανάλυση που έγινε στο Κεφάλαιο 2. Η επιλογή του αλγορίθμου έγινε κυρίως για λόγους σύγκρισης των Ουγγρικών αλγορίθμων, αφού στα δοκιμαστικά παραδείγματα που αναλύθηκαν ήταν η μόνη μέθοδος που κατάφερε να βρει τις βέλτιστες λύσεις και στα τέσσερα διαφορετικά προβλήματα. Παρά τη θεωρητική του ανάλυση, εκτελώντας τα δοκιμαστικά σύνολα δεδομένων, όχι μόνο ο αλγόριθμος δεν καταφέρνει να πλησιάσει τη βέλτιστη λύση, αλλά σε μερικά από τα σύνολα δεδομένων παρουσιάζει τη χειρότερη ποιότητα λύσης σε σχέση με τους υπόλοιπους αλγόριθμους.

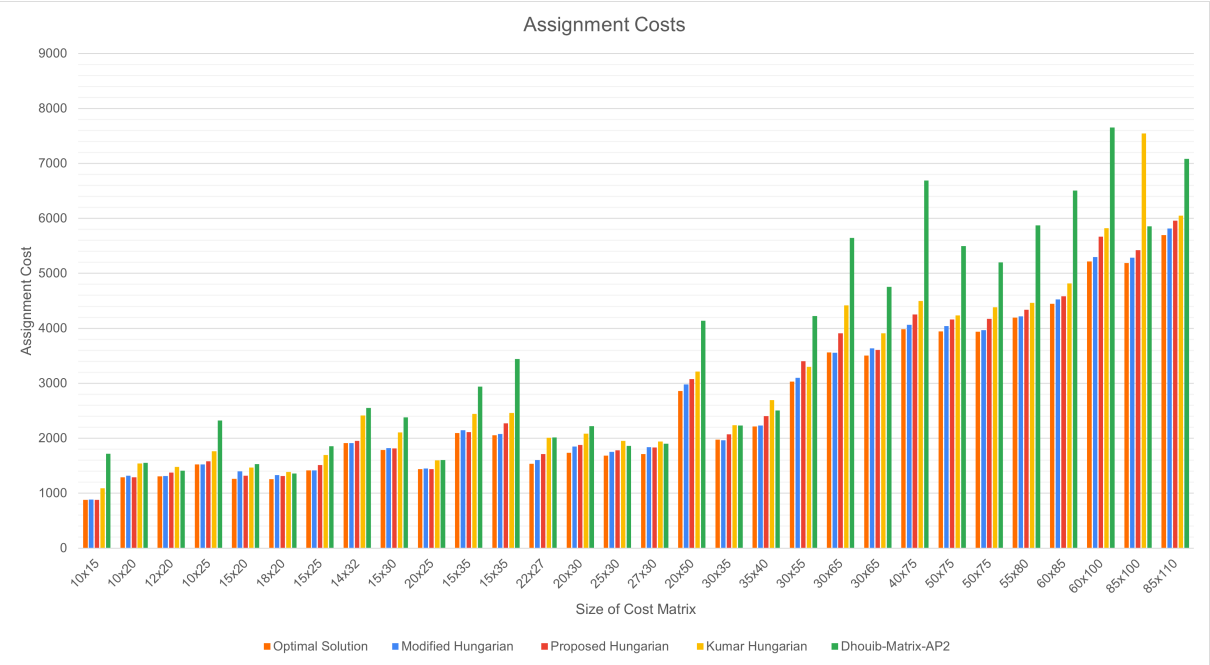
Δίνοντας ως είσοδο στους αλγορίθμους προβλήματα μεσαίου μεγέθους, τα προβλήματα δηλαδή που ανήκουν στην δεύτερη κατηγορία, παρατηρείται μικρή μεταβολή στις συμπεριφορές των αλγορίθμων. Πλέον, από τα Σχήματα 4.3 και 4.4, είναι εμφανές πως την καλύτερη απόδοση έχει ο Τροποποιημένος Αλγόριθμος του

Size of Cost Matrix	Number of integer
1,000 x 25,000	25,000,000 integer
1,000 x 13,000	26,000,000 integer
1,000 x 13,500	27,000,000 integer
1,000 x 14,000	28,000,000 integer
1,000 x 30,000	30,000,000 integer
1,500 x 20,000	30,000,000 integer
1,700 x 18,000	30,600,000 integer
1,700 x 18,500	31,450,000 integer
1,600 x 20,000	32,000,000 integer
1,610 x 20,000	32,200,000 integer
1,610 x 20,010	32,216,100 integer
1,610 x 20,030	32,248,300 integer
1,610 x 20,050	32,280,500 integer
1,610 x 20,100	32,361,000 integer
1,610 x 20,200	32,522,000 integer
1,610 x 20,300	32,683,000 integer
1,610 x 20,400	32,844,000 integer
1,610 x 20,500	33,005,000 integer
1,610 x 20,600	33,166,000 integer
1,610 x 20,700	33,327,000 integer

Πίνακας 4.3: Μεγάλα προβλήματα ανάθεσης

Rabbani, απέχοντας ελάχιστα από τη βέλτιστη λύση που υποδεικνύει το γραμμικό μοντέλο. Στα μεσαίου μεγέθους προβλήματα, αρχίζει σιγά σιγά να λαμβάνεται υπόψη και ο χρόνος εκτέλεσης, αφού σε αυτό το σημείο οι αλγόριθμοι έχουν διάρκεια μερικά δευτερόλεπτα. Ελέγχοντας τα χρονικά αποτελέσματα των εκτελέσεων στα διάφορα δοκιμαστικά σύνολα, παρατηρείται πως η πρόταση του Kumar έχει σταθερά τη μικρότερη διάρκεια εκτέλεσης σε όλα τα προβλήματα. Αυτό είναι λογικό, καθώς ο αλγόριθμος επιλύει μικρότερα προβλήματα από τους υπόλοιπους αλγόριθμους, έχοντας ως μέγιστο μέγεθος για τον πίνακα κόστους το πλήθος των μηχανών του εκάστοτε προβλήματος και όχι αυτό των εργασιών.

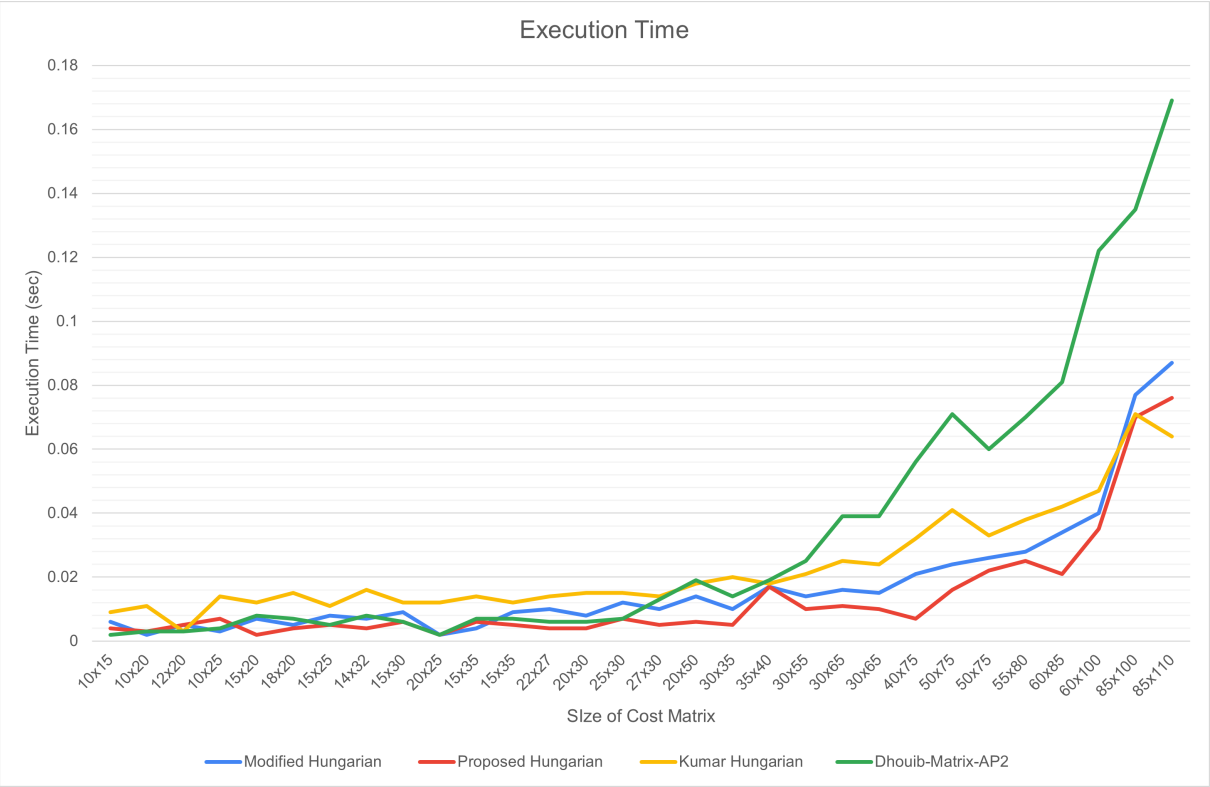
Σε αυτή την κατηγορία αρχίζει να εμφανίζεται η ανάγκη περιορισμού του αλγόριθμου Dhouib-Matrix-AP2. Λόγω της τυχαίας επιλογής που εκτελεί ο αλγόριθμος όταν υπάρχει ισότητα στα αθροίσματα σειρών ή στηλών, δημιουργείται ένα δέντρο αποφάσεων, το οποίο προφανώς επηρεάζει την ποιότητα της τελικής λύσης και τον χρόνο εκτέλεσης του αλγόριθμου. Μια από τις ιδιαιτερότητες που έχει ο αλγόριθμος είναι η επαναληπτικότητα που διαθέτει με σκοπό να καταλήξει στη καλύτερη δυνατή λύση. Σύμφωνα με τον Αλγόριθμο 5, ο DM-AP2 εκτελείται τόσες



Σχήμα 4.1: Κόστη ανάθεσης αλγορίθμων: Πρώτη κατηγορία προβλημάτων

φορές ώσπου να έχει N επαναλήψεις χωρίς βελτίωση της λύσης, όπου το N να είναι το πλήθος των εργασιών προς ανάθεση. Στα μεσαία και μεγάλα προβλήματα η ύπαρξη αυτού του περιορισμού καθίσταται μη ρεαλιστική, καθώς η χρονική πολυπλοκότητα του αλγορίθμου αυξάνεται ανάλογα με το πλήθος των εργασιών.

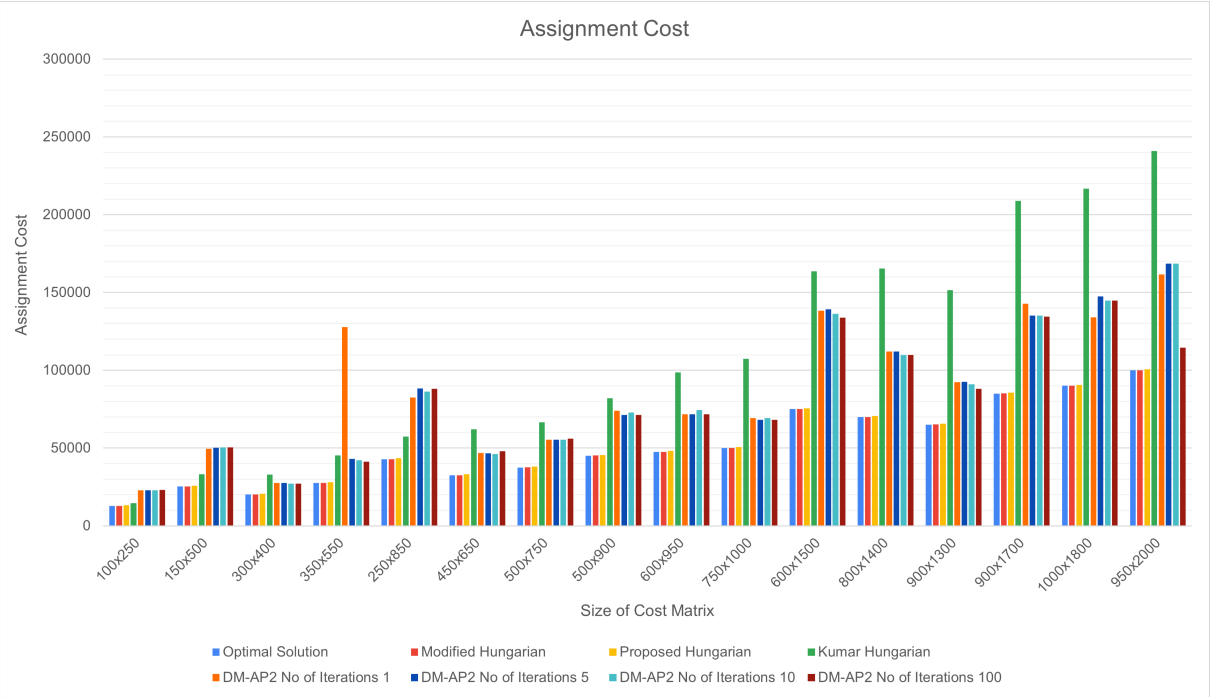
Με σκοπό τη βελτίωση του χρόνου εκτέλεσης του αλγορίθμου, επιβάλλεται ένα μέγιστο όριο επαναλήψεων που μπορεί να εκτελεστεί. Το όριο αυτό για τα προβλήματα μεσαίου μεγέθους κυμαίνεται κοντά στις δέκα επαναλήψεις. Αυτή η τιμή για την μεταβλητή επιλέχθηκε με βάση τον χρόνο εκτέλεσης του αλγορίθμου, ώστε η χρονική απόκλιση σε σύγκριση με τους υπόλοιπους αλγόριθμους να μην είναι ιδιαίτερα μεγάλη. Παρόλα αυτά, για εκτενή ανάλυση του αλγορίθμου εφαρμόστηκαν διάφορα πλήθη επαναλήψεων, τα οποία έπειτα συγκρίθηκαν μεταξύ τους. Η παράμετρος k για το πλήθος των επαναλήψεων επιλέχθηκε να λάβει τις τιμές ένα, πέντε, δέκα και εκατό. Στο Σχήμα 4.5, στα περισσότερα προβλήματα οι πρώτες τρεις επιλογές παραμέτρου αποφέρουν μικρής απόκλισης αποτελέσματα, ωστόσο λόγω της τυχαιότητας του αλγορίθμου τα αποτελέσματα διαφέρουν από πρόβλημα σε πρόβλημα. Παρατηρώντας τη χρονική συμπεριφορά του αλγόριθμου με το Σχήμα 4.6, εμφανίζονται ξαφνικές μεταβολές μεταξύ των χρόνων εκτέλεσης σε κάθε πρόβλημα. Όσο πιο μικρή τιμή έχει η παράμετρος k, τόσο λιγότερες αυξομειώσεις θα προκύψουν κατά την εκτέλεση του DM-AP2.



