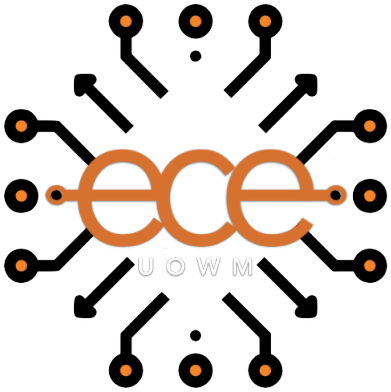


Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

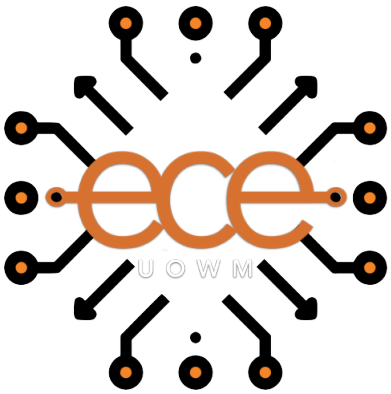
Υλοποίηση και Υπολογιστική Σύγκριση Αλγορίθμων για Προβλήματα Βέλτιστης Κοπής

Δημήτριος Κρύος (ΑΜ: 1285)

Επιβλέπων Καθηγητής: Νικόλαος Πλόσκας

Εργαστήριο Ευφυών Συστημάτων & Βελτιστοποίησης

24 Οκτωβρίου 2022



University of Western Macedonia Μακεδονίας

Department of Electrical & Computer Engineering

Development and Computational Comparison of Algorithms for the Solution of the Cutting Stock Problem

Dimitris Kryos (AM: 1285)

Supervising Professor: Nikolaos Ploskas

Intelligent Systems & Optimization Laboratory

24 October 2022

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η διερεύνηση και η μελέτη του προβλήματος βέλτιστης κοπής (cutting stock problem), πως επιδρά το πρόβλημα αυτό σε βιομηχανικές επιχειρήσεις παραγωγής υλικών αλλά και πως μπορεί να επιλυθεί. Το πρόβλημα υπάρχει εδώ και πολλά χρόνια, με αποτέλεσμα να υπάρχει ένα μεγάλος αριθμός εργασιών για την επίλυσή του. Το πρόβλημα αυτό υπάρχει σε πολλές μορφές διαστάσεων, ωστόσο στην παρούσα εργασία ασχολούμαστε με το μονοδιάστατο πρόβλημα βέλτιστης κοπής. Είναι ένα γραμμικό πρόβλημα συνδυαστικής βελτιστοποίησης, δηλαδή μοντελοποιείται ως πρόβλημα γραμμικού προγραμματισμού. Υπάρχουν πολλές εργασίες που χρησιμοποιούν ακριβείς και ευρετικούς αλγορίθμους. Σε αυτήν την εργασία παρουσιάζονται δύο αλγόριθμοι που χρησιμοποιούνται για την αντιμετώπιση του προβλήματος. Ο πρώτος ασχολείται με την επίτευξη της γραμμικής χαλάρωσης του προβλήματος και ο δεύτερος είναι ένας ακριβής αλγόριθμος. Ωστόσο, και οι δύο αλγόριθμοι συνδυάζουν μεθόδους και τεχνικές για να επιλύσουν το πρόβλημα βέλτιστης κοπής. Στόχος της εργασίας μας είναι η ανάλυση των αποτελεσμάτων της επίτευξης του προβλήματος και η σύγκριση μεταξύ των μεθόδων που παρουσιάζονται ως προς τον χρόνο και τη λύση για να διακρίνουμε την αποτελεσματικότερη και πιο αποδοτική μέθοδο.

Λέξεις κλειδιά: Πρόβλημα βέλτιστης κοπής, Γραμμικός προγραμματισμός, Ευρετικοί αλγόριθμοι, Ακριβείς αλγόριθμοι

Abstract

The purpose of this thesis is to investigate and study the optimal cutting stock problem, how this problem affects companies producing materials and how it can be solved. The problem has existed for many years, as a result of which there is a large number of works with algorithms to solve it. This problem exists in many dimensional forms, however in the present paper we deal with the one-dimensional cutting stock problem. This problem is a linear combinatorial optimization problem, i.e., it is modeled as a linear programming problem, where as mentioned there are many works that use heuristic and exact algorithms to solve it. In this thesis, two algorithms are presented that are used to deal with the problem. The first deals with achieving the linear relaxation of the problem and the second is an exact algorithm. However, both algorithms combine methods and techniques to solve the optimal cutting stock problem. The aim of our work is to analyze the results of achieving the problem and compare between the presented methods in terms of time and output to distinguish the most efficient method.

Keywords: Cutting stock problem, Linear programming, Heuristic algorithms, Exact algorithms

Δήλωση Πνευματικών Δικαιωμάτων

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 ''Υλοποίηση και Υπολογιστική Σύγκριση Αλγορίθμων για Προβλήματα Βέλτιστης Κοπής'' 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. Νικόλαο 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 βέλτιστης κοπής είναι ένα πρόβλημα που ανήκει στην οικογένεια προβλημάτων κοπής (cutting problems) [2] όπου συναντάται σε πολυάριθμες πραγματικές εφαρμογές από την επιστήμη των υπολογιστών, την κατασκευή, τη διαδικασία παραγωγής, κ.λπ.[3]. Το πρόβλημα βέλτιστης κοπής αρχικά διατυπώθηκε για πρώτη φορά το 1939 από τον Ρώσο οικονομολόγο Leonid Kantorovich [4], συγγραφέα του έργου "Μαθηματικές Μέθοδοι Οργάνωσης και Προγραμματισμού Παραγωγής", και μελετήθηκε σε διάφορες διαστάσεις για την αντιμετώπιση εφαρμογών του πραγματικού κόσμου σε διάφορες βιομηχανίες [5].

Η διασημότητα του προβλήματος προκύπτει από το γεγονός ότι ήταν η πρώτη εφαρμογή προβλήματος του γραμμικού προγραμματισμού (linear programming) [6]. Το πρόβλημα συνεπώς, είναι ένα πρόβλημα βελτιστοποίησης που ανήκει στην κατηγορία του γραμμικού προγραμματισμού, δηλαδή μοντελοποιείται ως πρόβλημα γραμμικού προγραμματισμού [7][8]. Είναι ένα NP-Complete πρόβλημα που έχει πολλές εφαρμογές στον χώρο των βιομηχανιών παραγωγής (ράβδων, επίπλων, ρολών χαρτιού, μετάλλων, γυαλιού, κ.λπ.), όπου απαιτούνται διαφορετικά σχέδια κοπής από τις παραγγελίες που δίνουν οι πελάτες και το πρόβλημα είναι η τοποθέτηση διαφόρων σχεδίων που απαιτούνται στην πρώτη ύλη που έχει η επιχείρηση στην κατοχή της. Με αυτόν τον τρόπο προσπαθούν να πετύχουν την ελαχιστοποίηση της φύρας (αποβλήτων ή απώλειας κοπής), επιτυγχάνοντας έτσι τη μείωση της παραγωγής των υλικών, συνεπώς τη μείωση του κόστους παραγωγής και τη μείωση του χρόνου παραγωγής [7][8].

Ωστόσο, λίγο αργότερα της διατύπωσης του προβλήματος από τον Leonid Kantorovich [4], οι Gilmore και Gomory[9] ήρθαν για να επιτύχουν μια σημαντική πρόοδο στην επίλυση προβλημάτων κοπής χρησιμοποιώντας τον γραμμικό προγραμματισμό στην τεχνική της καθυστερημένης δημιουργίας προτύπων (στηλών) για την ελαχιστοποίηση της απώλειας περικοπής υλικού και να επιλύσουν το μονοδιάστατο πρόβλημα [9][10]. Αυτό είχε ως αποτέλεσμα να ασχοληθεί ένας μεγάλος αριθμός ερευνητών με το πρόβλημα βέλτιστης κοπής, καθώς το οικονομικό κίνητρο ήταν πολύ μεγάλο για την εύρεση αποδοτικότερων λύσεων για τις βιομηχανίες [11]. Έτσι, το 1990 προτάθηκαν αλγόριθμοι για την επίλυσή του προβλήματος βέλτιστης κοπής όπου συνδύαζαν τη δημιουργία στήλης, το όριο και τη διακλάδωση [8].

