Σύγκριση Αλγορίθμων Βελτιστοποίησης Τοποθεσίας Αποθήκης: Gurobi (Pyomo) vs. Προσαρμοσμένος Αλγόριθμος Branch & Bound

Γρηγοράσκος Γεώργιος
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Δυτικής Μακεδονίας
Κοζάνη
29/01/2025
gio.grigoraskos@gmail.com / ece01797@uowm.gr

***Abstract*—**Αυτό το έγγραφο διερευνά ένα πρόβλημα βελτιστοποίησης τοποθεσίας αποθήκης**,** ένα βασικό παράδειγμα συνδυαστικής βελτιστοποίησης**.** Το πρόβλημα περιλαμβάνει τον καθορισμό της βέλτιστης τοποθέτησης των αποθηκών και την ανάθεση καταστημάτων για την ελαχιστοποίηση του κόστους**,** ενώ παράλληλα πληρούνται συγκεκριμένοι περιορισμοί**.** Η λύση χρησιμοποιεί μαθηματικά μοντέλα μικτού-ακέραιου γραμμικού προγραμματισμού **(MILP).**

***Index Terms*—**Συνδυαστική Βελτιστοποίηση**,** Γραμμικός Προγραμματισμός Μικτού Ακέραιου**,** Εφοδιαστική Αλυσίδα**,** Ελαχιστοποίηση Κόστους

Πίνακας Περιεχομένων

I. Εισαγωγή

A. Ιστορικό Υπόβαθρο

Το Πρόβλημα Τοποθεσίας Αποθηκών (Warehouse Location Problem - WLP) αποτελεί ένα από τα πιο σημαντικά και ευρέως μελετημένα προβλήματα συνδυαστικής βελτιστοποίησης, το οποίο διατυπώνεται συνήθως ως πρόβλημα μικτού Ακέραιου Γραμμικού Προγραμματισμού (MILP). Στο πλαίσιο του προβλήματος, η εταιρεία καλείται να προσδιορίσει την ιδανική τοποθεσία για τις αποθήκες της, με σκοπό τη βέλτιστη εξυπηρέτηση των καταστημάτων της και την ελαχιστοποίηση του συνολικού κόστους, το οποίο περιλαμβάνει τόσο το κόστος κατασκευής των αποθηκών όσο και το κόστος προμήθειας των προϊόντων από τις αποθήκες προς τα καταστήματα, ενώ ταυτόχρονα ο αριθμός των καταστημάτων που μπορεί να προμηθεύει η κάθε αποθήκη είναι περιορισμένος.

Τα προβλήματα αυτά είναι πολύπλοκα λόγω των πολλών πιθανών λύσεων. Πρέπει να ληφθούν υπόψη πολλοί παράγοντες (κόστοι, χωρητικότητα, διαθεσιμότητα). Οι αλγόριθμοι υπολογίζουν να βρουν την καλύτερη λύση που να ελαχιστοποιεί το συνολικό κόστος, ενώ παράλληλα να ικανοποιεί όλους τους περιορισμούς.

B. Κίνητρο

Στόχος της μελέτης είναι η σύγκριση ενός custom αλγορίθμου Branch & Bound (B&B) με τον Gurobi, έναν κορυφαίο εμπορικό solver που αξιοποιεί προηγμένες τεχνικές βελτιστοποίησης, μέσω της βιβλιοθήκης Pyomo. Ο custom αλγόριθμος Branch & Bound, που αναπτύχθηκε ειδικά για το συγκεκριμένο πρόβλημα, στοχεύει στην αποδοτική εξερεύνηση του χώρου λύσεων, προσφέροντας μία πιο στοχευμένη προσέγγιση, ενώ ο Gurobi εκμεταλλεύεται τις εξελιγμένες δυνατότητες του εμπορικού λογισμικού για να προσφέρει λύσεις σε πολύ σύντομο χρονικό διάστημα. Η σύγκριση μεταξύ των δύο μεθόδων αποσκοπεί στην αξιολόγηση της απόδοσης, της ακρίβειας και του χρόνου εκτέλεσης σε διαφορετικά μεγέθη και σύνθετα σενάρια προβλημάτων.