Σχήμα 4.2: Χρόνοι εκτέλεσης αλγορίθμων: Πρώτη κατηγορία προβλημάτων

Μέχρι στιγμής το μέγεθος των δεδομένων που καλούνται να επιλύσουν οι αλγόριθμοι δεν είναι ιδιαίτερα μεγάλο, με το πλήθος των ακέραιων που έχει ένα πρόβλημα να μην ξεπερνάει της δέκα εκατομμύρια μεταβλητές. Για την μεσαία κατηγορία συνόλου δεδομένων, το μοντέλο του Gurobi καταφέρνει να επιλύσει τα δοκιμαστικά σύνολα σε σχετικά μικρό χρόνο διάστημα, απαιτώντας όμως πολλούς υπολογιστικούς πόρους. Εξετάζοντας χρονικά τα αποτελέσματα όλων των αλγορίθμων, παρατηρείται πως αν και ο τροποποιημένος Ουγγρικός αλγόριθμος έχει καλύτερη ποιότητα λύσης, οι χρόνοι εκτέλεσης του, όπως απεικονίζονται με τα Σχήματα 4.7 και 4.8, είναι με διαφορά οι χειρότεροι. Συνεπώς, λαμβάνοντας υπόψη το χρόνο εκτέλεσης και την ποιότητα λύσης υφίσταται λογική η προτίμηση του λύτη Gurobi για προβλήματα που ανήκουν στην πρώτη και δεύτερη κατηγορία συνόλων. Εφόσον υπάρχει πρόσβαση σε υπολογιστικούς πόρους, η επιλογή της χρήσης του λύτη επιφέρει τη βέλτιστη λύση με σιγουριά σε μικρά χρονικά πλαίσια. Παρόλα αυτά, στην περίπτωση που οι διαθέσιμοι υπολογιστικοί πόροι είναι περιορισμένοι, ο προτεινόμενος Ουγγρικός αλγόριθμος ισορροπεί την καλή ποιότητα λύσης με τον ελάχιστο δυνατό χρόνο επίλυσης του προβλήματος.

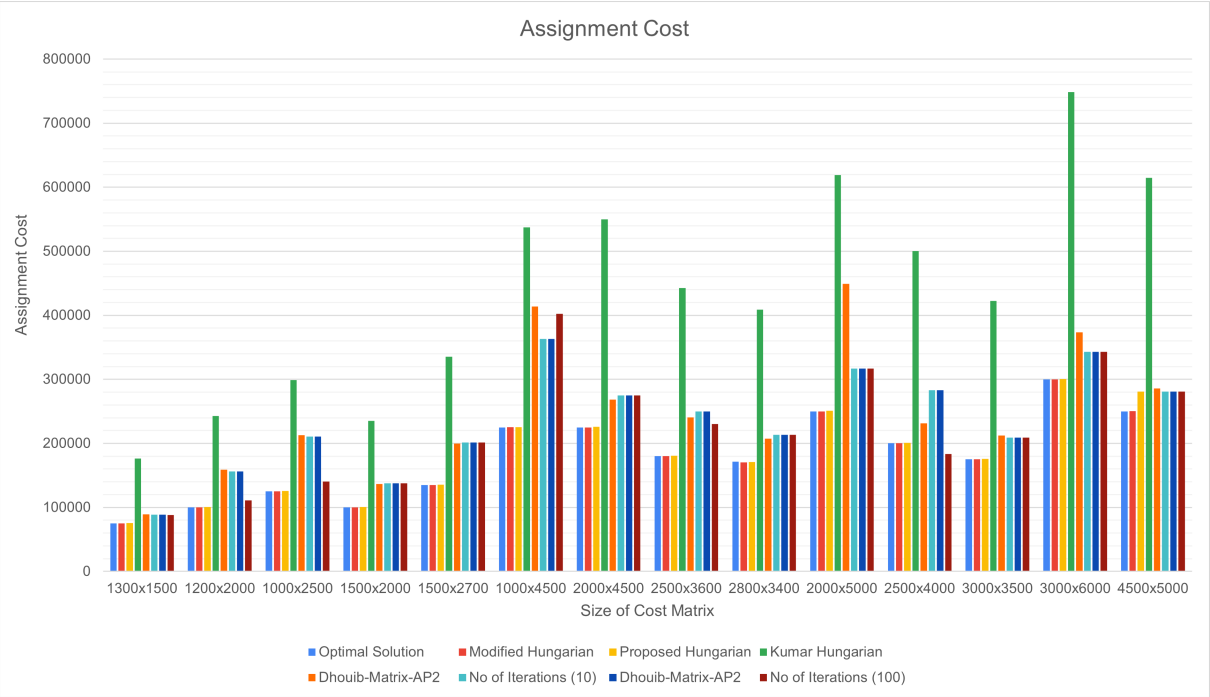
Στη τρίτη κατηγορία δοκιμαστικών δεδομένων, όπως φαίνεται στον Πίνακα 4.3,



Σχήμα 4.3: Κόστη ανάθεσης αλγορίθμων: Δεύτερη κατηγορία προβλημάτων 1

ανήκουν προβλήματα τα οποία έχουν κοντά στις τριάντα εκατομμύρια μεταβλητές. Παρατηρώντας τα αποτελέσματα στο Σχήμα 4.9 είναι πλέον ξεκάθαρο πως τα ευρήματα που προέκυψαν για τα δοκιμαστικά δεδομένα μεσαίου μεγέθους συνεχίζουν να ισχύουν. Καθιστά επομένως λογικό ότι οι συμπεριφορές και των τεσσάρων αλγορίθμων θα παραμείνουν ίδιες όσο και αν μεγαλώσει το πρόβλημα. Καθώς αυξάνεται το μέγεθος των προβλημάτων που καλούνται να λύσουν οι αλγόριθμοι, το χάσμα μεταξύ των λύσεων που επιφέρουν θα συνεχίσει και αυτό να μεγαλώνει σταδιακά. Όπως σημειώθηκε σωστά στο Κεφάλαιο 2, ο Ουγγρικός αλγόριθμος του Kumar εξακολουθεί να έχει τη χειρότερη ποιότητα λύσεων σε όλα τα δοκιμαστικά δεδομένα της κατηγορίας, ενώ ο αλγόριθμος DM-AP2 ακολουθεί, με ασταθή αποτελέσματα.

Στο Σχήμα 4.11 απεικονίζεται το τελικό κόστος επίλυσης κάθε προβλήματος που ανήκει στην τρίτη κατηγορία δεδομένων, από τους δύο Ουγγρικούς αλγόριθμους, τον τροποποιημένο και τον προτεινόμενο. Χρησιμοποιώντας τον λύτη Gurobi, τα προαναφερόμενα αποτελέσματα συγκρίνονται με τη βέλτιστη λύση. Παρατηρείται πως, όπως και στα μεσαία προβλήματα, ο Τροποποιημένος έχει καλύτερη απόδοση, ωστόσο η απόκλιση μεταξύ των δύο Ουγγρικών αλγορίθμων δεν είναι μεγάλη. Για τη σωστή σύγκριση των αλγορίθμων καθίσταται πλέον απαραί-

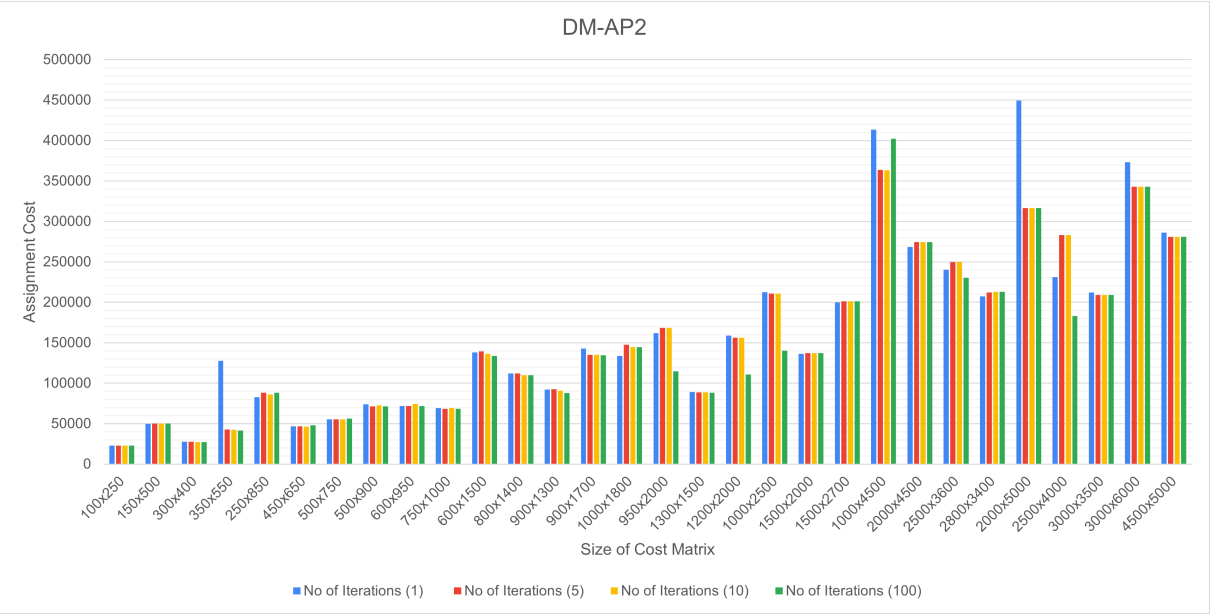


Σχήμα 4.4: Κόστη ανάθεσης αλγορίθμων: Δεύτερη κατηγορία προβλημάτων 2

τητο να ληφθούν υπόψη οι εκάστοτε χρόνοι εκτέλεσης. Για τα αντίστοιχα προβλήματα, οι χρόνοι εκτέλεσης κάθε αλγορίθμου παρουσιάζονται στο Σχήμα 4.10, οι οποίοι όπως φαίνεται ξεπερνούν τα δεκαπέντε λεπτά.

Συγκρίνοντας τους αλγορίθμους με βάση τον χρόνο, δεν αποτελεί έκπληξη πως ο ταχύτερος αλγόριθμος είναι του Kumar, ο οποίος είναι σχεδόν τέσσερις φορές πιο γρήγορος από τον προτεινόμενο Ουγγρικό και τον DM-AP2 αλγόριθμο. Για να υπάρχει ισορροπία στις συγκρίσεις μεταξύ των αλγορίθμων, ορίζεται ως παράμετρος k επαναλήψεων του αλγορίθμου DM-AP2 η τιμή 10, πλησιάζοντας τα χρονικά πλαίσια του Τροποποιημένου Ουγγρικού αλγόριθμου. Αναφορικά με τους δύο Ουγγρικούς αλγόριθμους, από το Σχήμα 4.12 είναι προφανές πως ο προτεινόμενος αλγόριθμος επιλύει τα προβλήματα με μεγαλύτερη ταχύτητα από τον Τροποποιημένο. Όπως φαίνεται αναλυτικά στον Πίνακα 4.4 ο προτεινόμενος αλγόριθμος είναι κατά μέσο όρο σχεδόν 29.5% ταχύτερος σε σύγκριση με τον Τροποποιημένο Ουγγρικό σε όλα τα δοκιμαστικά σύνολα δεδομένων.

Από όλους τους αλγόριθμους βελτιστοποίησης που υλοποιήθηκαν και εκτελέστηκαν, οι πιο αποδοτικοί σε γενικές γραμμές είναι ο προτεινόμενος Ουγγρικός αλγόριθμος και ο Τροποποιημένος Ουγγρικός αλγόριθμος του Rabbani. Από τη μια, ο Τροποποιημένος Ουγγρικός αλγόριθμος προσφέρει καλύτερη ποιότητα λύ-

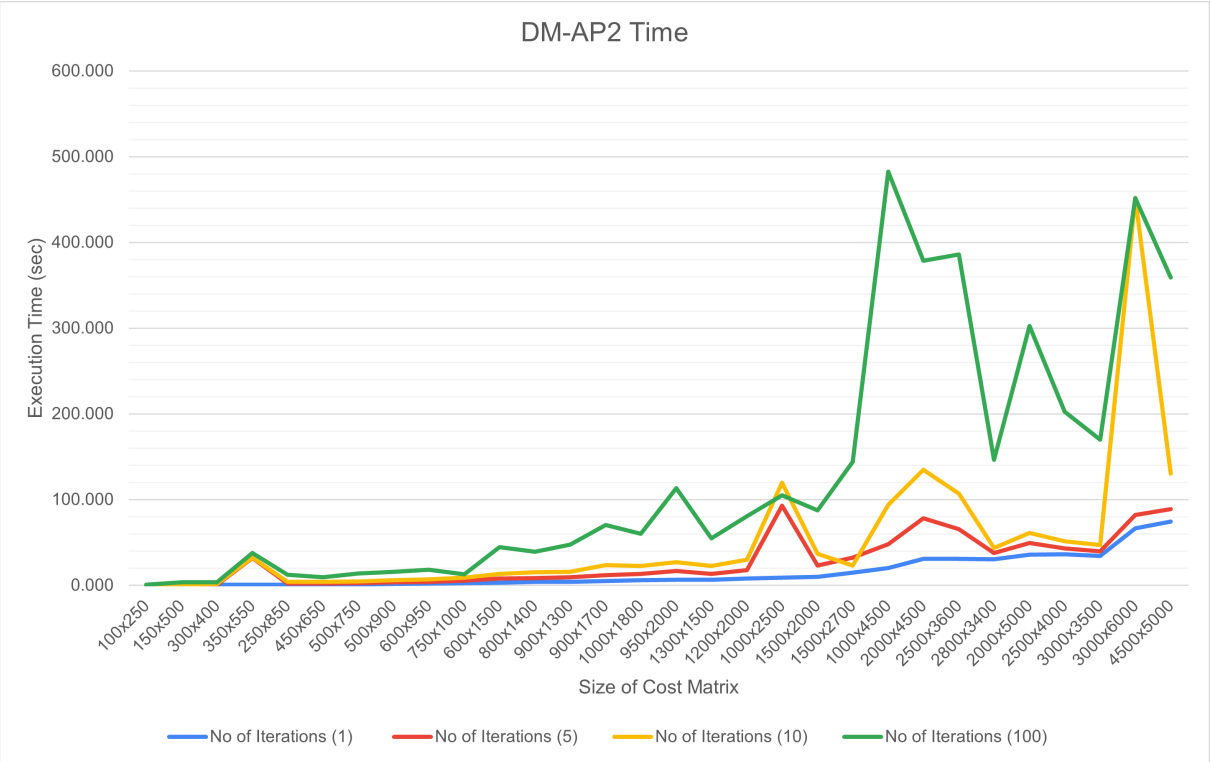


Σχήμα 4.5: Κόστη ανάθεσης DM-AP2: Δεύτερη κατηγορία προβλημάτων

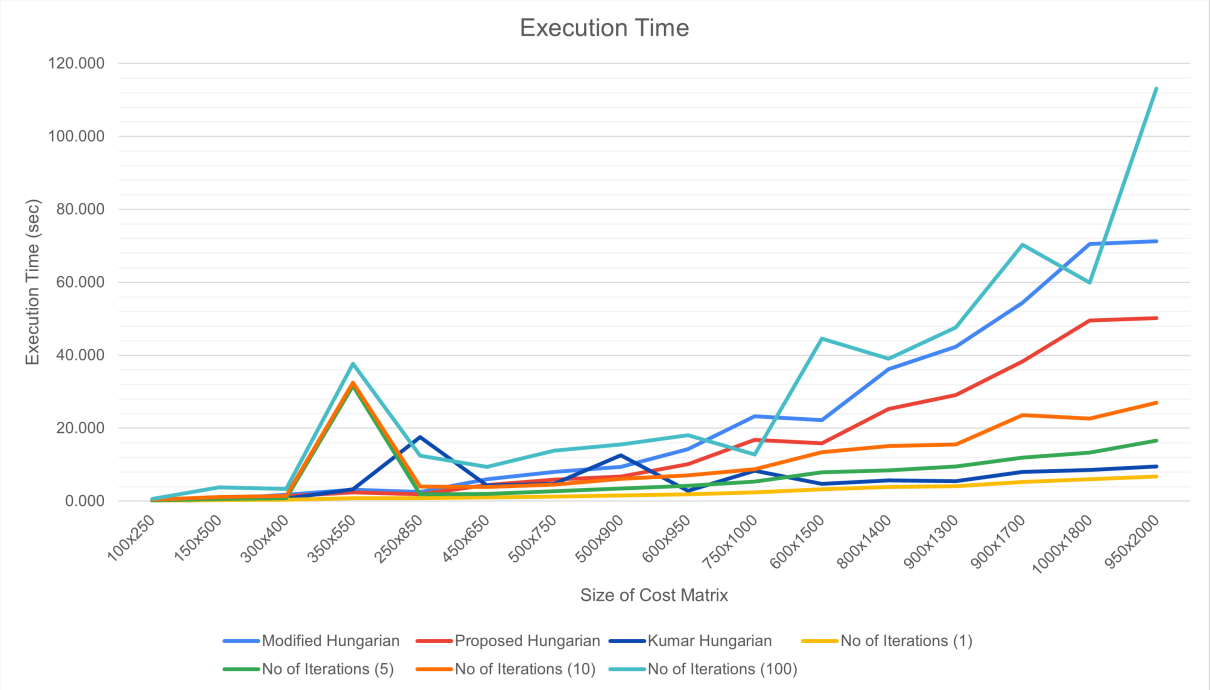
Size of Problem	Hungarian Speed rate
1,000 x 25,000	28.207%
2,000 x 13,000	30.586%
2,000 x 13,500	30.603%
2,000 x 14,000	30.781%
1,000 x 30,000	31.121%
1,500 x 20,000	34.541%
1,700 x 18,000	28.975%
1,700 x 18,500	28.914%
1,600 x 20,000	26.503%
1,610 x 20,000	28.647%
1,610 x 20,010	29.269%
1,610 x 20,030	29.687%
1,610 x 20,050	29.578%
1,610 x 20,100	28.219%
1,610 x 20,200	28.859%
1,610 x 20,300	28.947%
1,610 x 20,400	28.953%
1,610 x 20,500	28.939%
1,610 x 20,600	29.167%
1,610 x 20,700	28.381%