Ο Dyckhoff [2] ανέπτυξε ένα δικό του σύστημα ταξινόμησης για τα προβλήματα βέλτιστης κοπής και συσκευασίας κάδου (build packing problems) πρόβλημα παρόμοιο με το πρόβλημα βέλτιστης κοπής, όπου τα ταξινομεί ως 1/V/I/R, όπου το 1 είναι η διάσταση του προβλήματος, το V είναι τα διαφορετικά μεγάλα αντικείμενα, το I είναι τα πολλά πανομοιότυπα μεγάλα αντικείμενα και το R πολλά αντικείμενα με σχετικά μικρές διαστάσεις [11]. Ωστόσο, οι Wäscher et al. [12] έδειξαν ότι όλα σχεδόν τα προβλήματα βέλτιστης κοπής μπορούν να ταξινομηθούν σε προβλήματα ελαχιστοποίησης εισόδου, όπου θα χρησιμοποιούνται λιγότεροι όσο γίνεται πόροι, και σε προβλήματα όπου δίνονται πόροι αλλά αυτή τη φορά μεγιστοποιούν την παραγωγή [12].

1.2 Κίνητρα και στόχοι υλοποίησης

Ως κίνητρο για τη συγγραφή της παρούσας διπλωματικής εργασίας είναι το ίδιο το πρόβλημα βέλτιστης κοπής, όπου η παρουσία του είναι συνεχής εμπόδιο για τις επιχειρήσεις μέχρι και σήμερα, αφού τα απόβλητα των επιχειρήσεων συνεχίζουν να είναι μεγάλα σε αριθμό παρά την ελαχιστοποίηση της φύρας. Για παράδειγμα, μία μεγάλη βιομηχανία στην Βραζιλία ξόδευε περίπου ένα εκατομμύριο δολάρια ετησίως για τα απόβλητα έπειτα από τη διαδικασία κοπής τους [13]. Το ερώτημα είναι αν υπάρχουν πιο αποτελεσματικοί τρόποι για τη μείωση των αποβλήτων, ποιοι είναι αυτοί και ποιος από αυτούς είναι πιο αποτελεσματικός και συνεπώς πιο αποδοτικός για τις επιχειρήσεις.

Ως στόχος της παρούσας διπλωματικής εργασίας είναι ο τρόπος που μπορούμε να επιλύσουμε το πρόβλημα βέλτιστης κοπής με αλγορίθμους που συνδυάζουν διάφορες τεχνικές για να επιτύχουν τη λύση του, δηλαδή τη μείωση της φύρας με τον βέλτιστο τρόπο. Ουσιαστικά, στόχος είναι η δημιουργία αποτελεσματικών σχεδιασμών παραγωγής μηχανών κοπής πάνω σε πραγματικά προβλήματα, δηλαδή με μικρά, μεσαία και μεγάλα μήκη υλικών που πρόκειται να παραχθούν. Θα μελετήσουμε λοιπόν δύο κατηγορίες αλγορίθμων που επιλύουν το πρόβλημα βέλτιστης κοπής χρησιμοποιώντας μια σειρά μοντελοποίησης διαφόρων φάσεων για τη λύση υποπροβλημάτων, δυναμικό προγραμματισμό, μαθηματικά μοντέλα, άτυπες απαριθμήσεις και μεθόδους σε συνδυασμό για την επίλυση των προβλημάτων [13]. Η πρώτη κατηγορία αλγορίθμων αφορά τους ευρετικούς αλγόριθμους, όπου χρησιμοποιούν διάφορες τεχνικές για την επίλυση του προβλήματος και η δεύτερη κατηγορία αλγορίθμων είναι οι ακριβείς αλγόριθμοι, όπου συνδυάζουν άλλες τεχνικές από αυτές των ευρετικών. Τέλος, έχουμε ως στόχο τη σύγκριση μεθόδων του ακριβή αλγορίθμου για να δούμε ποια μέθοδος που χρησιμοποιεί παράγει αν όχι τις βέλτιστες τότε τις καλύτερες λύσεις ως προς την παραγωγή ράβδων στοκ σταθερού μήκους και τον χρόνο επίλυσης κατά τη διαδικασία επίλυσης του προβλήματος για την καλύτερη αντιμετώπιση του από τις επιχειρήσεις.

1.3 Διάρθρωση κειμένου

Σε αυτήν την εργασία θα εστιάσουμε πάνω σε δεδομένου μήκους μονοδιάστατα προβλήματα βέλτιστης κοπής (Standard 1D-cutting stock problems), δηλαδή σε προβλήματα όπου το στοκ της επιχείρησης που έχει στο απόθεμα της είναι ενός μεγέθους όπως αναφέρθηκε. Στο κεφάλαιο 2 θα γίνει αναφορά στον γραμμικό προγραμματισμό και θα ακολουθήσει αναφορά σε μοντέλα και τεχνικές επίλυσης γραμμικού προγραμματισμού. Ακόμη, θα ακολουθήσουν αναφορές από διάφορες βιβλιογραφικές πηγές που επιλύουν το πρόβλημα βέλτιστης κοπής με ευρετικούς και ακριβείς αλγόριθμους. Ωστόσο θα γίνει και αναφορά σε τεχνικές επίλυσης που επιλύονται με γραμμικό προγραμματισμό, όπως η αποσύνθεση των Dantzig-Wolfe, η τεχνική δημιουργίας στήλης (column generation), το πρόβλημα του σακιδίου (knapsack), τεχνικές διακλάδωσης και δέσμευσης (branch and bound), διακλάδωσης και κοπής (branch and cut) και άλλες τεχνικές που χρησιμοποιούνται.

Στο κεφάλαιο 3 θα γίνει η ανάλυση των κωδίκων που χρησιμοποιήσαμε, όπως ο κώδικας επίλυσης της γραμμικής χαλάρωσης (linear relaxation) και του ακριβούς αλγορίθμου που χρησιμοποιήθηκαν για την επίλυση του προβλήματος βέλτιστης κοπής. Ακόμη θα γίνει αναφορά σε μεθόδους αναζήτησης που χρησιμοποιούνται για την επίλυση του προβλήματος. Στο κεφάλαιο 4 θα ακολουθήσει η σύγκριση μεταξύ των αλγορίθμων με την εκτέλεση τεσσάρων διαφορετικών μεθόδων αναζητήσεων που χρησιμοποιήθηκαν για την επίλυση των προβλημάτων με βάση τα αποτελέσματα που παρήγαγαν. Ακόμη τα αποτελέσματα αυτά θα φανούν σε πίνακες που δημιουργήθηκαν κατά την επίλυσή τους. Τέλος, στο κεφάλαιο 5 παρατίθενται τα συμπεράσματα στα οποία καταλήξαμε από τη σύγκριση αλλά θα γίνει και η αναφορά σε μελλοντικούς τρόπους που μπορούν να επιλύσουν το πρόβλημα βέλτιστης κοπής στηριζόμενα στην παρούσα εργασία.

Κεφάλαιο 2

Μοντέλα και τεχνικές επίλυσης γραμμικού προγραμματισμού

Σε αυτό το κεφάλαιο θα εστιάσουμε στον γραμμικό προγραμματισμό, στη γραμμική χαλάρωση, σε γραμμικά μοντέλα και σε τεχνικές που επιλύουν το πρόβλημα της ελαχιστοποίησης της φύρας στο μονοδιάστατο πρόβλημα. Ακόμη, θα ακολουθήσει η παρουσίαση εργασιών σχετικά με ευρετικούς και ακριβείς αλγόριθμους που χρησιμοποιήθηκαν για την επίλυση του προβλήματος.

Όπως αναφερθήκαμε στην εισαγωγή, το πρόβλημα βέλτιστης κοπής μπορεί να μοντελοποιηθεί ως ένα πρόβλημα γραμμικού προγραμματισμού. Ο γραμμικός προγραμματισμός ασχολείται με την ελαχιστοποίηση ή τη μεγιστοποίηση μιας γραμμικής συνάρτησης ενός προβλήματος υπό κάποιους γραμμικούς περιορισμούς [14]. Για τα προβλήματα βέλτιστης κοπής ουσιαστικά, χρησιμοποιήθηκε διότι συνήθως είχε πολύ καλές ακέραιες λύσεις από τη λύση της γραμμικής χαλάρωσης (linear relaxation) σε προβλήματα όπου η μέση ζήτηση των μηκών των παραγγελιών από τους πελάτες είναι μεγάλη [15] Συνεπώς, είναι ένα ισχυρό εργαλείο που μπορεί να αντιμετωπίσει διάφορα προβλήματα, όπως συνδυαστικής βελτιστοποίησης, ανάθεσης κ.α. [15] [14].