II. Μαθηματική Διατύπωση του Προβλήματος

Ορίζουμε τις ακόλουθες δυαδικές μεταβλητές απόφασης:

- X_i: 1 αν κατασκευαστεί η αποθήκη i, αλλιώς 0.
- Y_{ij}: 1 αν το κατάστημα j εξυπηρετείται από την αποθήκη i, αλλιώς 0.

Οι περιορισμοί είναι:

- Περιορισμός Χωρητικότητας Αποθηκών**:**

$$\sum_j Y_{ij} \leq \text{Χωρητικότητα}_i \cdot X_i, \quad \forall i \tag{1}$$

- Περιορισμός Ανάθεσης Καταστημάτων**:**

$$\sum_i Y_{ij} = 1, \quad \forall j \tag{2}$$

- Λογικός Περιορισμός**:**

$$Y_i j \leq X_i, \quad \forall i, j \tag{3}$$

Ο στόχος είναι να ελαχιστοποιηθεί:

$$\min \sum_i \text{Κόστος\_ανοίγματος}_i \cdot X_i + \sum_{i,j} \text{Κόστος\_Τροφοδοσίας}_{ij} \cdot Y_{ij} \tag{4}$$

III. Προσεγγίσεις Επίλυσης

A. *Gurobi (*Παραδοτέο *1)*

Ο Gurobi είναι ένας ισχυρός εμπορικός solver βελτιστοποίησης που χρησιμοποιεί προηγμένες τεχνικές για την επίλυση προβλημάτων Μικτού Ακέραιου Γραμμικού Προγραμματισμού (MILP). Συγκεκριμένα, αξιοποιεί:

- Ευρετικούς αλγορίθμους, οι οποίοι επιταχύνουν τη διαδικασία εύρεσης καλών εφικτών λύσεων.
- **Cutting Planes**, μια τεχνική που προσθέτει επιπλέον περιορισμούς (cuts) στο μοντέλο για να μειώσει τον εφικτό χώρο αναζήτησης και να επιταχύνει τη σύγκλιση.
- Παραλληλοποίηση, επιτρέποντας την εκμετάλλευση πολλαπλών πυρήνων του επεξεργαστή για τη βελτίωση της ταχύτητας επίλυσης.

Αυτές οι τεχνικές καθιστούν τον Gurobi ιδιαίτερα αποδοτικό σε προβλήματα μεγάλης κλίμακας, όπως το Πρόβλημα Τοποθεσίας Αποθηκών. Ωστόσο, αξίζει να σημειωθεί ότι η χρήση του μέσω Pyomo μπορεί να επιφέρει επιπλέον καθυστερήσεις λόγω των επιβαρύνσεων εκτέλεσης (overhead) της Python, οι οποίες περιορίζουν την επίτευξη της μέγιστης απόδοσης του solver.

B. *Custom Branch & Bound (*Παραδοτέο *3)*

Η υλοποίηση του custom αλγορίθμου Branch & Bound περιλαμβάνει τις εξής τεχνικές:

- Αρχικοποίηση με Ευρετική Μέθοδο: Υπολογισμός ενός αρχικού άνω φράγματος μέσω της ***Fixed Cost Efficiency*** και ***Greedy Assignment***, μειώνοντας την έκταση του δέντρου αναζήτησης.
- Δομή Εξερεύνησης: Χρησιμοποιείται ***LIFO stack*** (αναζήτηση κατά βάθος - DFS) για την αποθήκευση κόμβων προς επέκταση, με την ελπίδα να βρεθεί γρήγορα κάποια καλή λύση σε μικρά και μεσαία προβλήματα. Για μεγάλα και πολύ μεγάλα προβλήματα, η στοίβα χρησιμοποιείται ως FIFO (αναζήτηση κατά πλάτος - BFS) μέχρι να βρεθεί μια ικανοποιητική λύση.
- Relaxed Model: Λύση του LP χαλαρωμένου προβλήματος μέσω του Gurobi για την εκτίμηση lower bound και την αποκοπή μη εφικτών κόμβων.
- Διακλάδωση με βάση τον cost-impact: Επιλογή της πιο κοστοβόρας μεταβλητής (από τις πρώτες 300) (είτε ***X[i]*** είτε ***Y[i,j]***) για δημιουργία διακλαδώσεων, βελτιστοποιώντας τη σύγκλιση.
- Δυναμικός Έλεγχος Λύσης**:** Ανίχνευση αν μια λύση είναι ακέραια ή αν απαιτείται περαιτέρω διακλάδωση.
- Περιορισμός Μέγιστου Βάθους**:** Ορισμός ανώτατου επιτρεπτού βάθους διακλάδωσης με κριτήρια την πολυπλοκότητα του προβλήματος και το τρέχον ποσοστό σύγκλισης, αποφεύγοντας εκθετική αύξηση του χώρου αναζήτησης και μειώνοντας τον χρόνο εκτέλεσης. Παράλληλα, αν η σύγκλιση είναι κάτω από ένα όριο, ανάλογο πάλι της πολυπλοκότητας προβλήματος, ο αλγόριθμος δεν τερματίζει, ελπίζοντας πως υπάρχει καλύτερη λύση πιο βαθιά στο δέντρο αναζήτησης.
- Αποκοπή Αναποτελεσματικών Κόμβων**:** Αν το κάτω φράγμα μιας λύσης υπερβαίνει το καλύτερο γνωστό άνω φράγμα, η διαδρομή εγκαταλείπεται.
- Χρονικό Όριο Εκτέλεσης: Ο αλγόριθμος τερματίζει μετά από 5 λεπτά ανεξαρτήτως προόδου, διασφαλίζοντας ότι δεν υπερβαίνει αποδεκτά χρονικά όρια υπολογιστικής ισχύος.
- Μεγαλύτερη ανοχή στη σύγκλιση στα μεγαλύτερα προβλήματα: Αν βρεθεί ακέραια λύση 99.98% κοντά στη λύση μικτών ακεραίων, θεωρείται η καλύτερη και σταματάει η αναζήτηση.

Η συνδυασμένη χρήση αυτών των τεχνικών στοχεύει στην επιτάχυνση της βελτιστοποίησης και στη μείωση του χώρου αναζήτησης.

IV. Υπολογιστική Σύγκριση

A. Πειραματική Διάταξη

- **CPU:** AMD R5 5600H (6 cores-12 threads-3.3GHz)
- **RAM:** 32 GB (3200MHz)
- **OS:** Windows 10
- **Software Versions:** Gurobi 12.0, Python 3.12.7
- Σύνολα Δεδομένων:
  - Μικρά (10 καταστήματα, 5 αποθήκες)
  - Μεσαία (150 καταστήματα, 30 αποθήκες)
  - Μεγάλα (500 καταστήματα, 150 αποθήκες)
  - Πολύ μεγάλα (1000 καταστήματα, 300 αποθήκες)

B. Μετρικές Απόδοσης

Αξιολογούμε:

- Χρόνο εκτέλεσης (δευτερόλεπτα)
- Χάσμα βέλτιστης λύσης (%)
- Αποτελεσματικότητα heuristic

C. Αποτελέσματα χρόνου και χάσματος λύσεων

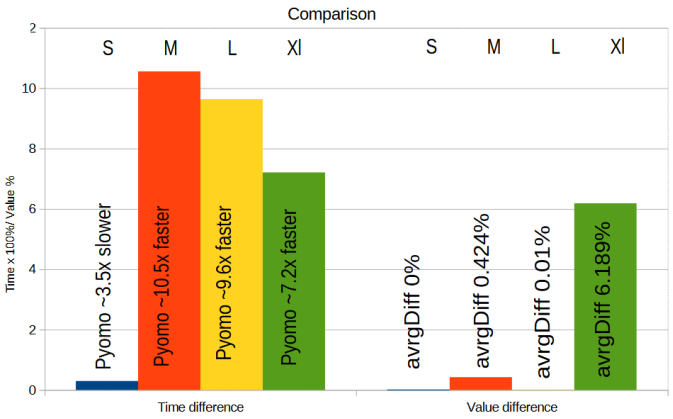

Figure 1. Σύγκριση χρόνου/απόκλιση τιμών

Παρατηρήσεις:

- Ο B&B είναι πιο γρήγορος σε μικρά προβλήματα: Σε μικρά προβλήματα (10s-5w), ο προσαρμοσμένος Branch & Bound (B&B) είναι 3.5 φορές ταχύτερος, καθώς η greedy λύση φαίνεται να οδηγεί γρήγορα στη σύγκλιση καθώς το δέντρο που προκύπτει είναι αρκετά μικρό ακόμη και αν εκτελεστεί πλήρης αναζήτηση δέντρου.
- Ο Gurobi υπερέχει σε μεγαλύτερα προβλήματα: Σε μεσαία προβλήματα (150s-30w), ο Pyomo είναι 10.5 φορές ταχύτερος από τον B&B, ενώ σε μεγάλα προβλήματα (500s-150w) 9.6 φορές ταχύτερος.
- Ο Gurobi διατηρεί πλεονέκτημα σε ακόμη μεγαλύτερες περιπτώσεις: Για ακόμα μεγαλύτερα προβλήματα (1000s-300w), ο Gurobi επιτυγχάνει 7.2 φορές ταχύτερη εκτέλεση,λαμβανοντας υποψην Gurobi επεστρεφε παντα ιδια ή καλύτερη λύση απο το B&B.
- Περιορισμοί του greedy B&B: Καθώς το πρόβλημα μεγαλώνει, η greedy προσέγγιση γίνεται λιγότερο αποτελεσματική, μειώνοντας έτσι το ποσοστό σύγκλισης και, κατ' επέκταση, αυξάνοντας σημαντικά το δέντρο και τον χρόνο αναζήτησης.

Αναλυτικοί χρόνοι:

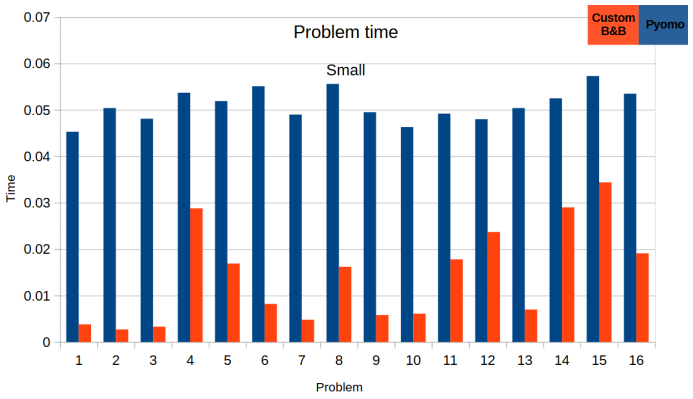

Figure 2. Αποτελέσματα χρόνου (μικρό πρόβλημα)

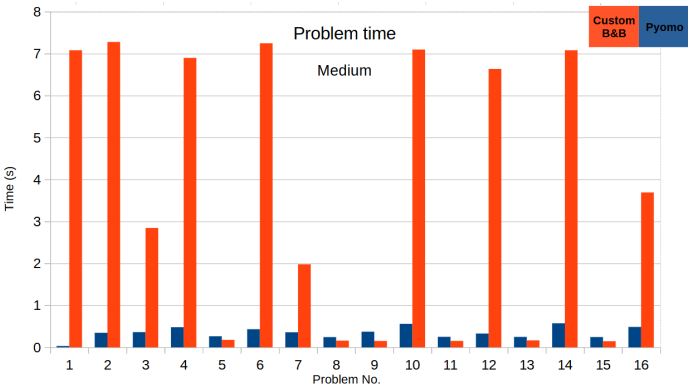

Figure 3. Αποτελέσματα χρόνου (μεσαίο πρόβλημα)