Πίνακας 4.4: Ποσοστά επίταχυνσης Ουγγρικού Αλγόριθμου

σεων, απέχοντας ελάχιστα από τη βέλτιστη λύση ακόμα και σε μεγάλα προβλήματα. Ωστόσο, ο χρόνος εκτέλεσης του αυξάνεται δραματικά όσο μεγαλώνει και το πρόβλημα που δέχεται ως είσοδο. Από την άλλη, ο προτεινόμενος Ουγγρικός αλγόριθμος, αν και τα αποτελέσματα που έχει είναι λιγότερο επιθυμητά, όπως

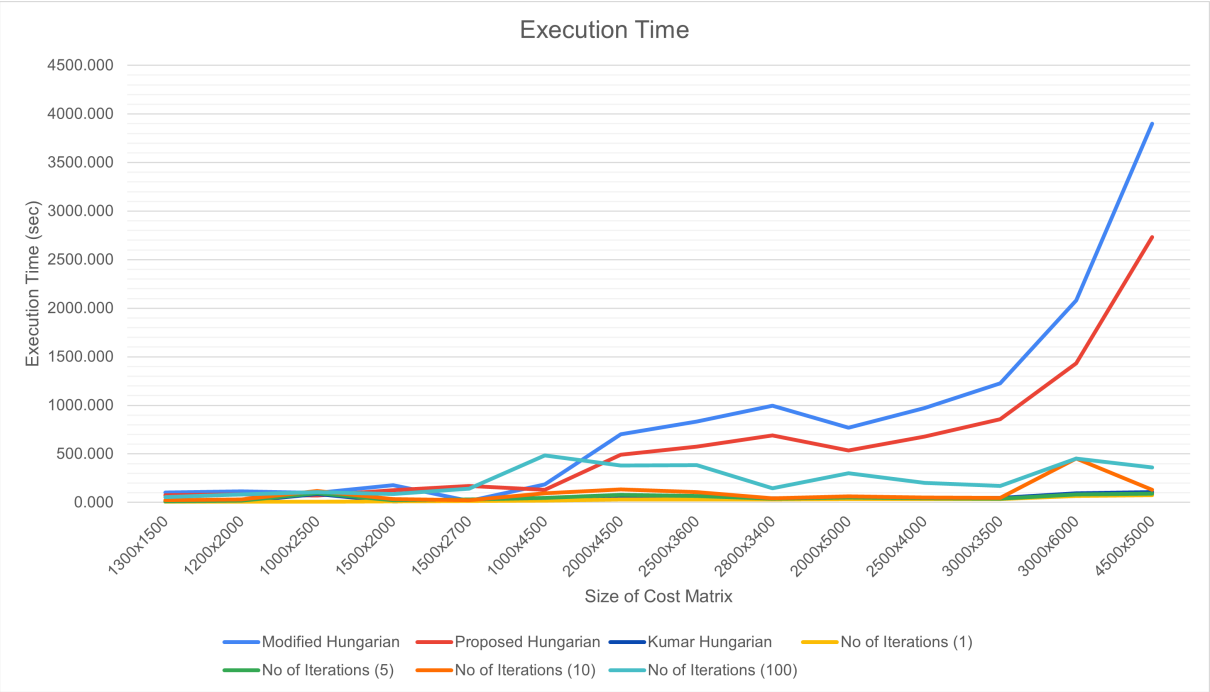


Σχήμα 4.6: Χρόνοι εκτέλεσης DM-AP2: Δεύτερη κατηγορία προβλημάτων



Σχήμα 4.7: Χρόνοι εκτέλεσης αλγορίθμων: Δεύτερη κατηγορία προβλημάτων 1

δείχνει το Σχήμα 4.11, έχει περίπου 0.067% απόκλιση από τη βέλτιστη λύση. Επιπλέον, ο χρόνος εκτέλεσης του αλγορίθμου είναι σημαντικά μικρότερος από τους υπόλοιπους, καθιστώντας τον ως την ιδανική λύση στο πρόβλημα της ανάθεσης εργασιών.

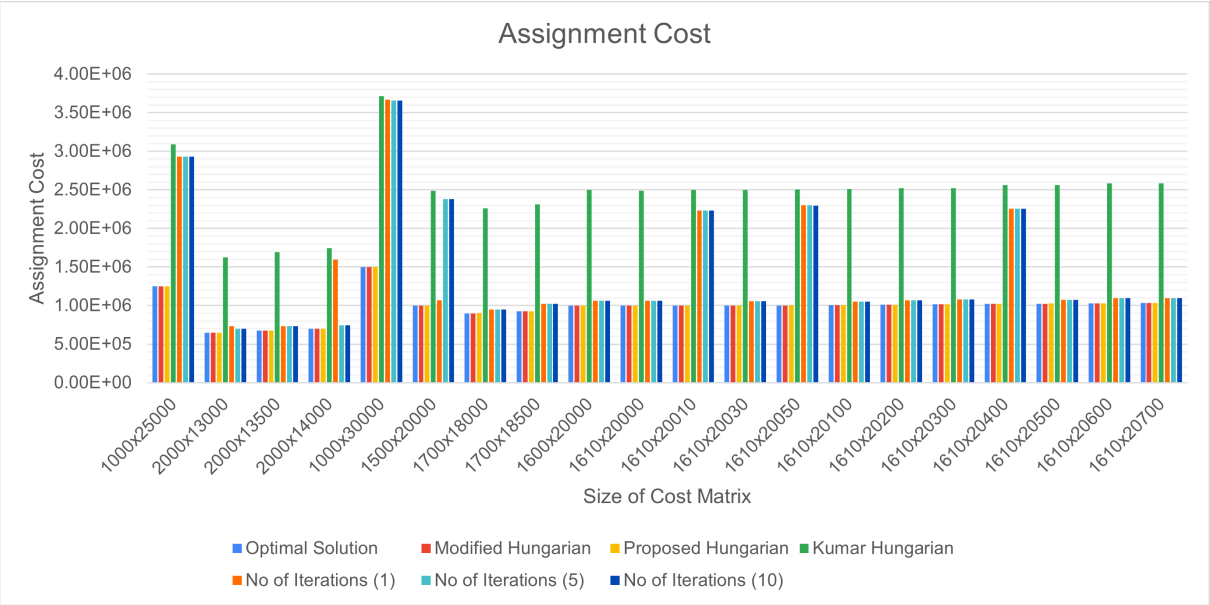


Σχήμα 4.8: Χρόνοι εκτέλεσης αλγορίθμων: Δεύτερη κατηγορία προβλημάτων 2

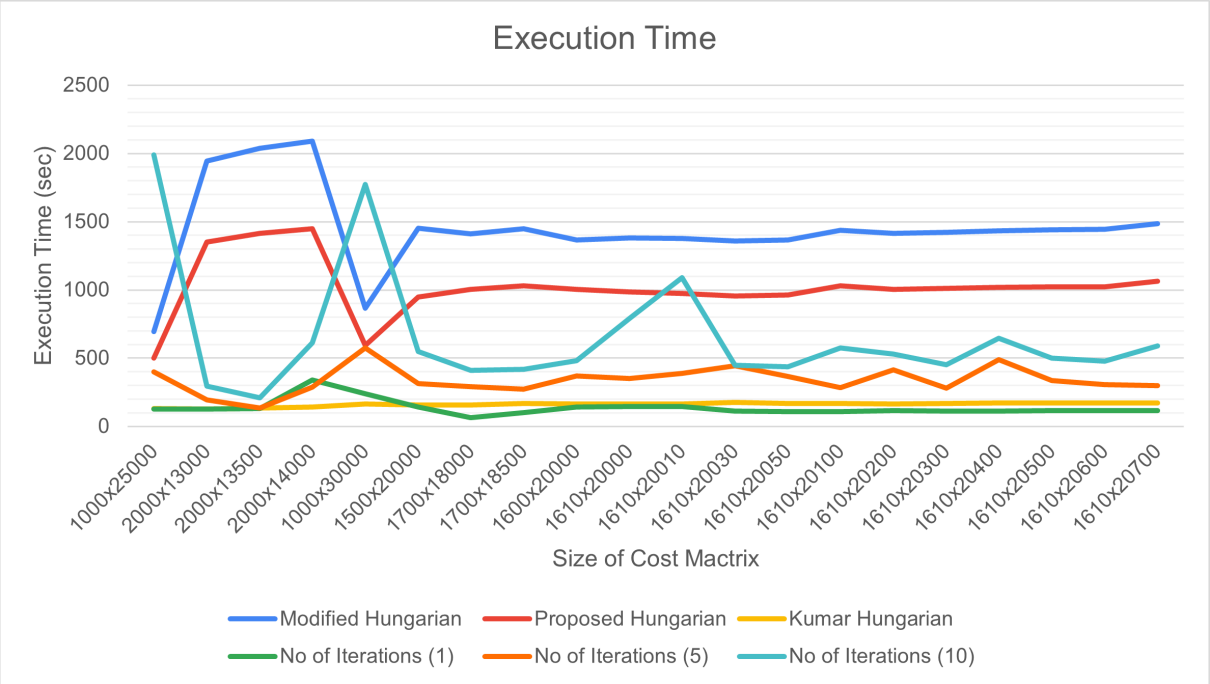
Για προβλήματα μεγαλύτερου μεγέθους, δεν υφίσταται η χρήση του λύτη Gurobi, καθώς προβλήματα με 20,000 εργασίες και 1,500 VM παράγουν ένα μοντέλο με περισσότερες από 50 εκατομμύρια μη μηδενικές μεταβλητές. Για να καταφέρει να επιλύσει το μοντέλο σωστά το πρόβλημα, χρειάζεται ο χρήστης να έχει πρόσβαση σε πολυάριθμους υπολογιστικούς πόρους. Δυστυχώς, στην προκειμένη περίπτωση μεγαλύτερα προβλήματα δεν ήταν εφικτό να επιλυθούν, καθώς η διαθέσιμη μνήμη RAM δεν ήταν επαρκής για τον λύτη.

4.2 Cloud Computing

Η υπολογιστική νέφους (Cloud Computing) είναι ένα μοντέλο το οποίο παρέχει πρόσβαση μέσω του διαδικτύου σε υπολογιστικούς πόρους, όπως υπηρεσίες, διακομιστές, αποθήκευση κλπ. Το μοντέλο του νέφους έχει στόχο να εξυπηρετεί τους χρήστες εύκολα, με ευελιξία και με ελάχιστη αλληλεπίδραση με τον πάροχο υπηρεσιών. Όλοι οι υπολογιστικοί πόροι παρέχονται από κεντρικά συστήματα απομακρυσμένα από το χρήστη [11]. Την τελευταία δεκαετία, η τεχνολογία του Cloud χρησιμοποιείται κατά κόρον λόγω των πλεονεκτημάτων και της ευκολίας που προσφέρει στον χρήστη. Η εύκολη πρόσβαση, η αξιοπιστία των υποδομών, η ευελιξία στη χρήση και ανάπτυξη νέων υπηρεσιών, καθώς επίσης και η εξοι-



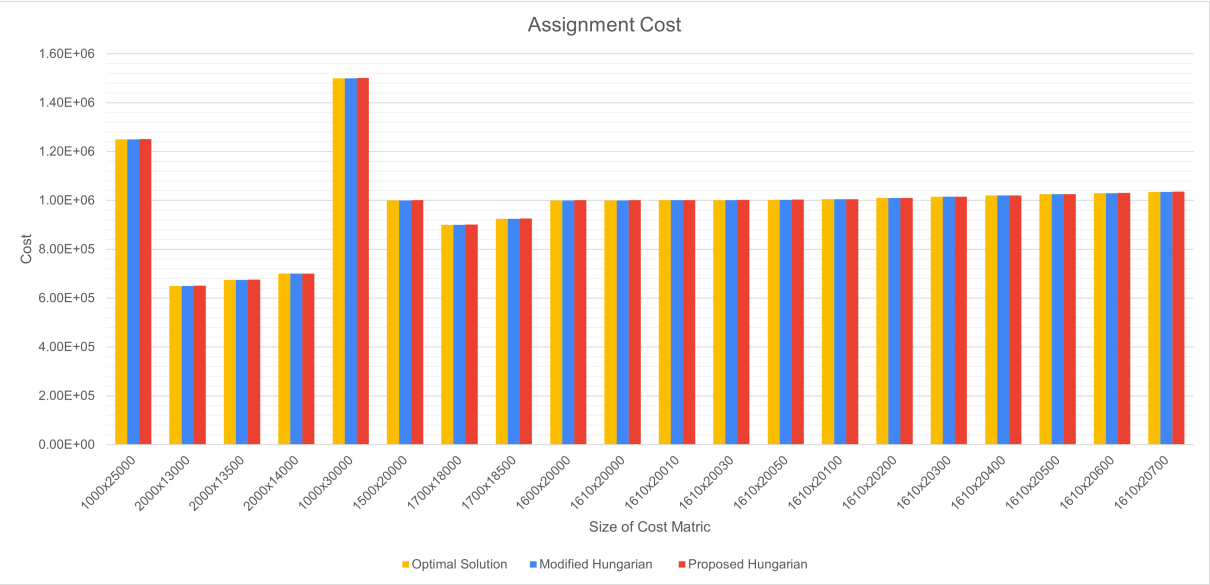
Σχήμα 4.9: Κόστη ανάθεσης αλγορίθμων: Τρίτη κατηγορία προβλημάτων