Στη συνέχεια, θα γίνει αναφορά σε μοντέλα που ελαχιστοποιούν την απώλεια κοπής του προβλήματος βέλτιστης κοπής και σε άλλες τεχνικές που χρησιμοποιούν γραμμικό προγραμματισμό για την επίλυσή του.

2.1 Μοντέλο Kantorovich (Kantorovich model)

Για την ελαχιστοποίηση του αριθμού των υλικών που χρησιμοποιούνται για την παραγωγή όλων των σχεδίων για το πρόβλημα βέλτιστης κοπής, ο Leonid Kantorovich[4] δημιούργησε ένα μοντέλο μαθηματικού προγραμματισμού[15] για το πρόβλημα βέλτιστης κοπής. Ουσιαστικά, πρόκειται για το πρώτο μοντέλο ακέραιου γραμμικού προγραμματισμού (integer linear programming problem) που έχει ως βάση την εκχώρηση μεταβλητών [16]. Ακολουθεί η διατύπωση παρακάτω.

$$
\begin{aligned}
\text{min} \quad & \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{y_k} \\
\textbf{Subject to} \quad & \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{x_{ik}} \geq \mathbf{b_i}, \quad \mathbf{i = 1, ..., m}, \\
& \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{w_i x_{ik}} \leq \mathbf{W y_k}, \quad \mathbf{k = 1, ..., K}, \\
& \mathbf{y_k = 0 \quad or \quad 1}, \quad \mathbf{k = 1, ..., K} \\
& \mathbf{x_{ik} \geq 0 \quad and \quad integer}, \qquad \mathbf{i = 1, ..., m}, \quad \mathbf{k = 1, ..., K},
\end{aligned}
$$

Όπου,

- Το K είναι ένα γνωστό άνω όριο (upper bound) των ράβδων που χρειάζονται σε μια βέλτιστη λύση.
- Το $\mathbf{y_k}$ είναι για να καταλαβαίνουμε αν χρησιμοποιείται η ράβδος ή όχι. Ουσιαστικά πρόκειται για μία μεταβλητή, όπου:
  - αν $\mathbf{y_k = 0}$ τότε η ράβδος δε χρησιμοποιείται,
  - αν $\mathbf{y_k = 1}$ τότε η ράβδος χρησιμοποιείται.
- Το $\mathbf{x_{ik}}$ είναι το το πλήθος ή ο αριθμός που μας δείχνει πόσες φόρες κόβεται το στοχείο i από το ρολό k.
- Το $\mathbf{b_i}$ είναι η ζήτηση των μηκών παραγγελίας.

Ουσιαστικά ο πρώτος περιορισμός αφορά τη ζήτηση $\mathbf{b_i}$, ενώ ο δεύτερος πρόκειται για τον περιορισμό του σακιδίου (knapsack). Ωστόσο, με την χαλάρωση των

δύο τελευταίων περιορισμών σε $\mathbf{0} \leq \mathbf{y_k}$ και $\mathbf{x_{ik}} \geq \mathbf{0}$ μπορεί να προκύψει από το πρόβλημα χαλάρωσης του γραμμικού προβλήματος, ένα κατώτερο όριο (lower bound) για το βέλτιστο [15]. Ακόμη, όταν οι τιμές μπορεί να είναι πραγματικές η χαλάρωση του μοντέλου του Kantorovich[4] είναι πολύ αδύναμη [17]. Ωστόσο, σύμφωνα με τους Martello και Toth [18], αποδείχθηκε ότι το όριο που παράγει η χαλάρωση του γραμμικού προβλήματος μπορεί να είναι πολύ αδύναμο, και ότι το μεγαλύτερο μειονέκτημα του είναι ότι λόγω του αδύναμου κάτω ορίου που έχει παράγονται περισσότερα απόβλητα [16].

Θα αναφερθούμε ξανά στο μοντέλο του Kantorovich[4] στο κεφάλαιο 3 της ανάλυσης κωδίκων και συγκεκριμένα στην υποενότητα 3.2 καθώς η επίλυση του αλγορίθμου που θα χρησιμοποιήσουμε είναι βασισμένη στο μοντέλο αυτό για την ελαχιστοποίηση της φύρας.

2.2 Μοντέλο Gilmore και Gomory (Gilmore and Gomory model)

Σε αυτήν την υποενότητα θα αναφερθούμε στην προσέγγιση επίλυσης του προβλήματος βέλτιστης κοπής από τους Gilmore και Gomory[9], παρουσιάζοντας το μοντέλο που πρότειναν για την αντιμετώπιση του προβλήματος. Αρχικά, ας υποθέσουμε ότι έχουμε ένα μονοδιάστατο πρόβλημα αποθέματος κοπής σε μία βιομηχανία παραγωγής μετάλλων όπου το μέγεθος των στοκ της επιχείρησης που έχει στο απόθεμά της είναι σταθερό μήκους L και έχουμε παραγγελία από πελάτη για μήκη τύπου $\mathbf{l_i}$, όπου $i = 1, 2, \cdots, m$ και η ζήτηση για το κάθε κομμάτι είναι τύπου $\mathbf{d_i}$. Ακόμη υπάρχει και το $\mathbf{a_{ij}}$, που είναι το πλήθος των i τεμαχίων σε συνδυασμό σχεδίου j, όπου για κάθε συνδυασμό προκύπτει και ένα κόστος. Για κάθε διαφορετικό συνδυασμό, το κάθε κόστος μπορεί να αντιπροσωπεύεται ως το κόστος των αποβλήτων ή το κόστος της ύλης που χρησιμοποιείται. Για την επίλυση του λοιπόν, πρέπει να βρεθεί ένα πλήθος συνδυασμών $\mathbf{x_i}$ ώστε να ελαχιστοποιήσουμε το συνολικό κόστος. Έτσι λοιπόν το παραπάνω πρόβλημα μπορούμε να το παρουσιάσουμε ως μαθηματική διατύπωση της μορφής όπως βλέπουμε παρακάτω.

$$
\begin{aligned}
&\min && \mathbf{Cost} = \mathbf{c}'\mathbf{x} \\
&\textbf{Subject to} && \mathbf{Ax} \leq \mathbf{d}, \\
& && \mathbf{x} \geq \mathbf{0}, \quad x \quad integer
\end{aligned}
$$

Για κάθε πρόβλημα βέλτιστης κοπής του πραγματικού κόσμου που έχει τη μορφή του παραπάνω μαθηματικού μοντέλου, θα ήταν ανέφικτη η δημιουργία όλων των πιθανών συνδυασμών την ίδια στιγμή. Οι Gilmore και Gomory [9][10] προσέγγισαν την επίλυση του προβλήματος χρησιμοποιώντας την καθυστερημένη δημιουργία στήλης (delayed column generation). Ουσιαστικά η τεχνική δημιουργίας στήλης όπως ονομάζεται και αλλιώς, είναι μία τεχνική που έχει προκαλέσει πολύ μεγάλο ενδιαφέρον σε προβλήματα με πολύ μεγάλο αριθμό μεταβλητών καθώς χρειάζεται μόνο ένα κλάσμα των μεταβλητών για να είναι βέλτιστη, διότι οι περισσότερες μεταβλητές θα λαμβάνουν τιμή μηδέν και άρα θα έχει ως αποτέλεσμα ότι η λύση μπορεί να βρεθεί χωρίς να προστεθούν παραπάνω μεταβλητές [19]. Η τεχνική δημιουργίας στήλης, είναι ένα πολύ σημαντικό και απαραίτητο εργαλείο που βοηθάει στην επίλυση

ενός μαθηματικού προγράμματος όπως το πρόβλημα βέλτιστης κοπής, προσθέτοντας επαναληπτικά μεταβλητές του μοντέλου για την υπολογιστική βελτιστοποίηση του στόχου [20]. Αν λοιπόν, καταστεί δυνατό κατά την εισαγωγή μεταβλητών, οι μεταβλητές αυτές δε βελτιώνουν τον στόχο τότε η τιμή της αντικειμενικής συνάρτησης δε θα βελτιωθεί άρα η διαδικασία θα σταματήσει [20]. Μπορούμε λοιπόν να χωρίσουμε την τεχνική αυτή σε τρία στάδια [21].