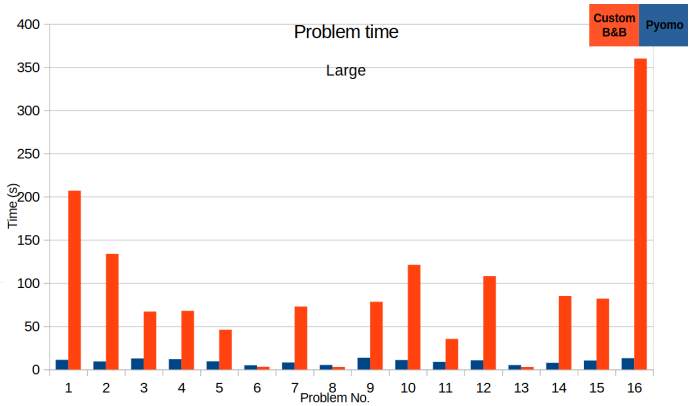

Figure 4. Αποτελέσματα χρόνου (μεγάλο πρόβλημα)

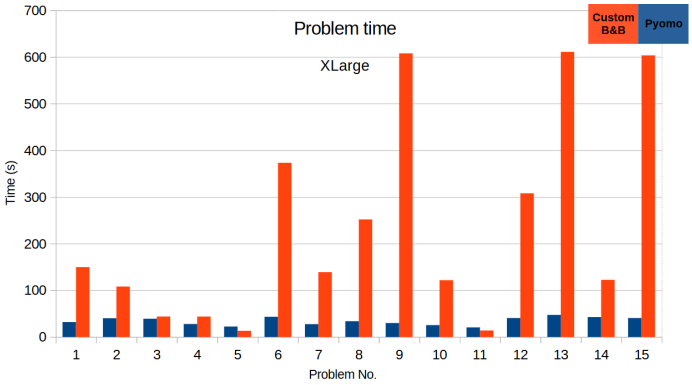

Figure 5. Αποτελέσματα χρόνου (πολύ μεγάλο πρόβλημα)

D. Αποτελεσματικότητα ευρετικών μεθόδων

- **Normal execution:** Ο αλγόριθμος ολοκληρώνει κανονικά την αναζήτηση.
- **Best solution found by greedy:**Η αρχική ευρετική λύση είναι η βέλτιστη.
- **Stopped by max depth:** Το όριο βάθους αποτρέπει την εκθετική αύξηση των κόμβων, αλλά μπορεί να σταματήσει πριν βρεθεί η καλύτερη λύση.
- **Stopped by 99.99%:** Ο αλγόριθμος προσεγγίζει τη βέλτιστη λύση, αλλά το branch-and-bound σταματάει πρωτού να συγκλίνει πλήρως λόγω του μεγάλου χώρου αναζήτησης.
- **Stopped by time:** Ο χρονικός περιορισμός των 5 λεπτών επηρεάζει έντονα την απόδοση, σταματώντας την εκτέλεση πριν την εύρεση μιας πολύ καλής λύσης.