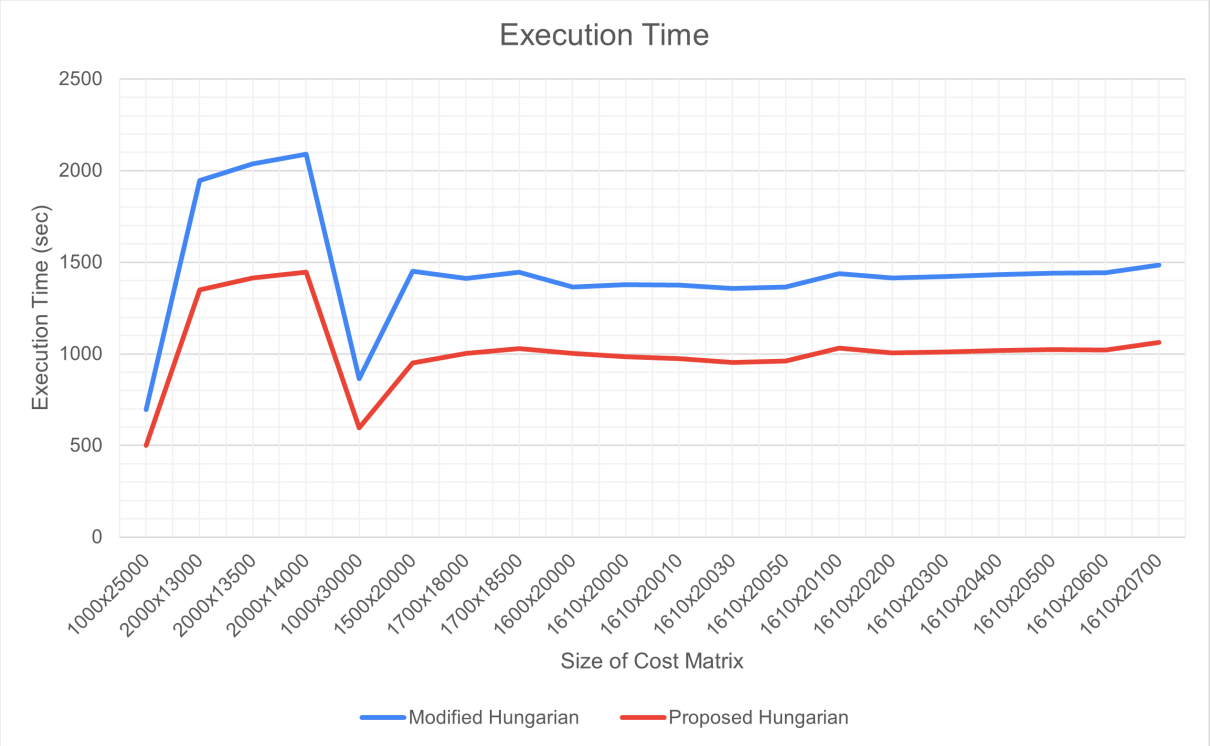
Σχήμα 4.10: Χρόνοι εκτέλεσης αλγορίθμων: Τρίτη κατηγορία προβλημάτων

κονόμηση κόστους, είναι τα κύρια χαρακτηριστικά που συνέβαλαν σε αυτή την εξάπλωση. Η τεχνολογία του υπολογιστικού νέφους χρησιμοποιείται συχνά για την παροχή υπηρεσιών όπως η κατά παραγγελία χρήση εικονικών μηχανών, το ηλεκτρονικό ταχυδρομείο και τα κοινωνικά δίκτυα.

Η αρχιτεκτονική του Cloud βασίζεται στην έννοια των υπηρεσιών και συγκεκριμένα σε τρία βασικά μοντέλα υπηρεσιών [12]. Η πιο βασική κατηγορία είναι η



Σχήμα 4.11: Κόστη ανάθεσης Ουγγρικών αλγορίθμων



Σχήμα 4.12: Χρόνοι εκτέλεσης Ουγγρικών αλγορίθμων

Υποδομή ως Υπηρεσία (IaaS), η οποία παρέχει τη σύνδεση μεταξύ των φυσικών εγκαταστάσεων και των διαδικτυακών υπηρεσιών. Η IaaS έχει την ευθύνη για την παροχή υπολογιστικών πόρων, εικονικών μηχανών VM και χώρο αποθήκευσης σύμφωνα με τον φυσικό εξοπλισμό των εγκαταστάσεων. Σε αυτό το επίπεδο ο χρήστης έχει τη δυνατότητα προσαρμογής των πόρων του συστήματος σύμφωνα με τις ανάγκες του, ωστόσο είναι υπεύθυνος για τη διαχείριση του λειτουργικού

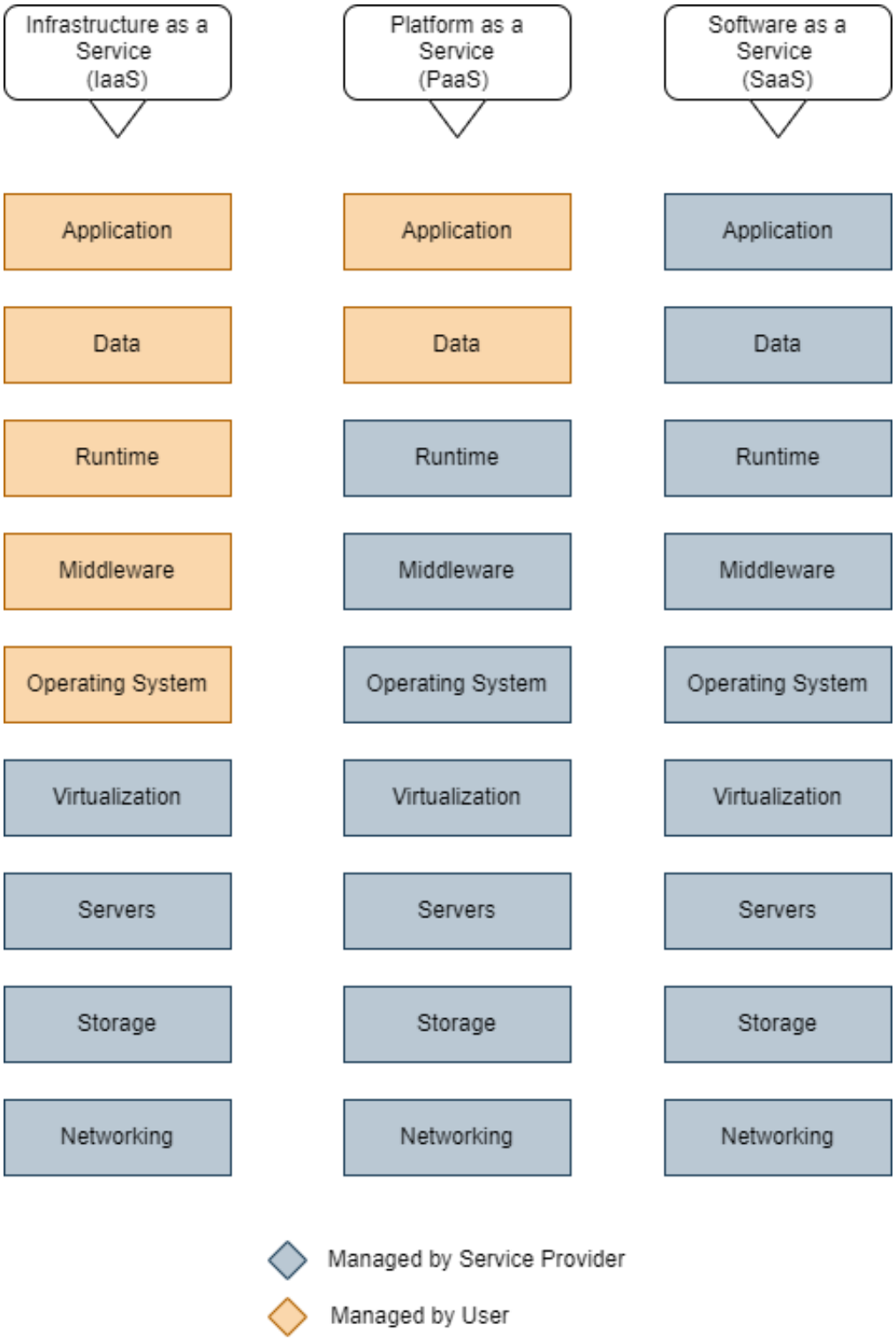
συστήματος, τις εικονικές μηχανές και το middleware [13]. Η επόμενη υπηρεσία είναι η Πλατφόρμα ως Υπηρεσία (PaaS) που αναπτύσσεται ένα επίπεδο πάνω από την IaaS. Η υπηρεσία είναι υπεύθυνη για τη διαχείριση του λειτουργικού συστήματος και του middleware, παρέχοντας στο χρήστη τα εργαλεία για την ελεύθερη ανάπτυξη, διαχείριση και εκτέλεση εφαρμογών. Στο τέλος υπάρχει το Λογισμικό ως Υπηρεσία (SaaS) με το οποίο παρέχεται στους χρήστες διαδικτυακή πρόσβαση σε λογισμικά μέσω του Cloud. Πλέον ο χρήστης δεν είναι υποχρεωμένος να εγκαταστήσει τοπικά εφαρμογές, διότι όλα τα εργαλεία διαχειρίζονται και ενημερώνονται από το SaaS.

Με τη ραγδαία ανάπτυξη του Cloud έχει δημιουργηθεί μεγάλη ζήτηση παροχής υπηρεσιών σε παγκόσμιο επίπεδο χρηστών. Οι απαιτήσεις σε υπολογιστικούς πόρους αυξάνεται ολοένα και περισσότερο, με κάθε εργασία να χρειάζεται διαφορετικό τρόπο διαχείρισης. Για την κάλυψη όλων των αναγκών που προκύπτουν διαρκώς, είναι απαραίτητη η χρήση αλγορίθμων ανάθεσης για βελτιστοποίηση και μείωση του κόστους δρομολόγησης εργασιών σε VM.

4.3 CloudSim

Το CloudSim είναι μια βιβλιοθήκη για προσομοίωση υπολογιστικού νέφους, σχεδιασμένο σε Java, που αναπτύχθηκε από ομάδα ερευνητών του πανεπιστημίου της Μελβούρνης [14]. Πλέον, η χρήση της εργαλειοθήκης CloudSim έχει γίνει εκ των πραγμάτων η πρώτη επιλογή ανάμεσα σε όλες τις διαθέσιμες πλατφόρμες προσομοίωσης νέφους [15]. Με τη βοήθεια του CloudSim είναι εφικτή η ομαλή μοντελοποίηση, προσομοίωση, καθώς και ο πειραματισμός εγκατάστασης και διαχείρισης εφαρμογών και υπηρεσιών σε συστήματα βασισμένα σε υπολογιστικό νέφος. Το CloudSim χρησιμοποιείται κυρίως για ερευνητικούς και ακαδημαϊκούς σκοπούς καθώς μπορεί μόνο να προσομοιώσει τις λειτουργίες ενός νέφους, αδυνατώντας να προσφέρει πραγματικές υποδομές και υπηρεσίες νέφους, όπως το Microsoft Azure ή το AWS της Amazon. Ο χρήστης μπορεί να ρυθμίσει εύκολα και απλά το περιβάλλον και το μοντέλο που επιθυμεί να προσομοιώσει, χωρίς να εμπλακεί περαιτέρω σε λεπτομέρειες που αφορά όλες τις προαναφερόμενες υποδομές [16].

Με το CloudSim ένα υπολογιστικό νέφος αναπαρίσταται με στοιχεία που έχουν τη μορφή κλάσεων, ενώ τα χαρακτηριστικά τους περιγράφονται με τη μορφή συ-



Σχήμα 4.13: Απεικόνιση υπηρεσιών Cloud

ναρτήσεων. Η βασική δομή του στηρίζεται στα Cloudlets, στα VM και στους Hosts, ενώ όλες οι λειτουργίες στο CloudSim πραγματοποιούνται ανάμεσα σε αυτά τα στοιχεία. Οι φυσικές υποδομές ενός νέφους μοντελοποιούνται από το κέντρο δεδομένων (Data center), και από εκεί θα κατανεμηθούν σε Hosts και στα ανάλογα

VM που διαθέτουν. Κάθε Host απεικονίζει φυσικούς πόρους, όπως ένα Server ή έναν υπολογιστή, και είναι υπεύθυνος για την πολιτική κατανομής μνήμης, εύρος ζώνης και γενικά πόρων στα VM που διαθέτει.

Η κλάση Cloudlet αντιπροσωπεύει τις εργασίες του νέφους και διαχειρίζεται τη δρομολόγηση και τη διαδοχική εκτέλεση τους. Στο CloudSim, ένα Cloudlet περιγράφει την πολυπλοκότητα μιας εργασίας όσον αφορά τις απαιτήσεις που έχει σε υπολογιστή ισχύ. Υπεύθυνος για την δρομολόγηση των Cloudlets είναι ο Broker, ο οποίος δέχεται ως είσοδο το σύνολο των Cloudlet προς δρομολόγηση και των VM. Ένας Broker μπορεί να χρησιμοποιήσει διάφορες πολιτικές δρομολόγηση, με την προεπιλεγμένη πολιτική να βασίζεται στον FCFS (First Come First Served - FCFS). Στην περίπτωση που θέλει ένας χρήστης να εφαρμόσει συγκεκριμένη πολιτική δρομολόγησης, έχει τη δυνατότητα να το πραγματοποιήσει αυτό επιτείνοντας την κλάση DatacenterBroker.

Γενικά, η διαδικασία χρονοπρογραμματισμού και ανάθεσης σε περιβάλλον Cloud μπορεί να αφορά είτε στην ανάθεση VMs σε Host, όπου ανάλογα με τις απαιτήσεις του VM μπορεί να υπάρχει αντιστοίχιση ”ένα προς ένα” ή ”πολλά προς ένα”, είτε στην ανάθεση Cloudlets σε VMs. Αυτή η ενότητα επικεντρώνεται στη δεύτερη περίπτωση προγραμματισμού, στον τρόπο δρομολόγησης, δηλαδή, των Cloudlets σε VMs. Το πρόβλημα της ανάθεσης σχετίζεται μόνο στο που θα γίνει η τοποθέτηση κάθε Cloudlets και όχι στην εύρεση συγκεκριμένης σειράς εκτέλεσης στο εσωτερικό του κάθε VM.

Για να γίνει η εφαρμογή των Ουγγρικών αλγόριθμων στο CloudSim, θα δημιουργηθεί μια νέα πολιτική βασισμένη σε αυτούς μέσω του DatacenterBroker που αναφέρθηκε παραπάνω. Αξίζει να σημειωθεί, πως ο Ουγγρικός αλγόριθμος έχει προσομοιωθεί με τη χρήση του CloudSim στο παρελθόν, ωστόσο επίλυε ισορροπημένα προβλήματα και γινόταν χρήση dummy VMs με μηδενικό χώρο. Παρόλα αυτά σύμφωνα με [17] τα αποτελέσματα των προσομοιώσεων έδειχναν μείωση χρόνου εκτέλεσης κατά 13% σε σύγκριση με τον FCFS, ενώ ο χρόνος τερματισμού μειώθηκε και αυτός κατά περίπου 41%.

4.3.1 Σύνολα δεδομένων

Για την αξιολόγηση αλγορίθμων ανάθεσης και χρονοπρογραμματισμού σε νέφος, καθίσταται ολοένα και σημαντικότερο η επιλογή του κατάλληλου συνόλου δεδομένων. Στον τομέα που ασχολείται με την αντιμετώπιση του προβλήματος της ανάθεσης εργασιών και γενικά προβλημάτων βελτιστοποίησης, η ποιότητα των συνόλων δεδομένων έχει άμεσο αντίκτυπο στις μετρικές απόδοσης των αλγορίθμων. Σύνολα δεδομένων τα οποία παρουσιάζουν ποικιλομορφία μεταξύ τους είναι απαραίτητα για την διεξοδική αξιολόγηση της αποτελεσματικότητας ενός αλγορίθμου ανάθεσης σε διάφορα πλαίσια. Γενικά, τα δεδομένα ενδέχεται να εκπροσωπούν ένα ευρύ φάσμα εργασιών, χρόνου εκτέλεσης ή και φόρτου εργασίας επιτρέποντας τη λεπτομερή αξιολόγηση της συμπεριφοράς ενός αλγόριθμου σε πολλαπλά σενάρια. Κατά την επιλογή ενός συνόλου δεδομένων, ο βασικός στόχος είναι τα δεδομένα να είναι όσο το δυνατόν πιο αντιπροσωπευτικά γίνεται. Μόνο με αυτόν τον τρόπο μπορεί να επικυρωθεί ένας αλγόριθμος αξιόπιστα. Οι μεταβολές του συνόλου δεδομένων, έστω και μικρές, αντικατοπτρίζονται στις επιδόσεις των αλγορίθμων κατανομής.