1. Το πρώτο στάδιο έχει να κάνει με τη δημιουργία ενός συνόλου συνδυασμών m, όπου για κάθε έναν συνδυασμό θα υπάρχει ένα μόνο κομμάτι στοκ.

2. Το δεύτερο στάδιο έχει να κάνει με την επίλυση του πρώτου περιορισμού του μαθηματικού μοντέλου παραπάνω, αγνοώντας τους ακέραιους περιορισμούς.

3. Το τρίτο στάδιο έχει να κάνει με τη χρησιμοποίηση των σκιώδη τιμών από την επίλυση του προβλήματος γραμμικού προγραμματισμού για τη δημιουργία νέου εφικτού συνδυασμού που όταν προστεθεί στο γραμμικό πρόγραμμα θα βελτιώσει την αντικειμενική συνάρτηση. Αν δε βελτιωθεί η αντικειμενική συνάρτηση, δηλαδή δε βρεθεί ένας συνδυασμός για να τη βελτιστοποιήσει τότε ολοκληρώνεται η λύση και έπειτα τερματίζει, διαφορετικά θα επιστρέψει στο δεύτερο στάδιο.

Ωστόσο, για τη δημιουργία συνδυασμού μιας εγγυημένης βελτίωσης για την αντικειμενική συνάρτηση οι Gilmore και Gomory[10] πρότειναν τη λύση του προβλήματος του σακιδίου (knapsack). Το πρόβλημα αυτό χαρακτηρίζεται ως ένα ακέραιο προγραμματιστικό πρόβλημα όπου η πραγματική μορφή του είναι η ακόλουθη.

$$\begin{aligned} &\max && \mathbf{cx} \\ &\textbf{Subject to} && \mathbf{ax} \leq \mathbf{b}, \\ &&& \mathbf{x} \geq \mathbf{0}, \quad \textbf{integer} \end{aligned}$$

Όπου,

- Το **c** και το **a** είναι n-διανύσματα όπου αποτελούνται από ακέραιους θετικούς.
- Το **b** είναι ένας θετικός ακέραιος αριθμός, ωστόσο υπάρχει και το κυρτό πολύεδρο των λύσεων που είναι εφικτοί για το πρόβλημα βέλτιστης κοπής στην προκειμένη περίπτωση (δηλαδή το x που ανήκει στο πρόβλημα βέλτιστης κοπής) που είναι γνωστό ως πολύεδρο του σακιδίου (knapsack polyhedron) [22].

Ουσιαστικά, στην επίλυση του προβλήματος βέλτιστης κοπής από τους Gilmore και Gomory[9], όπως παρατηρήθηκε το πρόβλημα του σακιδίου προκύπτει από το γεγονός ότι υπάρχει η τεχνική δημιουργίας στήλης. Έτσι για το παράδειγμα των Gilmore και Gomory[9] το πρόβλημα του σακιδίου έχει την εξής μορφή.

$$\begin{aligned} &\max && \mathbf{V} = \mathbf{s}'\mathbf{a} \\ &\textbf{Subject to} && \mathbf{I}'\mathbf{a} \leq \mathbf{L} \\ &&& \mathbf{a} \geq \mathbf{0}, \quad \mathbf{a} \quad \textbf{integer} \end{aligned}$$

Όπου,

- Το $\mathbf{s_i}$ είναι η σκιώδης τιμή του γραμμικού προβλήματος του για κάθε διαφορετικό i.
- Το $\mathbf{a_i}$ είναι το πλήθος του κάθε κομματιού i που υπάρχει στον συνδυασμό.
- Το $\mathbf{I_i}$ είναι το μήκος του κάθε τεμαχίου i.
- Το **L** ειναι το στοκ σταθερού μήκους που έχει η επιχείρηση στο απόθεμά της.

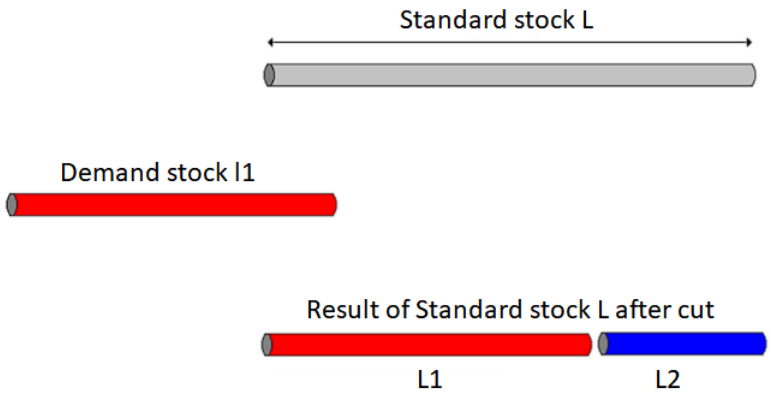
Το πρόβλημα του σακιδίου είναι ένα πρόβλημα όπου έχει πολλές εφαρμογές στον τομέα των βιομηχανιών όπως για παράδειγμα σε ελέγχους προϋπολογισμού μιας επιχείρησης, στο φόρτωμα φορτίων, στο μονοδιάστατο πρόβλημα όπως είδαμε και

σε άλλες εφαρμογές [23]. Ωστόσο οι Lorie και Savage [24] έχουν ασχοληθεί με αυτό το πρόβλημα καθώς έχουν σχολιάσει εφαρμογές που εμπεριέχεται το πρόβλημα του σακιδίου. Ακόμη, υπάρχουν πολλοί μέθοδοι επίλυσης του προβλήματος του σακιδίου όμως η πιο γνωστή είναι αυτή του δυναμικού προγραμματισμού, όπου από τις πρώτες εργασίες της σχέσης του δυναμικού προγραμματισμού και του προβλήματος του σακιδίου είναι αυτές των Bellman [25][26][27][28] και Dantzig [29][23]. Οι Gilmore και Gomory[7][10] έχουν συνεισφέρει μαζί με άλλους επιστήμονες όπως ο Greenberg[30], Yormark [31], Dreyfus και Prather [32] για την ανάπτυξη του δυναμικού προγραμματισμού σε μεθόδους, όπου μία από αυτές είναι η ανάπτυξη αποτελεσματικών ορίων [23]. Υπάρχουν και άλλες τεχνικές που επιλύουν το πρόβλημα του σακιδίου, όπως είναι οι μέθοδοι ακέραιου γραμμικού προγραμματισμού που περιέχονται οι αλγόριθμοι διακλάδωσης και δέσμευσης (branch and bound) και ευρετικοί αλγόριθμοι [23]. Κάποιες από αυτές της μεθόδους θα αναλυθούν σε επόμενα κεφάλαια. Συνοπτικά λοιπόν, σε αυτήν την υποενότητα είδαμε το μοντέλο των Gilmore και Gomory [9] [10], όπου κατάφεραν να λύσουν το πρόβλημα της φύρας (ή απώλεια κοπής) διατυπώνοντας το πρόβλημα ως γραμμικό πρόβλημα και λόγω του μεγάλου αριθμού μοτίβων χρησιμοποίησαν μια παρόμοια τεχνική με αυτή της αποσύνθεσης των Dantzig-Wolfe[15] που θα δούμε αργότερα, δηλαδή της δημιουργίας στηλών για να επιλύσουν το μονοδιάστατο πρόβλημα. Τέλος, θα αναφερθούμε ξανά στην τεχνική δημιουργίας στήλης και στο πρόβλημα του σακιδίου στο κεφάλαιο της ανάλυσης των κωδίκων καθώς θα χρησιμοποιηθούν για την επίλυση του προβλήματος.