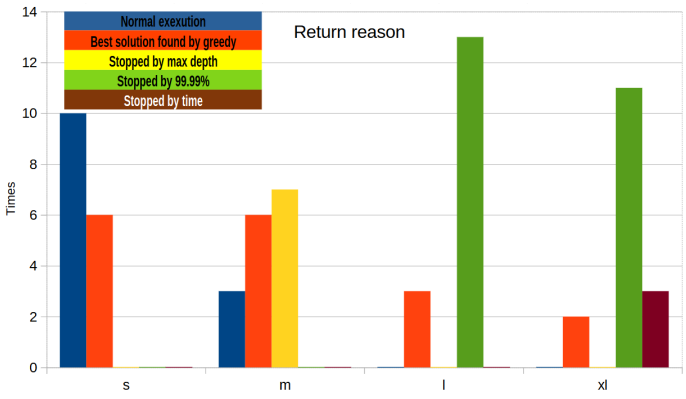

Figure 6. Σύγκριση χρόνου/απόκλισης τιμών

- Μικρά προβλήματα **(s, m):** Οι ευρετικές βοηθούν στην εύρεση καλής λύσης γρήγορα.
- Μεγάλα προβλήματα **(l):** Κυριαρχεί η διακοπή λόγω 99,99% βελτιστότητας, δείχνοντας ότι ο αλγόριθμος πλησιάζει λύση αλλά δεν προλαβαίνει να συγκλίνει.
- Όριο βάθους**:** Περιορίζει την εκθετική αύξηση κόμβων στα μεσαία προβλήματα.
- Όριο χρόνου**:** Στα μεγαλύτερα προβλήματα, ο αλγόριθμος δεν ολοκληρώνει την αναζήτηση εντός 2 λεπτών.

Συμπέρασμα**:** Η πρώτη greedy λύση είναι περισσότερο αποτελεσματική σε μικρά προβλήματα. Όσο μεγαλώνει το πρόβλημα, μειώνονται οι πιθανότητες να βρεθεί καλή λύση με τη greedy προσέγγιση. Επίσης, καθώς μεγαλώνει ο χώρος των λύσεων, χάνουμε βέλτιστες λύσεις λόγω του χρονικού περιορισμού.

E. Πιθανές Βελτιώσεις

- Βελτιώσεις στον **custom B&B:**
  - Αποθήκευση του `parent_node[lower_bound]` για προ-αποκοπή πριν από την εφαρμογή της χαλαρωμένης λύσης.
  - Πέρασμα μεταβλητών και περιορισμών στον Gurobi (μείωση overhead κατά τη διάρκεια της χαλάρωσης).
  - Βελτίωση της συνθήκης τερματισμού χρόνου για αύξηση της σύγκλισης.
- Αύξηση χωρικής πολυπλοκότητας**,** αλλά μείωση χρονικής πολυπλοκότητας**:** Η αύξηση της χωρικής πολυπλοκότητας μπορεί να οδηγήσει σε μειωμένη χρονική πολυπλοκότητα, προσφέροντας καλύτερη απόδοση σε μεγαλύτερα προβλήματα.

Συμπέρασμα ανάλυσης

- Ο προσαρμοσμένος αλγόριθμος B&B είναι αποδοτικός σε μικρά προβλήματα, καθώς εκτελεί πλήρη αναζήτηση δέντρου και βρίσκει γρήγορα τη βέλτιστη λύση, καθώς η ρίζα έχει μικρή ή καθόλου απόκλιση από τη βέλτιστη τιμή.
- Όσο αυξάνεται το μέγεθος του προβλήματος, η greedy προσέγγιση γίνεται λιγότερο αποτελεσματική, οδηγώντας σε αυξημένη απόκλιση από τη βέλτιστη λύση.
- Ο αλγόριθμος συχνά σταματά στο 99.99% συγκλίνουσας λύσης για μεγάλα και πολύ μεγάλα προβλήματα, χωρίς να εξασφαλίζει την απόλυτα βέλτιστη απάντηση.
- Στα πολύ μεγάλα προβλήματα περιορίζεται από το χρονικό όριο εκτέλεσης, παράγοντας λύσεις που μπορεί να μην είναι ικανοποιητικές, αυξάνοντας σημαντικά την τιμή της μέσης απόκλισης.

V. Συμπέρασμα

Η σύγκριση έδειξε ότι:

- Ο Gurobi υπερτερεί σε μεγάλες περιπτώσεις λόγω παραλληλοποίησης και plane cutting.
- Ο προσαρμοσμένος B&B είναι αποδοτικός για μικρότερα προβλήματα.
- Μελλοντική εργασία μπορεί να βελτιώσει τον B&B με καλύτερη στρατηγική κλαδέματος και παράλληλη υλοποίηση.

Βιβλιογραφία

References

[1] Gurobi Optimization, LLC. ”Gurobi Optimizer Reference Manual,” 2023.

[2] D.-Z. Du P.M. Pardalos,X. Hu ,W. Wu, ”Εισαγωγή στη Συνδυαστική Βελτιστοποίηση,”