Δυστυχώς όμως, δεν διατίθενται εύκολα δεδομένα μεγάλης κλίμακας, με αποτέλεσμα η απόπειρα αναπαράστασης ενός υπολογιστικού νέφους όσο πιο κοντά στην πραγματικότητα είναι αδύνατη. Σε μια προσπάθεια προσομοίωσης δυναμικού φορτίου εργασιών για το νέφος, χρησιμοποιήθηκε η μέθοδος προσομοίωσης Monte Carlo. Με αυτή τη μέθοδο δίνεται η δυνατότητα παρασκευής εργασιών Google Cloud Jobs (GoCJ), τα οποία βασίζονται στις αναλύσεις που έχουν πραγματοποιηθεί σε συστάδες τις Google [4]. Τα δεδομένα που κατασκευάζονται στην προκειμένη περίπτωση αντιπροσωπεύουν το μέγεθος μια εργασίας σε όρους εκατομμυρίων εντολών (Million Instructions - MI). Κατά την εκτέλεση του Αλγόριθμου 10, γίνεται εισαγωγή του επιθυμητού πλήθους εργασιών και παράγεται το σύνολο δεδομένων. Τα μεγέθη εργασιών που κατασκευάστηκαν εξάγονται όλα μαζί σε ένα αρχείο κειμένου ”GoCJ_Dataset_XXX.txt”, όπου το XXX είναι το πλήθος των εργασιών που περιέχει το αρχείο. Για παράδειγμα το αρχείο με το όνομα ”GoCJ_Dataset_250.txt” περιέχει δεδομένα για 250 εργασίες, με κάθε γραμμή να αντιπροσωπεύει μια εργασία. Το μέγεθος κάθε εργασίας μετριέται με τη μονάδα μέτρησης εκατομμύριου εντολών ανά δευτερόλεπτο (MIPS).

Αλγόριθμος 10: Αλγόριθμος κατασκευής συνόλου δεδομένων [4]

```
Input: Number of Tasks, File Original_DataSet.txt
Output: taskList
fileReader ← readFile(Original_DataSet.txt)
bufferReader ← read(fileReader)
Percentage ← 0
taskSize ← 0
taskList ← Empty
while bufferReader not empty do
    taskSize ← Long.parseLong(bufferReader)
    dataTable.add(Percentage, taskSize)
    Percentage += 2
end
while i in No of Tasks do
    rand ← Random.NextInt(100)
    if rand%2 = 0 then
        taskList[i] = dataTable.get(rand)
    end
    else
        taskList[i] = getJobSize(rand)
    end
    i++
end
```

Στον Πίνακα 4.5 απεικονίζονται όλες οι ρυθμίσεις του συστήματος που προσομοιώθηκε με τη χρήση του CloudSim. Στις εκτελέσεις που θα παρουσιαστούν παρακάτω χρησιμοποιήθηκε πληθώρα διαφορετικών VM, των οποίων το μέγεθος MIPS επιλέγεται από το αντίστοιχο αρχείο "VMs_XX", όπου το XX είναι το πλήθος των VM που περιέχει το αρχείο. Η έκδοση του CloudSim που χρησιμοποιήθηκε είναι η 3.0.3.

Data centers	1
Total Host Machines	1
System Architecture	x86
Operating System	Linux
VM Monitor	Xen
VM RAM	256 MB
VM Number of CPUs	1
VM Bandwidth	100 Mbps
VM Image Size	1,000 MB

Πίνακας 4.5: Ρυθμίσεις Συστήματος

4.4 Αποτελέσματα αλγορίθμων

Η προσομοίωση στο CloudSim επικεντρώθηκε κυρίως στην εκτέλεση των δύο νεότερων Ουγγρικών αλγορίθμων, του τροποποιημένου [2] και του προτεινόμενου από το Κεφάλαιο 3. Στην Ενότητα 4.1.2, έπειτα από μια σειρά εκτέλεσης δοκιμαστικών δεδομένων, τα αποτελέσματα ανέδειξαν πως ήταν οι δύο επικρατέστεροι αλγόριθμοι. Για την προσομοίωση των δυο Ουγγρικών αλγορίθμων στο CloudSim χρειάζεται να γίνει υλοποίηση των αλγορίθμων σε Java, ώστε οι κώδικες να είναι συμβατοί με το framework.

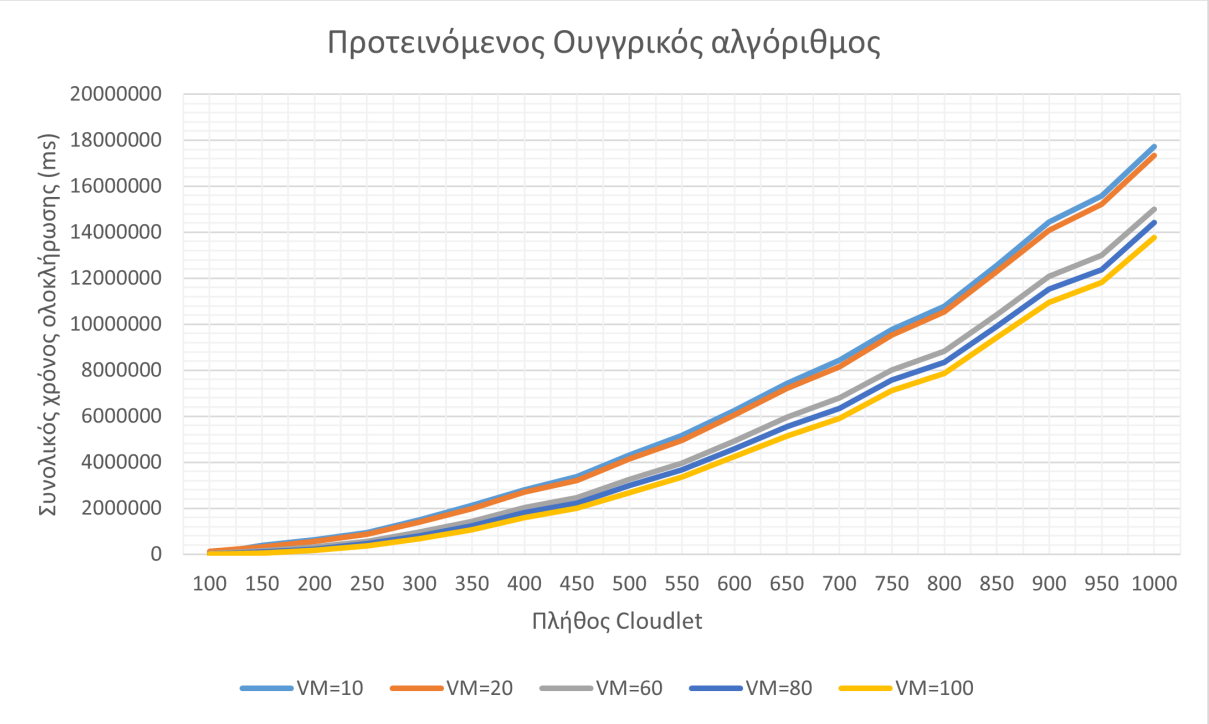
Για να εφαρμοστεί σωστά ο Ουγγρικός αλγόριθμος στο CloudSim θεωρείται απαραίτητο να δημιουργηθεί ένας πίνακας κόστους που να αντικατοπτρίζει ορθά το κόστος ανάθεσης κάθε Cloudlet σε κάθε VM. Σύμφωνα με το [17], ο πίνακας κόστους παρουσιάζει τον χρόνο εκτέλεσης ενός $Cloudlet_i$ σε ένα VM_j, όπου κάθε χρόνος είναι ίσος με το μέγεθος MIPS του Cloudlet δια το συνολικό μέγεθος του ανάλογου VM. Ο αντίστοιχος τύπος απεικονίζεται με την Εξίσωση 4.1.

$$CostMatrix[i][j] = lengthofClouddet[i]/MIPSofVirtualMachine[j] \tag{4.1}$$

Πραγματοποιήθηκαν συνολικά 19 εκτελέσεις των αλγορίθμων για διαφορετικό πλήθος εργασιών. Η μικρότερη ανάθεσης που εκτελέστηκε περιείχε 100 διαφορετικές εργασίες, ενώ σε κάθε εκτέλεση το πλήθος τους αυξάνεται κατά 50, μέχρι να φτάσουν στην τελική εκτέλεση δρομολογώντας 1,000 εργασίες. Επίσης, τα αποτελέσματα εκτελέσεων των αλγορίθμων χωρίζονται σε περαιτέρω κατηγορίες, αυτή τη φορά μεταβάλλοντας το πλήθος των διαθέσιμων VM στο νέφος. Το πλήθος των VM που επιλέχθηκαν προς χρήση ξεκινάει από 10 και καταλήγει στη χρήση 100 VM. Οι ενδιάμεσες τιμές που επιλέχθηκαν είναι 20, 60 και 80 VM, εκτελώντας όλους τους αλγόριθμους για πέντε διαφορετικές περιπτώσεις.

Με τη χρήση του Σχήματος 4.14, είναι δυνατή η μελέτη της συμπεριφοράς του προτεινόμενου Ουγγρικού αλγόριθμου. Παρατηρώντας το σχήμα, διαπιστώνεται πως η συμπεριφορά του αλγορίθμου μεταβάλλεται έντονα όσο αλλάζει το πλήθος των διαθέσιμων προς ανάθεση VM. Θεωρείται λογικό πως όσο αυξάνεται το πλήθος των αξιοποιήσιμων VM θα γίνεται καλύτερη κατανομή εργασιών στο νέφος. Η μεγάλη απόκλιση διακρίνεται μεταξύ της χρήσης 20 και 60 VM, με τη δεύτερη

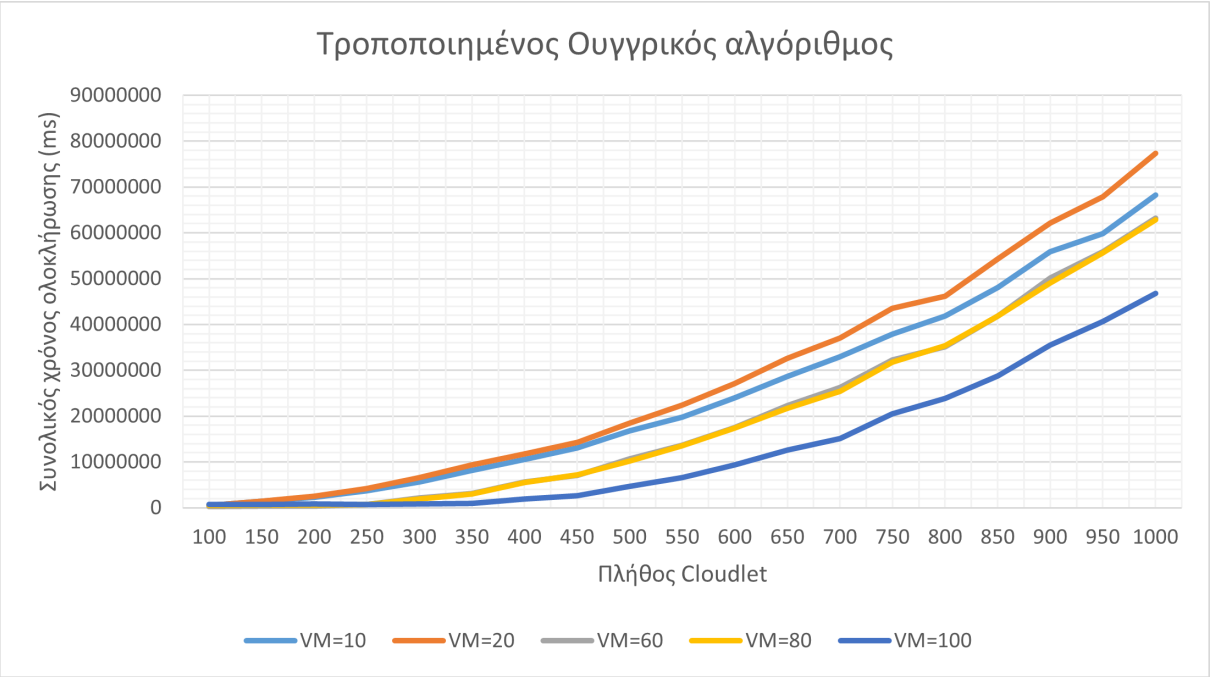
επιλογή να φέρει καλύτερα αποτελέσματα. Επιπλέον, όπως φαίνεται στο σχήμα επιλέγοντας 80 ή 100 VM δεν αλλάζει ιδιαίτερα το τελικό αποτέλεσμα. Οπότε στην προκειμένη περίπτωση, η επιλογή χρήσης 60 VM είναι ιδανική, φέροντας καλύτερα αποτελέσματα με μικρότερο κόστος χρήσης.



Σχήμα 4.14: Αποτελέσματα προτεινόμενου Ουγγρικού αλγόριθμου στο CloudSim

Εξετάζοντας το Σχήμα 4.15, παρατηρείται αντίστοιχη συμπεριφορά και για τον τροποποιημένο Ουγγρικό αλγόριθμο. Σε αυτή τη περίπτωση όμως, η χρήση διαφορετικού πλήθους VM είναι ξεκάθαρη, με τη χρήση περισσότερων VM να καθίσταται κατά μακράν ως η καλύτερη επιλογή. Συνολικά, όλα τα αποτελέσματα των αλγορίθμων που προσομοιώθηκαν με το CloudSim απεικονίζονται στο Σχήμα 4.16. Σε αυτό φαίνεται ξεκάθαρα πως ο χειρότερος αλγόριθμος είναι ο τροποποιημένος Ουγγρικός αλγόριθμος. Σε αντίθεση με τα αποτελέσματα της Ενότητας 4.1.2, ο προτεινόμενος Ουγγρικός αλγόριθμος έχει μακράν τα καλύτερα αποτελέσματα. Σε όλες τις διαφορετικές περιπτώσεις εφαρμογής των αλγόριθμων, ο προτεινόμενος Ουγγρικός αλγόριθμος ήταν περίπου 75,5% καλύτερος από τον τροποποιημένο Ουγγρικό αλγόριθμο.