2.3 Μοντέλα μίας κοπής (One cut models)

Σε αυτήν την υποενότητα θα γίνει αναφορά στα μοντέλα μίας κοπής και στον τρόπο λειτουργίας τους. Για τα μοντέλα μίας κοπής χρησιμοποιούνται μεταβλητές απόφασης όπου η κάθε μία μεταβλητή χρησιμοποιείται σε μία λειτουργία κοπής ενός μικρού κομματιού σε αντίθεση με το μοντέλο των Gilmore και Gomory[10] που είδαμε, όπου η κάθε μια μεταβλητή χρησιμοποιείται σε μεγάλο πλήθος αριθμού λειτουργίας κοπής για μεγάλα κομμάτια αντικειμένου για να παραχθούν μικρότερα. Στα μοντέλα μίας κοπής αν έχουμε ένα κομμάτι υλικού δεδομένου μήκους, το κομμάτι αυτό θα χωριστεί σε δύο μικρότερα τμήματα κομματιών, όπου για κάθε κοπή θα παραχθεί ένα κομμάτι υλικού μήκους από την παραγγελία του πελάτη που προέρχεται, είτε από τη διαδικασία κοπής ενός στοκ κομματιού, είτε από κομμάτι που έχει ήδη πραγματοποιηθεί μία προηγούμενη κοπή και έχει περισσέψει [15]. Στο Σχήμα 2.1 βλέπουμε τη διαδικασία τομής, δηλαδή την κοπή σε δύο τμήματα του δεδομένου μήκους στοκ που χρησιμοποιείται όπου επιβάλλεται το ένα από τα δύο τμήματα που κόπηκαν να εξυπηρετούν μία παραγγελία μήκους του πελάτη.

Σχήμα 2.1: Μοντέλο μίας κοπής (One-Cut Model)



Βλέπουμε ότι:

- Το standard stock **L** είναι το στοκ σταθερού μήκους που έχει η επιχείρηση στο απόθεμά της.

- Το demand stock **l1** είναι το απαιτούμενο μήκος στοκ μιας παραγγελίας.

Παρατηρούμε λοιπόν, ότι το στοκ σταθερού μεγέθους θα κοπεί σε δύο τμήματα, όπως προαναφέρθηκε, προκειμένου να καλύψει την παραγγελία του απαιτούμενου στοκ. Έτσι από το αποτέλεσμα που προκύπτει το τμήμα **L1** θα παραχθεί για την ικανοποίηση της παραγγελίας και το τμήμα **L2** θα κρατηθεί για να χρησιμοποιηθεί σε επόμενο απαιτούμενο στοκ αν χωράει.

Τα μοντέλα μίας κοπής έχουν ισοδύναμες ακέραιες εφικτές λύσεις με τις λύσεις που πρότειναν οι Gilmore και Gomory[9] στο δικό τους μοντέλο όπως έδειξε ο Dyckhoff[2][33]. Ακόμη, μπορεί να μην επιτύχουν μία ακριβή λύση ως προς την ακεραιότητα του γραμμικού προβλήματος λόγω του ότι ο χώρος λύσης δεν είναι συμμετρικός και ας μην έχει αύξηση ο αριθμός των μεταβλητών που είναι και ψευδοπολυωνυμικός $\mathbf{O(mW_{max})}$. Ωστόσο, σύμφωνα με τον Johnson [34], τα μοντέλα μίας κοπής είναι μεγαλύτερα ως προς τη συμμετρία σε σχέση με το μοντέλο των Gilmore και Gomory[9][10] [15]. Δύο πολύ γνωστές εκδοχές των μοντέλων μίας κοπής είναι αυτή του μοντέλου του Dyckhoff (Dyckhoff model) [2] και του μοντέλου Stadtler (Stadtler model), όπου ο Stadtler [35] στο μοντέλο του ακολουθεί τη διαδικασία του Dyckhoff[2] [33] προτείνοντας μια επέκταση μιας δομής του προβλήματος βέλτιστης κοπής.

Ακόμη, υπάρχουν και άλλα μοντέλα που ελαχιστοποιούν την απώλεια κοπής όπως είναι το πλήρες μοντέλο (complete-cut)[35][35], το μοντέλο με ευρετήριο θέσης (position-indexed models), το μοντέλο κυρτότητας, μίας κοπής μοντέλο, όπως το one-cut LP-model του Rao[36] και Dyckhoff [33][37], το εκτεταμένο μοντέλο (Extended model), κ.α. [15].

2.4 Αποσύνθεση των Dantzig-Wolfe Decomposition

Η τεχνική αποσύνθεσης Dantzig-Wolfe που θα δούμε σε αυτήν την υποενότητα είναι μία τεχνική που μπορεί να χρησιμοποιηθεί για να παρέχει μια αποτελεσματική λύση με ισχυρές χαλαρώσεις (linear relaxations) με τη λήψη μοντέλων για ακέραια και συνδυαστικά προβλήματα βελτιστοποίησης [15]. Αυτή η τεχνική ουσιαστικά είναι κατάλληλη να χρησιμοποιηθεί σε προβλήματα που οι περιορισμοί τους επιδέχονται την αποσύνθεση σε υποπροβλήματα, δηλαδή σε προβλήματα που είναι προτιμότερο να λύσει κάποιος τα υποπροβλήματα μεγάλου αριθμού με καλή δομή παρά το κύριο πρόβλημα (master problem) όπως αναφέρεται, έτσι ώστε να πετύχει λιγότερη ενδεχομένως πολυπλοκότητα αλλά και καλύτερο χρόνο επίλυσης του προβλήματος [38]. Έτσι οι Dantzig-Wolfe χρησιμοποίησαν την τεχνική της δημιουργίας της στήλης, λόγω του ότι οι στήλες ήταν μεγάλες σε αριθμό στο κύριο πρόβλημα και δεν μπορούσαν πρακτικά να τις απαριθμήσουν. Έτσι η τεχνική δημιουργίας στήλης, πετύχαινε να δημιουργήσει ένα κύριο πρόβλημα στο οποίο θα χρησιμοποιείται ένα σύνολο μεταβλητών, όπου δε θα προστίθενται άλλες μεταβλητές σε αυτό, εκτός αν χρειάζονται [39]. Ουσιαστικά η τεχνική δημιουργίας στήλης που χρησιμοποιούν οι Dantzig-Wolfe για την επίλυση του μονοδιάστατου προβλήματος θα δημιουργεί κάθε μία επόμενη στήλη μετά την επίλυση του προβλήματος του σακιδίου (knapsack) και μετά από κάθε βήμα περιστροφής της επαναληπτικής μεθόδου simplex [40].

Η μέθοδος simplex που χρησιμοποιείται είναι μία μέθοδος που χρησιμοποιήθηκε από τον George Dantzig το 1947, η οποία χρησιμοποιείται για την επίλυση προβλημάτων γραμμικής βελτιστοποίησης και ο αλγόριθμος της μεθόδου αναλύεται σε δύο στάδια που έχουν ως εξής:

1. Στο πρώτο στάδιο προσπαθεί να βρει μία λύση στο πρόβλημα ή βεβαιώνεται ότι δεν υπάρχει λύση.

2. Στο δεύτερο στάδιο προσπαθεί από τη λύση που βρέθηκε να δει αν μπορεί να τη βελτιώσει. Ουσιαστικά, για το δεύτερο στάδιο υπάρχει μία μεγάλη επανάληψη καθώς, κάθε φορά μία λύση θα βελτιώνεται μέχρι να μην μπορεί να βελτιωθεί άλλο.

Η μέθοδος simplex ή αλλιώς ο αλγόριθμος simplex είναι ένα πολύ σημαντικό εργαλείο που χρησιμοποιείται για την εύρεση βέλτιστων λύσεων. Ακόμη, θα αναφερθούμε ξανά σε αυτήν τη μέθοδο στο κεφάλαιο της ανάλυσης κωδίκων.

Η λειτουργία αποσύνθεσης των Dantzig-Wolfe ωστόσο μπορεί να αναλυθεί σε έξι βήματα για την καλύτερη κατανόηση της λειτουργίας του αλγορίθμου. Τα βήματα έχουν ως εξής:

1. Στο πρώτο βήμα ξεκινάμε έχοντας μία εφικτή λύση στο μειωμένο κύριο πρόβλημα, όπου για κάθε ένα υποπρόβλημα θα δημιουργήσουμε καινούργιες αντικειμενικές συναρτήσεις για τη βελτίωση του στόχου του κυρίου προβλήματος αφού τα υποπροβλήματα θα δώσουν λύσεις.

2. Στο δεύτερο βήμα τα προηγούμενα υποπροβλήματα θα επιλυθούν ξανά για να δώσουν βέλτιστη λύση στο κύριο πρόβλημα αφού έχουν παραχθεί νέες αντικειμενικές συναρτήσεις για το κάθε ένα.