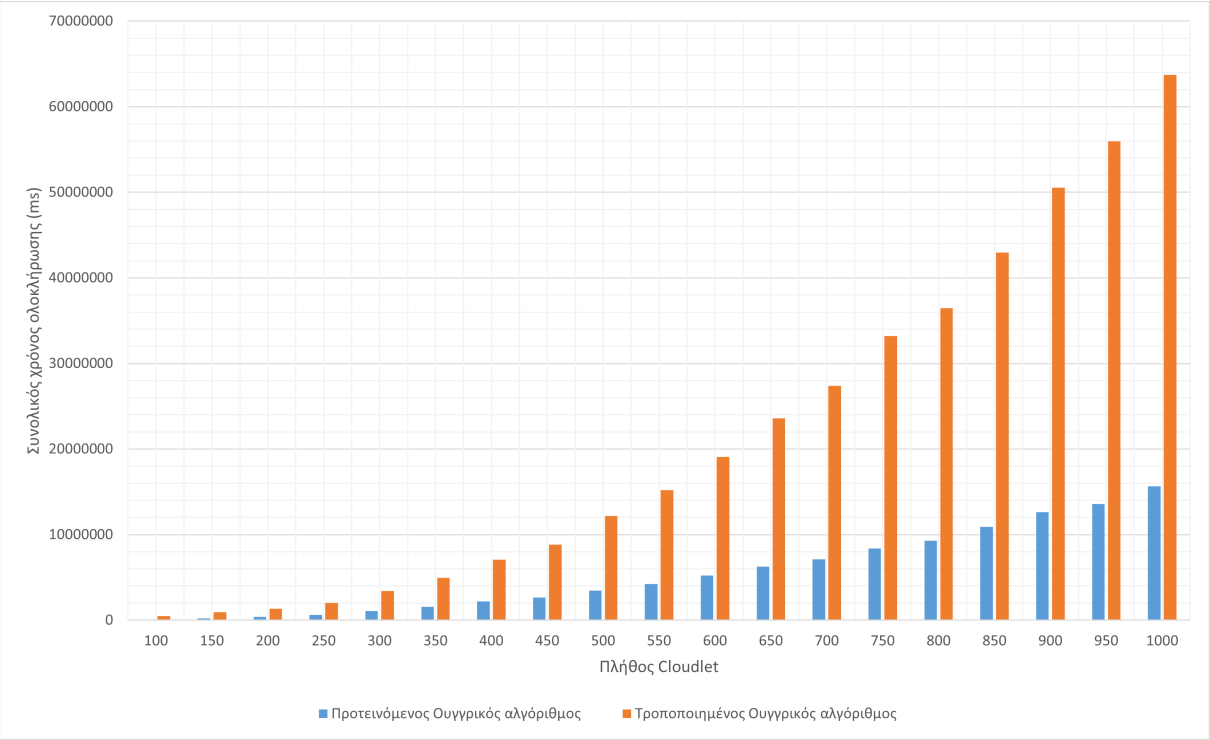
Ωστόσο, αν το μέγεθος μια εργασίας είναι ίδιο και ο χώρος ενός VM παραμένει σταθερός, ο πίνακας κόστους που δημιουργείται από την παραπάνω εξίσωση δεν αντικατοπτρίζει ρεαλιστικά τη συμπεριφορά ενός νέφους. Σε ένα περιβάλλον



Σχήμα 4.15: Αποτελέσματα τροποποιημένου Ουγγρικού αλγόριθμου στο CloudSim

υπολογιστικού νέφους, είναι πιθανό δύο VM με ίδια χαρακτηριστικά, φαινομενικά πανομοιότυπα δηλαδή, να έχουν απόκλιση στις αποδόσεις τους. Ορισμένοι παράγοντες που συμβάλλουν σε αυτό το φαινόμενο μπορεί να σχετίζονται με την κατανομή των διαθέσιμων πόρων που έχει ένα VM ή να εμπλέκονται κάποιοι εξωτερικοί παράγοντες, όπως ο χρόνος μεταφοράς ή η καθυστέρηση του δικτύου. Επιπρόσθετα, η απόδοση ενός VM δεν εξαρτάται αποκλειστικά από τα χαρακτηριστικά του, αλλά και από πιθανές παρεμβολές γειτονικών VM που βρίσκονται στον ίδιο Host [18].

Από τη φύση τους, οι δύο Ουγγρικοί αλγόριθμοι έχουν την τάση να δρομολογούν τις εργασίες με βάση το μικρότερο διαθέσιμο κόστος του πίνακα. Αφού δρομολογήσουν ένα Cloudlet σε κάθε διαθέσιμο VM, οι επόμενες αναθέσεις θα δρομολογηθούν σύμφωνα με τις μικρότερες τιμές κόστους. Συγκεκριμένα, για τον προτεινόμενο Ουγγρικό αλγόριθμο όλες οι εργασίες από εκείνο το σημείο μέχρι την ολοκλήρωση της διαδικασίας ανάθεσης θα δρομολογηθούν στο VM που διαθέτει τον μεγαλύτερο χώρο. Σύμφωνα με την Συνάρτηση 4.1, ανεξαρτήτως του μεγέθους του Cloudlet, η τιμή στον πίνακα κόστους εξαρτάται από το μέγεθος του ανάλογου VM. Επειδή το μέγεθος του Cloudlet παραμένει σταθερό, όσο αυξάνεται το μέγεθος των VM, το πηλίκο που αντιπροσωπεύει το κόστος σε κάθε θέση του πίνακα μειώνεται, με αποτέλεσμα τα μικρότερα κόστη ανάθεσης για κάθε εργασία



Σχήμα 4.16: Αποτελέσματα στο CloudSim

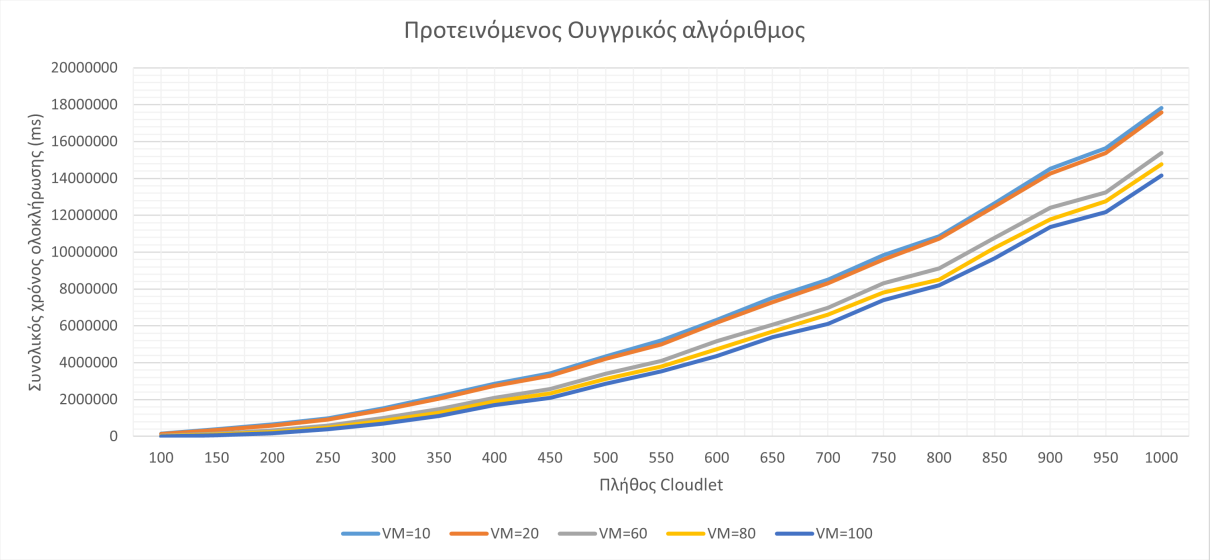
να αντιστοιχούν σε ένα VM.

Για αυτούς τους λόγους, κατά τη δημιουργία του πίνακα κόστους του προβλήματος εισάγεται μια μεταβλητή βαρύτητας, η οποία λαμβάνει τιμές μεταξύ 0.8 και 1.5. Για κάθε στοιχείο του πίνακα κόστους, η μεταβλητή λαμβάνει τυχαία μια τιμή στο εύρος και πολλαπλασιάζεται με αυτή. Ο νέος τύπος κατασκευής του πίνακα κόστους παρουσιάζεται με τη Συνάρτηση 4.2. Με τη χρήση της νέας μεταβλητής, ο Ουγγρικός αλγόριθμος δεν θα οδηγείται πλέον στη δρομολόγηση εργασιών αποκλειστικά σε ένα VM.

$$CostMatrix[i][j] = (lengthofClouddet[i] * k)/MIPSofVirtualMachine[j] \quad (4.2)$$

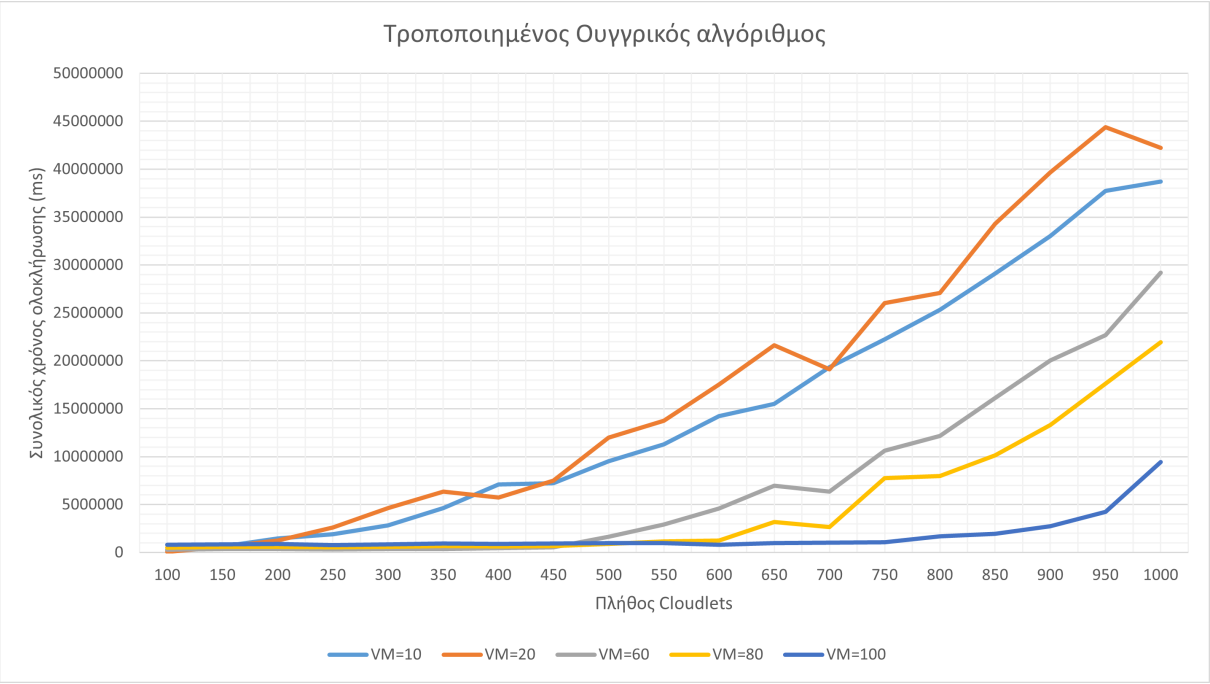
Συγκρίνοντας τα αποτελέσματα που προκύπτουν από τα διαφορετικά σενάρια, γίνεται φανερό πως με τη χρήση της μεταβλητής βαρύτητας, το συνολικό κόστος επίλυσης αυξάνεται και για τους δύο Ουγγρικούς αλγορίθμους. Η μεγαλύτερη διαφορά απεικονίζεται στο Σχήμα 4.18, όπου με τη χρήση περισσότερων VM το τελικό αποτέλεσμα είναι ξεκάθαρα καλύτερο. Γενικά, συγκρίνοντας τα αποτελέσματα που έγινε χρήση της μεταβλητής βαρύτητας με αυτά χωρίς τη μεταβλητή,

παρατηρείται πως η αξιοποίηση 100 VM μαζί με τη μεταβλητή οδηγεί σε μικρότερο χρόνο δρομολόγησης των εργασιών. Το ίδιο δεν μπορεί να παρατηρηθεί για τον προτεινόμενο Ουγγρικό αλγόριθμο. Αν και η συμπεριφορά του αλγόριθμου και στις δύο περιπτώσεις είναι παρόμοια, τα αποτελέσματα με τη χρήση της μεταβλητής έχουν λίγο χειρότερη ποιότητα λύσης, όπως φαίνεται από το Σχήμα 4.17. Εξαίρεση αποτελούν οι πρώτες εκτελέσεις, με το πλήθος των εργασιών αν είναι 100 και 150 αντίστοιχα. Σε αυτές μόνο τις περιπτώσεις ο προτεινόμενος Ουγγρικός αλγόριθμος φέρει καλύτερα κόστη ανάθεσης.

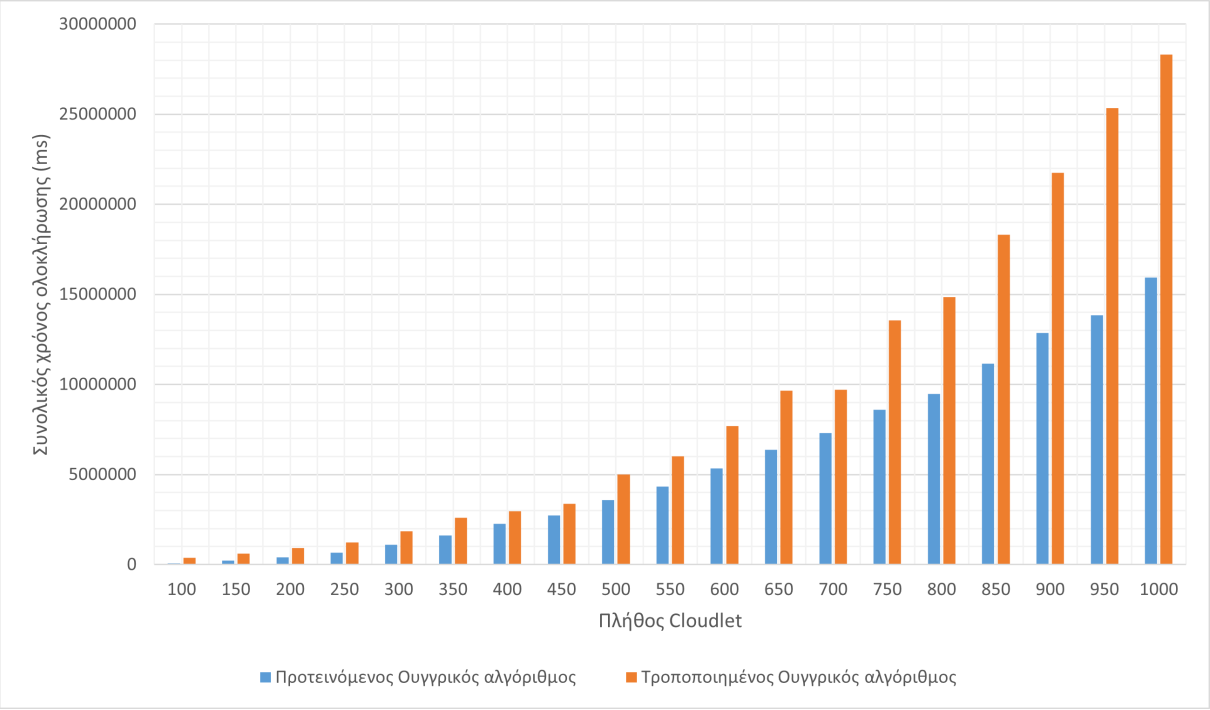


Σχήμα 4.17: Αποτελέσματα προτεινόμενου Ουγγρικού αλγόριθμου στο CloudSim

Συγκρίνοντας τη συνολική απόδοση των αλγορίθμων, φαίνεται από το Σχήμα 4.19 πως και οι δύο αλγόριθμοι κινούνται στα ίδια πλαίσια με προηγουμένως. Παρόλο που ο τροποποιημένος Ουγγρικός αλγόριθμος έχει πολύ καλύτερα αποτελέσματα από ότι είχε πριν, ο προτεινόμενος αλγόριθμος συνεχίζει να έχει με διαφορά μικρότερο κόστος ανάθεσης.



Σχήμα 4.18: Αποτελέσματα τροποποιημένου Ουγγρικού αλγόριθμου στο CloudSim



Σχήμα 4.19: Αποτελέσματα στο CloudSim με χρήση μεταβλητής βαρύτητας

Κεφάλαιο 5

Συμπεράσματα

Το πρόβλημα της ανάθεσης εργασιών αποτελεί ένα από τα θεμελιώδη προβλήματα βελτιστοποίησης που διατυπώθηκε τη δεκαετία του 1950. Το πρόβλημα της ανάθεσης αναφέρεται στον τρόπο ανάθεσης ενός συνόλου εργασιών σε ένα σύνολο εργατών, είτε ελαχιστοποιώντας είτε μεγιστοποιώντας το τελικό κόστος. Συνήθως στα προβλήματα βελτιστοποίησης υπάρχουν ορισμένοι παράγοντες που περιορίζουν τον τρόπο ανάθεσης, περιπλέκοντας την επίλυση του προβλήματος. Ο Ουγγρικός αλγόριθμος είναι ένας κλασικός αλγόριθμος συνδυαστικής βελτιστοποίησης που έχει σκοπό να επιλύει προβλήματα ανάθεσης εργασιών. Ωστόσο, η εφαρμογή του αλγόριθμου έχει ως περιοριστικό παράγοντα τη χρήση ίσου πλήθους εργασιών και μηχανών. Τέτοιου είδους περιπτώσεις ονομάζονται ισορροπημένα προβλήματα ανάθεσης.

Στα πλαίσια της παρούσας διπλωματικής σκοπός είναι η μελέτη εκδοχών του Ουγγρικού αλγορίθμου που στοχεύουν στην επίλυση μη ισορροπημένων προβλημάτων, στις περιπτώσεις δηλαδή που οι εργασίες προς ανάθεση δεν είναι ίσες με το πλήθος των διαθέσιμων μηχανών. Πέρα από τη μελέτη δυο ήδη υπαρχόντων εκδοχών του Ουγγρικού αλγόριθμου, καθώς επίσης και του Dhouib-Matrix-AP2, ενός στοχαστικού ευρετικού αλγόριθμου, προτείνεται και μια νέα εκδοχή του Ουγγρικού αλγόριθμου. Εφαρμόζοντας τους αλγόριθμους σε 80 διαφορετικά δοκιμαστικά σύνολα δεδομένων, παρατηρείται πως ο αλγόριθμος που προτείνεται έχει λιγότερο από 0.07% απόκλιση από τη βέλτιστη λύση, ενώ είναι σχεδόν 29.5% ταχύτερος από την επόμενη καλύτερη εκδοχή του Ουγγρικού αλγόριθμου.

Η μετάβαση από τη θεωρία στην εφαρμογή του Ουγγρικού αλγόριθμου αποτελεί ένα σημαντικό κομμάτι για την αξιολόγηση του. Με στόχο την προσομοίωση

των αλγορίθμων σε περιβάλλον νέφους χρησιμοποιήθηκε το framework CloudSim. Για την προσομοίωση στο CloudSim επιλέχθηκε ο Ουγγρικός αλγόριθμος που προτείνεται στην εργασία και ο τροποποιημένος Ουγγρικός αλγόριθμος του Rabbani [2], που είχε την καλύτερη ποιότητα λύσεων. Οι δύο αλγόριθμοι εκτελέστηκαν δρομολογώντας πολλαπλές εργασίες διαφορετικού μεγέθους. Επιπλέον, έγινε χρήση διαφορετικού πλήθους VM, ώστε να εξεταστούν οι συμπεριφορές των αλγορίθμων για διάφορα σενάρια εκτέλεσης. Παρατηρήθηκε πως ο προτεινόμενος αλγόριθμος όχι μόνο διατήρησε τον μικρότερο χρόνο εκτέλεσης, αλλά το τελικό κόστος ανάθεσης ήταν περίπου 75.5% μικρότερο σε σχέση με τον τροποποιημένο Ουγγρικό αλγόριθμο.

Παρά την απλότητα που διαθέτει ο Ουγγρικός αλγόριθμος, τα τελευταία κυρίως χρόνια έχει ξεκινήσει μια προσπάθεια μετεξέλιξης του ώστε να ταιριάζει στις ανάγκες που προκύπτουν με την εξέλιξη της τεχνολογίας. Αν και η μετατροπή του Ουγγρικού αλγόριθμου, ώστε να έχει τη δυνατότητα επίλυσης μη ισορροπημένων προβλημάτων, αποτελεί σημαντικό επίτευγμα για τον τομέα της πληροφορικής, χρειάζονται περαιτέρω μελέτες για να εφαρμοστεί στην πράξη. Εισάγοντας περισσότερους περιορισμούς, όπως ο μέγιστος χρόνος ή χώρος που μπορεί να διαθέτει ένα VM, είναι μια πιθανή επέκταση της έρευνας πάνω στον Ουγγρικό αλγόριθμο. Σχετικά με την εφαρμογή του αλγορίθμου σε περιβάλλον cloud, ο αλγόριθμος μπορεί να επεκταθεί προσθέτοντας περιορισμό στο πλήθος των εργασιών που έχει τη δυνατότητα να διαχειριστεί ένα VM. Παρόλο που οι προτεινόμενες επεκτάσεις αυξάνουν την πολυπλοκότητα του αλγόριθμου, σταδιακά μπορεί να γίνει προσαρμογή του με σκοπό να μπορεί να εφαρμόζεται σε ρεαλιστικά σενάρια.

Βιβλιογραφία

[1] A. Kumar, "A modified method for solving the unbalanced assignment problems," Applied Mathematics and Computation, vol. 176, no. 1, pp. 76–82, 2006.

[2] Q. Rabbani, A. Khan, and A. Quddoos, "Modified hungarian method for unbalanced assignment problem with multiple jobs," Applied Mathematics and Computation, vol. 361, pp. 493–498, 2019.

[3] S. Dhouib, "Innovative method to solve the minimum spanning tree problem: The dhouib-matrix-mstp (dm-mstp)," Results in Control and Optimization, vol. 14, p. 100359, 2024.

[4] A. Hussain and M. Aleem, "Gocj: Google cloud jobs dataset for distributed and cloud computing infrastructures," Data, vol. 3, no. 4, 2018.

[5] H. W. Kuhn, "The hungarian method for the assignment problem," Naval Research Logistics Quarterly, vol. 2, no. 1-2, pp. 83–97, 1955.

[6] N. Tomizawa, "On some techniques useful for solution of transportation network problems," Networks, vol. 1, no. 2, pp. 173–194, 1971.

[7] S. Dhouib, "An intelligent assignment problem using novel heuristic: The dhouib-matrix-ap1 (dm-ap1): Novel method for assignment problem," International Journal of Intelligent Systems and Applications in Engineering, vol. 10, p. 135–141, Mar. 2022.

[8] S. Dhouib, "Novel optimization method for unbalanced assignment problems with multiple jobs: The dhouib-matrix-ap2," Intelligent Systems with Applications, vol. 17, p. 200179, 2023.

[9] W. Liuyi, H. Zongtao, L. Chengju, and C. Qijun, "Graph based twin cost matrices for unbalanced assignment problem with improved ant colony algorithm," Results in Applied Mathematics, vol. 12, p. 100207, 2021.

[10] J. Majumdar and A. K. Bhunia, "An alternative approach for unbalanced assignment problem via genetic algorithm," Applied Mathematics and Computation, vol. 218, no. 12, pp. 6934–6941, 2012.

[11] P. M. Mell and T. Grance, "The NIST definition of cloud computing," tech. rep., Gaithersburg, MD, 2011.

[12] J. Naren, S. Sowmya, and P. Deepika, "Layers of cloud – iaas, paas and saas: A survey," International Journal of Computer Science and Information Technology, vol. Vol. 5 (3), pp. 4477 – 4480, 06 2014.

[13] S. S. Manvi and G. Krishna Shyam, “Resource management for infrastructure as a service (iaas) in cloud computing: A survey,” Journal of Network and Computer Applications, vol. 41, pp. 424–440, 2014.

[14] R. Buyya, S. Pandey, and C. Vecchiola, “Cloudbus toolkit for market-oriented cloud computing,” in Cloud Computing (M. G. Jaatun, G. Zhao, and C. Rong, eds.), (Berlin, Heidelberg), pp. 24–44, Springer Berlin Heidelberg, 2009.

[15] J. Byrne, S. Svorobej, K. M. Giannoutakis, D. Tzovaras, P. J. Byrne, P. O. Östberg, A. Gourinovitch, and T. Lynn, “A review of cloud computing simulation platforms and related environments,” in Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,, pp. 679–691, INSTICC, SciTePress, 2017.

[16] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” Software: Practice and Experience, vol. 41, no. 1, pp. 23–50, 2011.

[17] M. I. Bala and M. A. Chishti, “Load balancing in cloud computing using hungarian algorithm,” International Journal of Wireless and Microwave Technologies, vol. 9, pp. 1–10, 11 2019.

[18] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, “Modeling virtual machine performance: challenges and approaches,” SIGMETRICS Perform. Eval. Rev., vol. 37, p. 55–60, jan 2010.

Παραρτήματα

Παράρτημα Α΄

Αποτελέσματα αλγορίθμων

Ακολουθούν τα αναλυτικά αποτελέσματα των εκτελέσεων του Κεφαλαίου 4.1.

Gurobi	Number of integer	Optimal Solution	Modified Hungarian Cost	Proposed Hungarian Cost	Kumar Hungarian Cost	Dhouib-Matrix-AP2 Cost
10 x 15	150	878	886	878	1,089	1,718
10 x 20	200	1,287	1,318	1,291	1,543	1,551
12 x 20	240	1,306	1,312	1,373	1,480	1,410
10 x 25	250	1,524	1,524	1,579	1,765	2,325
15 x 20	300	1,263	1,399	1,316	1,467	1,527
18 x 20	360	1,253	1,332	1,311	1,384	1,360
15 x 25	375	1,417	1,413	1,510	1,692	1,857
14 x 32	448	1,910	1,913	1,954	2,412	2,548
15 x 30	450	1,787	1,823	1,815	2,104	2,381
20 x 25	500	1,435	1,449	1,437	1,600	1,604
15 x 35	525	2,094	2,146	2,110	2,444	2,937
15 x 35	525	2,055	2,076	2,274	2,460	3,442
22 x 27	594	1,536	1,605	1,709	2,008	2,013
20 x 30	600	1,737	1,849	1,877	2,080	2,217
25 x 30	750	1,684	1,751	1,780	1,949	1,860
27 x 30	810	1,711	1,835	1,832	1,942	1,899
20 x 50	1,000	2,861	2,981	3,074	3,214	4,137
30 x 35	1,050	1,977	1,962	2,071	2,236	2,233
35 x 40	1,400	2,213	2,234	2,405	2,696	2,506
30 x 55	1,650	3,028	3,096	3,400	3,301	4,221
30 x 65	1,950	3,563	3,557	3,910	4,416	5,646
30 x 65	1,950	3,505	3,634	3,609	3,911	4,752
40 x 75	3,000	3,982	4,061	4,250	4,500	6,687
50 x 75	3,750	3,942	4,040	4,160	4,233	5,497
50 x 75	3,750	3,938	3,967	4,173	4,381	5,198
55 x 80	4,400	4,196	4,216	4,338	4,463	5,874
60 x 85	5,100	4,444	4,524	4,581	4,816	6,506
60 x 100	6,000	5,219	5,299	5,665	5,822	7,653
85 x 100	8,500	5,189	5,286	5,423	7,545	5,856
85 x 110	9,350	5,698	5,814	5,956	6,051	7,085

Πίνακας Α΄.1: Κόστη ανάθεσης μικρών προβλημάτων

Gurobi	Number of integer	Modified Hungarian Time (sec)	Proposed Hungarian Time (sec)	Kumar Hungarian Time (sec)	Dhouib-Matrix-AP2 Time (sec)
10 x 15	150	0.006	0.004	0.009	0.002
10 x 20	200	0.002	0.003	0.011	0.003
12 x 20	240	0.005	0.005	0.003	0.003
10 x 25	250	0.003	0.007	0.014	0.004
15 x 20	300	0.007	0.002	0.012	0.008
18 x 20	360	0.005	0.004	0.015	0.007
15 x 25	375	0.008	0.005	0.011	0.005
14 x 32	448	0.007	0.004	0.016	0.008
15 x 30	450	0.009	0.006	0.012	0.006
20 x 25	500	0.002	0.002	0.012	0.002
15 x 35	525	0.004	0.006	0.014	0.007
15 x 35	525	0.009	0.005	0.012	0.007
22 x 27	594	0.01	0.004	0.014	0.006
20 x 30	600	0.008	0.004	0.015	0.006
25 x 30	750	0.012	0.007	0.015	0.007
27 x 30	810	0.01	0.005	0.014	0.013
20 x 50	1,000	0.014	0.006	0.018	0.019
30 x 35	1,050	0.01	0.005	0.02	0.014
35 x 40	1,400	0.017	0.017	0.018	0.019
30 x 55	1,650	0.014	0.01	0.021	0.025
30 x 65	1,950	0.016	0.011	0.025	0.039
30 x 65	1,950	0.015	0.01	0.024	0.039
40 x 75	3,000	0.021	0.007	0.032	0.056
50 x 75	3,750	0.024	0.016	0.041	0.071
50 x 75	3,750	0.026	0.022	0.033	0.06
55 x 80	4,400	0.028	0.025	0.038	0.07
60 x 85	5,100	0.034	0.021	0.042	0.081
60 x 100	6,000	0.04	0.035	0.047	0.122
85 x 100	8,500	0.077	0.07	0.071	0.135
85 x 110	9,350	0.087	0.076	0.064	0.169

Πίνακας Α΄.2: Χρόνοι εκτέλεσης μικρών προβλημάτων

Gurobi	Number of integer	Optimal Solution	Modified Hungarian Cost	Proposed Hungarian Cost	Kumar Hungarian Cost
100 x 250	25,000	12,765	12,837	13,117	14,564
150 x 500	75,000	25,326	25,427	25,684	33,060
300 x 400	120,000	20,070	20,059	20,643	33,035
350 x 550	192,500	27,566	27,663	28,107	45,369
250 x 850	212,500	42,707	42,765	43,360	57,345
450 x 650	292,500	32,546	32,586	33,071	62,179
500 x 750	375,000	37,526	37,667	38,031	66,517
500 x 900	450,000	45,034	45,183	45,589	82,119
600 x 950	570,000	47,517	47,586	48,082	98,523
750 x 1,000	750,000	50,007	50,010	50,653	107,365
600 x 1,500	900,000	75,034	75,133	75,627	163,649
800 x 1,400	1,120,000	70,008	70,012	70,600	165,315
900 x 1,300	1,170,000	65,003	65,116	65,600	151,478
900 x 1,700	1,530,000	85,004	85,065	85,572	208,867
1,000 x 1,800	1,800,000	90,002	90,021	90,539	216,747
950 x 2,000	1,900,000	100,004	100,036	100,654	241,049
1,300 x 1,500	1,950,000	75,000	75,138	75,744	176,133
1,200 x 2,000	2,400,000	100,000	100,008	100,529	242,779
1,000 x 2,500	2,500,000	125,001	125,085	125,682	298,643
1,500 x 2,000	3,000,000	100,000	100,139	100,571	235,125
1,500 x 2,700	4,050,000	135,000	135,036	135,664	335,180
1,000 x 4,500	4,500,000	225,011	225,136	225,493	537,549
2,000 x 4,500	9,000,000	225,000	225,031	225,670	549,675
2,500 x 3,600	9,000,000	180,000	180,118	180,612	442,351
2,800 x 3,400	9,520,000	171,368	170,057	170,588	408,648
2,000 x 5,000	10,000,000	250,000	250,064	250,691	618,920
2,500 x 4,000	10,000,000	200,000	200,008	200,605	500,294
3,000 x 3,500	10,500,000	175,000	175,019	175,514	422,308
3,000 x 6,000	18,000,000	300,000	300,142	300,624	748,285
4,500 x 5,000	22,500,000	250,000	250,104	280,738	614,579