3. Στο βήμα τρία αφού χρησιμοποιηθεί η τεχνική δημιουργίας στήλης για τη δημιουργία νέων στηλών από τις λύσεις των υποπροβλημάτων, το κύριο πρόβλημα με κύριο κριτήριο τη βελτίωση του στόχου του αρχικού προβλήματος θα εισάγει μία ή όλες τις στήλες.

4. Στο βήμα τέσσερα θα χρησιμοποιηθεί η μέθοδος simplex από το κύριο πρόγραμμα με τόσες επαναλήψεις όσος είναι και ο αριθμός των στηλών που εισήχθησαν.

5. Στο βήμα πέντε θα γίνει ένας έλεγχος, όπου αν δε βελτιώνεται ο στόχος, που σημαίνει ότι η λύση είναι η βέλτιστη, θα προχωρά στο βήμα έξι, όπου θα τελειώνει το πρόγραμμα, ενώ αν βελτιώνεται καταλήγει στο συμπέρασμα ότι δε βρήκε ακόμη τη βέλτιστη λύση για τον στόχο και επιστρέφει στο βήμα ένα.

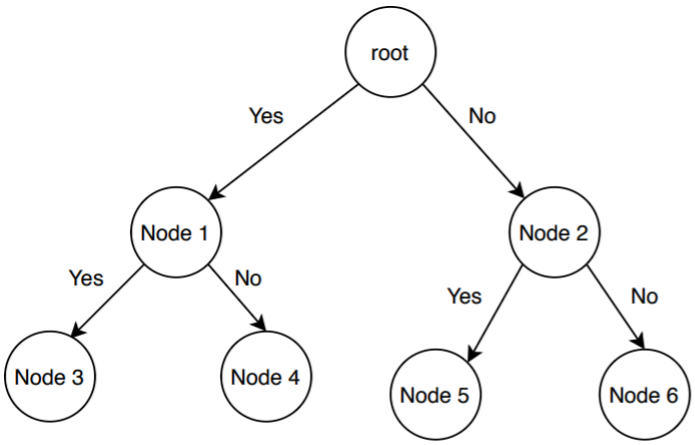
Η αποσύνθεση αυτή των Dantzig-Wolfe για την εφαρμογή της σε ακέραια προβλήματα είναι σαν μια αναδιατυπωμένη συγκεκριμένη μορφή προβλήματος που έχει ως στόχο να παράγει ένα πιο αυστηρό όριο χαλάρωσης γραμμικού προγραμματισμού, όπου οδηγεί σε ένα κύριο πρόβλημα ακέραιου προγραμματισμού, όπου για την αντιμετώπιση των μεγάλων αριθμών μεταβλητών χρησιμοποιεί τον αλγόριθμο διακλάδωσης και τιμής (branch and price) για να δημιουργήσει στήλες για

τον προγραμματισμό ακέραιων προβλημάτων. [41]. Επιπρόσθετα, η αποσύνθεση αυτή μπορεί να χρησιμοποιηθεί και για να επιλύσει το δυϊκό (dual) Lagrangian πρόβλημα ενός γραμμικού μικτού ακεραίου προβλήματος (Mixed Integer Problem) αξιοποιώντας τη χαλάρωση του Langranian προβλήματος σε συνδυασμό με τους περιορισμούς στη δυϊκή δομή του γραμμικού ακέραιου μικτού προγραμματισμού [42]. Τέλος, η τεχνική αποσύνθεσης των Dantzig-Wolfe έχει παίξει σημαντικό ρόλο σε μεγάλα προβλήματα όπου έχει εφαρμοστεί με μεγάλη επιτυχία, όπως μπορούμε να δούμε στις εργασίες [1] [43].

2.5 Μέθοδος διακλάδωσης και δέσμευσης (Branch and bound)

Σε αυτήν την υποενότητα θα γίνει αναφορά στη μέθοδο διακλάδωσης και δέσμευσης (branch and bound), όπου θα χρησιμοποιηθεί και στους αλγορίθμους που χρησιμοποιούμε στο κεφάλαιο της ανάλυσης κωδίκων για την επίλυση του μονοδιάστατου προβλήματος βέλτιστης κοπής. Η μέθοδος διακλάδωσης και δέσμευσης (branch and bound) είναι ένα πολύ γνωστό εργαλείο που χρησιμοποιείται για να λυθούν μεγάλα προβλήματα συνδυαστικής βελτιστοποίησης NP-hard [44]. Ένας αλγόριθμος διακλάδωσης και δέσμευσης ψάχνει σε ένα πρόβλημα όλο τον χώρο των λύσεων για να βρει μία καλύτερη λύση από την ήδη υπάρχουσα [44]. Ουσιαστικά εκτελεί αναζήτηση σε ένα δένδρο αναζήτησης όπου η ρίζα παρουσιάζεται ως το αρχικό πρόβλημα και τα παιδιά της ρίζας και οι απόγονοί της ως υποπροβλήματα. Ένα δένδρο αναζήτησης για παράδειγμα έχει τη μορφή του Σχήματος 2.2.

Σχήμα 2.2: Μορφή δένδρου διακλάδωσης και δέσμευσης (Branch and Bound)



Ακόμη υπάρχουν και φύλλα τα οποία είναι οι εφικτές λύσεις για το πρόβλημα. Η λειτουργία ενός αλγορίθμου διακλάδωσης και δέσμευσης είναι ότι αρχικά κατά τη διαδικασία του η καλύτερη λύση είναι το άπειρο και αυτό συμβαίνει διότι δεν έχουν ακόμα εξερευνηθεί οι υπόλοιποι κόμβοι όπου εξερευνούνται σε κάθε επανάληψη [44]. Η διαδικασία χωρίζεται σε τρία βήματα. Το πρώτο βήμα είναι σε ποιον κόμβο θα εκτελεστεί διότι υπάρχουν πολλές μέθοδοι αναζήτησης που μπορεί να συνδυαστεί, όπως για παράδειγμα είναι η αναζήτηση κατά βάθος, η αναζήτηση κατά

πλάτος, κ.λπ., όπου για κάθε αναζήτηση σημειώνεται διαφορετικός κόμβος προς επεξεργασία. Το δεύτερο βήμα είναι ο υπολογισμός ορίου (bound) και το τρίτο είναι η διακλάδωση (branch). Ουσιαστικά η μέθοδος διακλάδωσης και δέσμευσης είναι μία ακριβής μέθοδος, δηλαδή βρίσκει τη βέλτιστη λύση σε προβλήματα ακέραιου και γραμμικού προγραμματισμού, όπου στηρίζεται στην καταμέτρηση των λύσεων που είναι υποψήφιες λύσεις για το πρόβλημα αλλά και στον υπολογισμό των ορίων (άνω και κάτω) που περικόπτουν μεγάλα υποσύνολα των λύσεων αυτών για να πετύχουν τη βελτιστοποίηση [45][46]. Τέλος, υπάρχουν πολλοί ακριβείς μέθοδοι που είναι βασισμένοι στη μέθοδο της διακλάδωσης και δέσμευσης, όπως για παράδειγμα αυτές των Scheithauer και Terno [47], Vance[48], Vanderbeck [49], κ.α.

2.6 Μέθοδος διακλάδωσης και κοπής (Branch and cut)

Οι αλγόριθμοι διακλάδωσης και κοπής χρησιμοποιούνται για την επίλυση προβλημάτων, όπως τα προβλήματα μέγιστης περικοπής, το πρόβλημα μέγιστης ικανοποίησης, το πρόβλημα συσκευασίας, και άλλα συνδυαστικά προβλήματα [50]. Ωστόσο οι αλγόριθμοι αυτοί μπορούν να παράγουν μία πολύ καλή βελτιστοποίηση σε προβλήματα ακέραιου προγραμματισμού κάτι που τους κάνει να είναι πολύ επιτυχημένοι αλγόριθμοι [50]. Οι αλγόριθμοι διακλάδωσης και κοπής (branch and cut) ανήκουν στους αλγορίθμους διακλάδωσης και δέσμευσης που αναφέραμε, όπου στηρίζονται στη χαλάρωση του γραμμικού προγραμματισμού ενός μοντέλου, όπου το κύριο χαρακτηριστικό του για την ενίσχυσή του είναι ότι η χαλάρωση αυτή σε κόμβους διακλάδωσης και δέσμευσης γίνεται πιο ισχυρή από τα επίπεδα κοπής [51]. Ακόμη πρέπει να επισημάνουμε ότι οι αλγόριθμοι διακλάδωσης και αποκοπής μπορούν να συνδυαστούν και με άλλες ευρετικές μεθόδους όπως είναι οι μεθευρετικές, η τοπική αναζήτηση, αλλά μπορεί να συνδυαστεί και με την τεχνική της δημιουργίας στήλης και να δημιουργηθεί ένας αλγόριθμος διακλάδωσης και κοπής και τιμής (branch and cut and price), θέλοντας να πετύχουν ότι νέες μεταβλητές θα προστεθούν σε οποιονδήποτε κόμβο διακλάδωσης και δέσμευσης του μοντέλου [51]. Ένα παράδειγμα λειτουργίας του αλγορίθμου διακλάδωσης και δέσμευσης είναι αυτό του αλγορίθμου που ακολουθεί.