Πίνακας Α΄.3: Κόστη ανάθεσης μεσαίων προβλημάτων

Gurobi	Number of integer	Modified Hungarian Time (sec)	Proposed Hungarian Time (sec)	Kumar Hungarian Time (sec)
100 x 250	25,000	0.173	0.144	0.116
150 x 500	75,000	0.632	0.498	0.511
300 x 400	120,000	1.742	1.331	0.706
350 x 550	192,500	3.197	2.383	3.282
250 x 850	212,500	2.530	1.920	17.520
450 x 650	292,500	5.980	4.460	4.350
500 x 750	375,000	8.047	5.874	4.711
500 x 900	450,000	9.430	6.770	12.590
600 x 950	570,000	14.232	10.184	2.850
750 x 1,000	750,000	23.241	16.787	8.344
600 x 1,500	900,000	22.210	15.850	4.710
800 x 1,400	1,120,000	36.230	25.270	5.650
900 x 1,300	1,170,000	42.330	29.090	5.440
900 x 1,700	1,530,000	54.410	38.280	8.020
1,000 x 1,800	1,800,000	70.510	49.510	8.590
950 x 2,000	1,900,000	71.270	50.210	9.490
1 300 x 1,500	1,950,000	102.110	72.860	8.800
1,200 x 2,000	2,400,000	114.720	81.100	10.950
1,000 x 2,500	2,500,000	98.615	69.323	85.067
1,500 x 2,000	3,000,000	178.310	125.940	13.100
1,500 x 2,700	4,050,000	14.679	171.826	19.762
1,000 x 4,500	4,500,000	184.040	130.682	23.977
2,000 x 4,500	9,000,000	701.800	492.476	50.154
2,500 x 3,600	9,000,000	833.980	575.270	38.000
2,800 x 3,400	9,520,000	997.230	691.910	41.230
2,000 x 5,000	10,000,000	769.240	537.320	46.630
2,500 x 4,000	10,000,000	971.820	679.230	45.690
3,000 x 3,500	10,500,000	1,227.460	857.490	48.190
3,000 x 6,000	18,000,000	2,079.120	1,433.800	92.930
4,500 x 5,000	22,500,000	3,898.510	2,731.840	107.420

Πίνακας Α΄.4: Χρόνοι εκτέλεσης μεσαίων προβλημάτων

Gurobi	Number of integer	Optimal Solution	Dhouib-Matrix-AP2			
			No of Iterations (1) Cost	No of Iterations (5) Cost	No of Iterations (10) Cost	No of Iterations (100) Cost
100 x 250	25,000	12,765	22,938	22,938	22,938	22,993
150 x 500	75,000	25,326	49,527	50,291	50,347	50,347
300 x 400	120,000	20,070	27,541	27,541	27,172	27,172
350 x 550	192,500	27,566	127,797	42,944	42,162	41,314
250 x 850	212,500	42,707	82,485	88,246	86,178	88,080
450 x 650	292,500	32,546	46,783	46,661	46,203	47,943
500 x 750	375,000	37,526	55,240	55,240	55,240	56,102
500 x 900	450,000	45,034	74,002	71,351	72,724	71,351
600 x 950	570,000	47,517	71,713	71,665	74,378	71,665
750 x 1,000	750,000	50,007	69,148	68,150	69,148	68,150
600 x 1,500	900,000	75,034	138,232	139,162	136,209	133,879
800 x 1,400	1,120,000	70,008	112,163	112,037	109,787	109,787
900 x 1,300	1,170,000	65,003	92,225	92,492	90,930	88,012
900 x 1,700	1,530,000	85,004	142,711	135,154	135,154	134,463
1,000 x 1,800	1,800,000	90,002	133,941	147,523	144,672	144,672
950 x 2,000	1,900,000	100,004	161,705	168,472	168,472	114,506
1,300 x 1,500	1,950,000	75,000	88,992	88,814	88,814	88,363
1,200 x 2,000	2,400,000	100,000	158,904	156,250	156,250	110,808
1,000 x 2,500	2,500,000	125,001	212,519	210,705	210,705	140,420
1,500 x 2,000	3,000,000	100,000	136,547	137,406	137,406	137,406
1,500 x 2,700	4,050,000	135,000	199,854	201,436	201,436	201,436
1,000 x 4,500	4,500,000	225,011	413,506	363,517	363,268	402,034
2,000 x 4,500	9,000,000	225,000	268,583	274,646	274,646	274,646
2,500 x 3,600	9,000,000	180,000	240,347	249,787	249,787	230,250
2,800 x 3,400	9,520,000	171,368	207,264	212,191	213,191	213,191
2,000 x 5,000	10,000,000	250,000	449,231	316,653	316,653	316,653
2,500 x 4,000	10,000,000	200,000	231,128	283,199	283,199	183,199
3,000 x 3,500	10,500,000	175,000	211,999	208,951	208,951	208,951
3,000 x 6,000	18,000,000	300,000	373,295	342,874	342,874	342,874
4,500 x 5,000	22,500,000	250,000	285,984	280,886	280,886	280,886

Πίνακας Α΄.5: Κόστη ανάθεσης μεσαίων προβλημάτων DM-AP2

Gurobi	Number of integer	Dhouib-Matrix-AP2			
		No of Iterations (1) Time (sec)	No of Iterations (5) Time (sec)	No of Iterations (10) Time (sec)	No of Iterations (100) Time (sec)
100 x 250	25,000	0.094	0.215	0.372	0.680
150 x 500	75,000	0.283	0.664	1.167	3.767
300 x 400	120,000	0.407	0.879	1.435	3.385
350 x 550	192,500	0.776	31.618	32.517	37.686
250 x 850	212,500	0.800	1.880	4.050	12.490
450 x 650	292,500	1.010	2.020	3.900	9.420
500 x 750	375,000	1.273	2.743	4.502	13.812
500 x 900	450,000	1.580	3.470	6.070	15.580
600 x 950	570,000	1.917	4.200	7.034	18.133
750 x 1,000	750,000	2.449	5.397	8.791	12.782
600 x 1,500	900,000	3.260	7.911	13.440	44.560
800 x 1,400	1,120,000	3.850	8.490	15.090	39.010
900 x 1,300	1,170,000	4.120	9.500	15.570	47.630
900 x 1,700	1,530,000	5.240	11.930	23.560	70.310
1,000 x 1,800	1,800,000	6.010	13.370	22.600	59.940
950 x 2,000	1,900,000	6.720	16.650	26.990	113.090
1,300 x 1,500	1,950,000	6.280	13.470	22.510	54.630
1,200 x 2,000	2,400,000	8.040	17.720	30.120	80.720
1,000 x 2,500	2,500,000	8.847	93.041	119.680	105.044
1,500 x 2,000	3,000,000	9.880	23.040	36.670	87.320
1,500 x 2,700	4,050,000	14.679	32.415	23.047	143.865
1,000 x 4,500	4,500,000	19.998	47.975	93.920	482.661
2,000 x 4,500	9,000,000	31.110	78.100	134.590	378.730
2,500 x 3,600	9,000,000	30.750	65.580	106.960	386.060
2,800 x 3,400	9,520,000	30.510	37.730	43.530	146.530
2,000 x 5,000	10,000,000	35.830	49.360	61.060	302.720
2,500 x 4,000	10,000,000	36.130	42.850	51.370	202.690
3,000 x 3,500	10,500,000	34.520	39.610	46.870	170.110
3,000 x 6,000	18,000,000	66.570	82.140	451.990	451.990
4,500 x 5,000	22,500,000	74.280	88.740	130.500	358.980

Πίνακας Α΄.6: Χρόνοι εκτέλεσης μεσαίων προβλημάτων DM-AP2

Gurobi	Number of integer	Optimal Solution	Modified Hungarian Cost	Proposed Hungarian Cost	Kumar Hungarian Cost
1,000 x 25,000	25,000,000	1,250,022	1,250,043	1,250,737	3,090,790
2,000 x 13,000	26,000,000	650,000	650,053	650,542	1,626,801
2,000 x 13,500	27,000,000	675,000	675,030	675,855	1,692,495
2,000 x 14,000	28,000,000	700,000	700,094	700,711	1,744,815
1,000 x 30,000	30,000,000	1,500,034	1,500,118	1,500,648	3,716,660
1,500 x 20,000	30,000,000	1,000,000	1,000,018	1,000,727	2,489,940
1,700 x 18,000	30,600,000	900,000	900,148	900,709	2,258,870
1,700 x 18,500	31,450,000	925,000	925,068	925,799	2,309,240
1,600 x 20,000	32,000,000	1,000,000	1,000,022	1,000,638	2,499,700
1,610 x 20,000	32,200,000	1,000,000	1,000,096	1,000,584	2,486,740
1,610 x 20,010	32,216,100	1,000,502	1,000,555	1,001,180	2,499,240
1,610 x 20,030	32,248,300	1,001,500	1,001,581	1,002,099	2,497,070
1,610 x 20,050	32,280,500	1,002,500	1,002,548	1,003,102	2,502,970
1,610 x 20,100	32,361,000	1,005,000	1,005,106	1,005,495	2,509,320
1,610 x 20,200	32,522,000	1,010,000	1,010,019	1,010,506	2,522,140
1,610 x 20,300	32,683,000	1,015,001	1,015,067	1,015,536	2,,524,000
1,610 x 20,400	32,844,000	1,020,001	1,020,099	1,020,681	2,559,950
1,610 x 20,500	33,005,000	1,025,000	1,025,039	1,025,693	2,561,840
1,610 x 20,600	33,166,000	1,030,000	1,030,087	1,030,779	2,585,310
1,610 x 20,700	33,327,000	1,035,000	1,035,142	1,035,694	2,582,970

Πίνακας Α΄.7: Κόστη ανάθεσης μεγάλων προβλημάτων

Gurobi	Number of integer	Optimal Solution	Modified Hungarian Time (sec)	Proposed Hungarian Time (sec)	Kumar Hungarian Time (sec)
1,000 x 25,000	25,000,000	1,250,022	695.95	499.64	132.192
2,000 x 13,000	26,000,000	650,000	1,944.84	1,349.99	127.08
2,000 x 13,500	27,000,000	675,000	2,036.32	1,413.13	136.12
2,000 x 14,000	28,000,000	700,000	2,090	1,446.66	141.47
1,000 x 30,000	30,000,000	1,500,034	864.03	595.13	165.587
1,500 x 20,000	30,000,000	1,000,000	1,450.66	949.58	156.865
1,700 x 18,000	30,600,000	900,000	1,411.86	1,002.77	155.149
1,700 x 18,500	31,450,000	925,000	1,446.9	1,028.54	166.279
1,600 x 20,000	32,000,000	1,000,000	1,365.44	1,003.55	164.487
1,610 x 20,000	32,200,000	1,000,000	1,378.95	983.92	165.765
1,610 x 20,010	32,216,100	1,000,502	1,376.1	973.32	165.454
1,610 x 20,030	32,248,300	1,001,500	1,357.1	954.21	175.03
1,610 x 20,050	32,280,500	1,002,500	1,364.66	961.01	168.311
1,610 x 20,100	32,361,000	1,005,000	1,437.26	1,031.67	168.226
1,610 x 20,200	32,522,000	1,010,000	1,413.14	1,005.32	165.912
1,610 x 20,300	32,683,000	1,015,001	1,422.47	1,010.7	167.236
1,610 x 20,400	32,844,000	1,020,001	1,432.91	1,018.03	170.089
1,610 x 20,500	33,005,000	1,025,000	1,440.27	1,023.46	172.121
1,610 x 20,600	33,166,000	1,030,000	1,442.81	1,021.98	171.63
1,610 x 20,700	33,327,000	1,035,000	1,484.77	1,063.37	172.615

Πίνακας Α΄.8: Χρόνοι εκτέλεσης μεγάλων προβλημάτων

			Dhouib-Matrix-AP2		
Gurobi	Number of integer	Optimal Solution	No of Iterations (1) Cost	No of Iterations (5) Cost	No of Iterations (10) Cost
1,000 x 25,000	25,000,000	1,250,022	2,932,681	2,930,504	2,931,182
2,000 x 13,000	26,000,000	650,000	730,381	696,293	696,293
2,000 x 13,500	27,000,000	675,000	730,341	730,341	731,857
2,000 x 14,000	28,000,000	700,000	1,595,783	743,817	743,817
1,000 x 30,000	30,000,000	1,500,034	3,668,250	3,659,690	3,656,763
1,500 x 20,000	30,000,000	1,000,000	1,067,816	2,377,461	2,377,461
1,700 x 18,000	30,600,000	900,000	951,424	951,424	951,424
1,700 x 18,500	31,450,000	925,000	1,025,069	1,025,069	1,025,069
1,600 x 20,000	32,000,000	1,000,000	1,061,055	1,061,055	1,061,055
1,610 x 20,000	32,200,000	1,000,000	1,059,649	1,059,649	1,059,649
1,610 x 20,010	32,216,100	1,000,502	2,234,443	2,233,345	2,232,797
1,610 x 20,030	32,248,300	1,001,500	1,058,614	1,058,614	1,058,614
1,610 x 20,050	32,280,500	1,002,500	2,297,674	2,297,558	2,297,325
1,610 x 20,100	32,361,000	1,005,000	1,049,719	1,049,719	1,049,719
1,610 x 20,200	32,522,000	1,010,000	1,070,330	1,070,330	1,070,330
1,610 x 20,300	32,683,000	1,015,001	1,077,080	1,077,080	1,077,080
1,610 x 20,400	32,844,000	1,020,001	2,254,372	2,252,894	2,254,372
1,610 x 20,500	33,005,000	1,025,000	1,073,026	1,073,026	1,073,026
1,610 x 20,600	33,166,000	1,030,000	1,094,377	1,094,377	1,094,377
1,610 x 20,700	33,327,000	1,035,000	1,097,722	1,097,722	1,097,722

Πίνακας Α΄.9: Κόστη ανάθεσης μεγάλων προβλημάτων DM-AP2

			Dhouib-Matrix-AP2		
Gurobi	Number of integer	Optimal Solution	No of Iterations (1) Time (sec)	No of Iterations (5) Time (sec)	No of Iterations (10) Time (sec)
1,000 x 25,000	25,000,000	1,250,022	127.34	398.63	1,990.01
2,000 x 13,000	26,000,000	650,000	127.25	193.64	294.4
2,000 x 13,500	27,000,000	675,000	131.12	136.06	210.01
2,000 x 14,000	28,000,000	700,000	339.83	286.093	610.322
1,000 x 30,000	30,000,000	1,500,034	239.8	574.67	1,773.31
1,500 x 20,000	30,000,000	1,000,000	140.9	312.9	546.88
1,700 x 18,000	30,600,000	900,000	62.94	289.76	409.37
1,700 x 18,500	31,450,000	925,000	102.46	273.86	417.4
1,600 x 20,000	32,000,000	1,000,000	140.65	371.2	480.78
1,610 x 20,000	32,200,000	1,000,000	144.1	350.06	792.38
1,610 x 20,010	32,216,100	1,000,502	144.22	388.67	1,088.17
1,610 x 20,030	32,248,300	1,001,500	113.45	443.02	449.47
1,610 x 20,050	32,280,500	1,002,500	109.09	365.15	437.4
1,610 x 20,100	32,361,000	1,005,000	109.74	282.83	576.24
1,610 x 20,200	32,522,000	1,010,000	116.09	412.79	529.36
1,610 x 20,300	32,683,000	1,015,001	112.12	279.73	451.3
1,610 x 20,400	32,844,000	1,020,001	112.67	487.93	645.99
1,610 x 20,500	33,005,000	1,025,000	113.79	334.14	500.86
1,610 x 20,600	33,166,000	1,030,000	114.64	305.38	477.57
1,610 x 20,700	33,327,000	1,035,000	115.31	298.12	589.4

Πίνακας Α΄.10: Χρόνοι εκτέλεσης μεγάλων προβλημάτων DM-AP2