(1) Let L be a list of unsolved problems. Initialize L with (1).;
(2) **Repeat**
(3) Choose a problem Π from L and delete it from L.
(4) **Repeat** $(iterate)$
(5) Solve the (linear) relaxation of Π. Let $\bar{\mathbf{x}}$ be an optimal solution.
(6) If $\bar{\mathbf{x}}$ is feasible for Π, Π is solved: **goto** (10)
(7) Look for violated inequalities and add them to the LP.
(8) **Until** there are no violated inequalities
(9) Split Π into subproblems and add the to L.
(10)**Until** L=Ø.
(11)Print the optimal solution.
(12)**STOP**.

Αλγόριθμος 1: Αλγόριθμος διακλάδωσης και κοπής

Όπου,

- Το L είναι μια λίστα για τα άλυτα προβλήματα, ωστόσο είναι οργανωμένη ως δένδρο διακλάδωσης και δέσμευσης.
- Το είναι υποπροβλήματα ή προβλήματα που αντιστοιχούν σε κάθε σε έναν κόμβο στο δένδρο, τα άλυτα προβλήματα είναι τα φύλλα του δένδρου και ο κύριος κόμβος, δηλαδή ο κόμβος που αντιστοιχεί σε ολόκληρο το πρόβλημα ονομάζεται ρίζα [52].
- Το $\overline{\mathbf{x}}$ είναι μία βέλτιστη λύση.

Η λειτουργία του συγκεκριμένου αλγορίθμου όπως βλέπουμε είναι ότι θα διαλέξει ένα πρόβλημα από τη λίστα και θα το διαγράψει. Αμέσως μετά θα γίνει επίλυση της χαλάρωσης, όπου το $\overline{\mathbf{x}}$ θα είναι η βέλτιστη λύση, και σε περίπτωση που το $\overline{\mathbf{x}}$ (βέλτιστη λύση) δεν είναι εφικτό τότε θα κοιτάξει για παραβιασμένες ανισότητες, θα τις προσθέσει στο γραμμικό πρόβλημα, θα διαχωρίσει το πρόβλημα σε υποπροβλήματα και θα τα βάλει στη λίστα, αλλιώς αν το $\overline{\mathbf{x}}$ είναι εφικτό θα εκτελέσει τότε το επόμενο πρόβλημα από την αρχή με την ίδια διαδικασία μέχρι η λίστα να είναι κενή και θα εκτυπώσει τη βέλτιστη λύση. Η μέθοδος που χρησιμοποιείται σε έναν αλγόριθμο διακλάδωσης και κοπής για την επίλυση των γραμμικών προβλημάτων είναι η δυϊκή μέθοδος simplex (dual simplex) λόγω του ότι η βάση ενός γραμμικού προβλήματος όταν γίνεται πρόσθεση επιπέδων κοπής είναι δυϊκά εφικτή [52].

2.7 Μέθοδος διακλάδωσης και τιμής (Branch and price)

Η μέθοδος διακλάδωσης και τιμής (branch and price) είναι και αυτή μία μέθοδος που ανήκει στην κατηγορία της μεθόδου διακλάδωσης και δέσμευσης. Είναι μια υβριδική μέθοδος που συνδυάζει τη μέθοδο διακλάδωσης και δέσμευσης και την τεχνική δημιουργίας στήλης. Ουσιαστικά, πρόκειται για μία μέθοδο συνδυαστικής βελτιστοποίησης που χρησιμοποιείται για να επιλύσει μεγάλης κλίμακας προβλήματα μικτού γραμμικού προγραμματισμού (mixed integer linear programming) και ακέραιου γραμμικού προγραμματισμού (integer linear programming) όπως μπορούμε να δούμε στις εργασίες [1] [53], των οποίων τα οποία προβλήματα περιέχουν μεγάλο αριθμό μεταβλητών [48]. Για παράδειγμα, για την επίτευξη της επίλυσης των προβλημάτων μεγάλης κλίμακας που προαναφέραμε, απαιτούνται τυποποιήσεις, όπου η χαλάρωσή τους μέσω του γραμμικού προγραμματισμού, δηλαδή οι γραμμικές χαλαρώσεις, θα δίνουν πολύ καλές προσεγγιστικές λύσεις στο σύνολο (ή χώρο) των εφικτών λύσεων και η μέθοδος διακλάδωσης και τιμής δίνει μία τέτοια λύση καθώς είναι μια προσέγγιση που επιλύει τέτοιες ισχυρές συνθέσεις, δηλαδή που περιέχουν μεγάλο πλήθος μεταβλητών [48]. Για τη μέθοδο διακλάδωσης και τιμής για κάθε κόμβο του δένδρου αναζήτησης διακλάδωσης και δέσμευσης μπορεί να γίνει μία πρόσθεση στηλών στη γραμμική χαλάρωση. Έτσι, ξεκινώντας τον αλγόριθμο διακλάδωσης και τιμής αρχικά ο αριθμός των στηλών θα εξαιρεθεί από τη γραμμική χαλάρωση του προβλήματος για να πετύχουν τη μείωση των υπολογιστικών απαιτήσεων και των απαιτήσεων μνήμης, και έτσι μετά να προστεθούν και πάλι οι στήλες στη γραμμική χαλάρωση. Ουσιαστικά, αυτό θα έχει ως αποτέλεσμα για τη μεγάλη κλίμακα προβλημάτων, ότι οι στήλες αυτές θα είναι μη βασικές και η μεταβλητή τους να είναι ίση με το μηδέν για κάθε βέλτιστη λύση που θα παράγουν κάνοντας έτσι ένα μεγάλο πλήθος στηλών να μην προσφέρουν χρησιμότητα και συνεπώς λύση για την επίλυση του προβλήματος που αντιμετωπίζουμε. Μπορούμε ωστόσο, να διαχωρίσουμε τη λειτουργία του αλγορίθμου διακλάδωσης και τιμής σε βήματα για την καλύτερη κατανόηση της λειτουργίας του αλγορίθμου αυτού δίνοντας ένα παράδειγμα από την εργασία [1]. Τα βήματα αυτά για την επίλυση του προβλήματος έχουν ως εξής:

1. Στο βήμα ένα αρχικά, για τη δημιουργία του κυρίου προβλήματος, χρησιμοποιεί την αναδιατυπωμένη αποσύνθεση των Dantzig-Wolfe, όπου η χρήση της γίνεται για τη λήψη μιας προβληματικής σύνθεσης όπου με την επίλυση της χαλάρωσης θα προσφέρει καλύτερα όρια σε σχέση με τη χαλάρωση της κανονικής αποσύνθεσής τους.

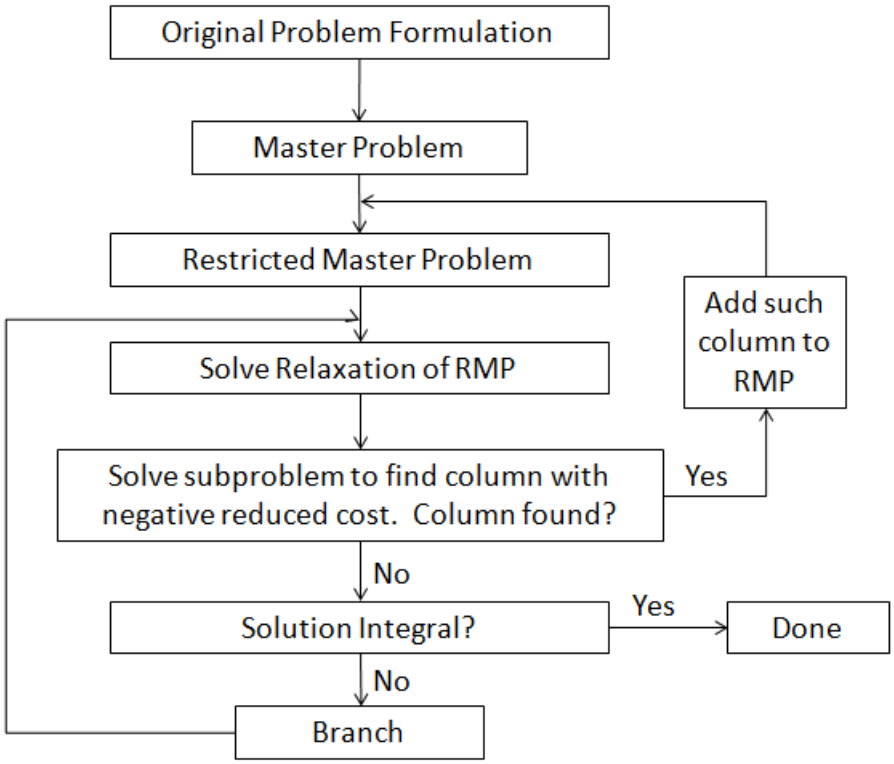
2. Στο βήμα δύο επιλύεται ένα περιορισμένο κύριο πρόβλημα (restricted master problem), λόγω του μεγάλου πλήθους μεταβλητών που περιέχει η αποσύνθεση αυτή [54].

3. Στο βήμα τρία για την εύρεση στηλών που μπορούν να μειώσουν την αντικειμενική μας συνάρτηση για το πρόβλημα της φύρας επιλύεται το υποπρόβλημα. Αυτή η διαδικασία ωστόσο θα παρέχει με την αναζήτησή της, μία στήλη όπου θα έχει ένα μειωμένο αρνητικό κόστος. Αν αυτή η στήλη βρεθεί τότε θα προστεθεί η στήλη στο περιορισμένο κύριο πρόβλημα και θα γυρίσει στο βήμα δύο, διαφορετικά θα προχωρήσει στο βήμα τέσσερα.

4. Στο βήμα τέσσερα γίνεται έλεγχος για το αν η λύση είναι ολοκληρωμένη, δηλαδή οι μεταβλητές της λύσης να έχουν όλες ακέραιες τιμές. Αν όχι τότε εφαρμόζεται η διακλάδωση (branch) και επιστρέφει στο βήμα τρία, διαφορετικά αν η λύση είναι ολοκληρωμένη τότε σταματά.

Ακόμη, μπορούμε να δούμε την αναπαράσταση των βημάτων σε μορφή διαγράμματος όπως αυτή του Σχήματος 2.3 [1].

Σχήμα 2.3: Διάγραμμα αλγορίθμου διακλάδωσης και τιμής (Branch and Price Diagram) [1]



Συνεπώς, όταν παρατηρούμε ότι για την επίλυση της γραμμικής χαλάρωσης του προβλήματος για κάθε κόμβο σε ένα δένδρο διακλάδωσης και δέσμευσης χρησιμοποιείται η τεχνική δημιουργίας στήλης, καταλαβαίνουμε ότι μιλάμε για τη μέθοδο διακλάδωσης και τιμής [55].

2.8 Μέθοδος επιπέδου κοπής (Cutting planes)

Σε αυτήν την υποενότητα θα γίνει μία σύντομη αναφορά στη μέθοδο επιπέδου κοπής (cutting planes). Η μέθοδος επιπέδου κοπής αναπτύχθηκε από τον Ralph Gomory το 1956, όπου είχε ως στόχο την επίλυση ακέραιων γραμμικών προβλημάτων [56] και προβλημάτων μικτού ακέραιου γραμμικού προγραμματισμού. Στόχος της μεθόδου αυτής είναι η επίτευξη της εύρεσης ακέραιων λύσεων μέσω της λύσης της γραμμικής χαλάρωσης του μοντέλου που χρησιμοποιείται. Ένας αλγόριθμος επιπέδου κοπής εκτελείται για να αναζητήσει λύσεις ορθών ανισώσεων που θα μπορούν να μη δέχονται και αυτόματα να κόβουν τις λύσεις που δεν είναι ακέραιες σε περιπτώσεις όπου [57]:

- 1η περίπτωση. Όταν στο αρχικό ακέραιο γραμμικό μοντέλο που έχουμε οι υπάρχοντες περιορισμοί των ανισοτήτων δεν ικανοποιούν μία λύση ακέραια.
- 2η περίπτωση. Όταν και πάλι σε αυτό το μοντέλο είναι μεγάλος ο αριθμός των περιορισμών.

Ουσιαστικά, αυτό που προσπαθεί να καταφέρει αυτή η μέθοδος είναι να βελτιώνει το χώρο αναζήτησης επαναληπτικά με τη χρήση περικοπών, δηλαδή γραμμικών ανισοτήτων για τη διατήρηση της αρχικής εφικτής περιοχής. Η βέλτιστη λύση για ένα μοντέλο γραμμικό μπορεί να προέλθει από ένα γωνιακό ή ακέραιο σημείο που βρίσκεται στην εφικτή περιοχή όπως αναφέρθηκε αλλά αυτή η βέλτιστη λύση μπορεί και να μην είναι ακέραια. Στην περίπτωση που δεν είναι ακέραια η λύση, δηλαδή ο αριθμός, τότε κόβει από την εφικτή περιοχή ένα κομμάτι για να διαχωρίσει τη βέλτιστη λύση. Αυτό, θα έχει ως αποτέλεσμα την εισαγωγή ενός νέου προβλήματος διαχωρισμού και θα προστεθεί μία κοπή (ή περικοπή) στο γραμμικό μοντέλο, όπου θα πετυχαίνει τη διαγραφή της μη ακέραιας λύσης που υπάρχει [57]. Αυτή η μέθοδος περικοπής δε θα σταματάει μέχρι η βέλτιστη λύση να είναι ακέραια. Ωστόσο μπορούμε να αναλύσουμε την παραπάνω διαδικασία με βήματα που παρουσιάζουμε παρακάτω για τη μέθοδο επιπέδου κοπής. Η διαδικασία έχει ως εξής:

1. Στο βήμα ένα επιλύει τη χαλάρωση ακέραιου προγραμματισμού.

2. Στο βήμα δύο, αν οι μεταβλητές της λύσης που προτείνεται είναι ακέραιες, τότε βρίσκει τη βέλτιστη λύση και σταματάει, διαφορετικά προχωράει με την τεχνική περικοπής, όπου προσθέτει έναν περιορισμό σαν μια γραμμική ανισότητα που ικανοποιείται από όλον τον αριθμό των εφικτών λύσεων.

3. Στο βήμα τρία, βάζει τον περιορισμό που προαναφέρθηκε στο στο μαθηματικό μοντέλο, έπειτα επιλύει το μοντέλο αυτό και τέλος πηγαίνει στο βήμα δύο.

Τέλος, στην περίπτωση που υπάρξει σύσφιξη της χαλάρωσης του γραμμικού προβλήματος έπειτα από τη χρήση της μεθόδου επιπέδου κοπής σε μία μέθοδο διακλάδωσης και τιμής, τότε αυτό ονομάζεται μέθοδος διακλάδωσης και περικοπής (branch and price and cut) [55].

2.9 Βιβλιογραφική ανασκόπηση

Σε αυτήν την υποενότητα θα γίνει αναφορά σε αλγορίθμους που επιλύουν το πρόβλημα της βέλτιστης κοπής, έπειτα θα ακολουθήσουν αλγόριθμοι από διάφορες πηγές και θα δούμε με ποιους τρόπους έχουν αυτοί οι αλγόριθμοι επιλύσει το πρόβλημα βέλτιστης κοπής.

Αρχικά, θα χωρίσουμε τους αλγορίθμους που επιλύουν το πρόβλημα σε δύο κατηγορίες, όπου:

1. Η πρώτη κατηγορία που θα δούμε εμπεριέχει τους ευρετικούς αλγορίθμους.
2. Η δεύτερη κατηγορία που θα δούμε εμπεριέχει τους ακριβείς αλγορίθμους.

Για τους ευρετικούς αλγορίθμους, θα δούμε πηγές όπου επιλύουν το πρόβλημα με υβριδικό αλγόριθμο, με γενετικό αλγόριθμο ή και με άλλους ευρετικούς αλγορίθμους. Για τους ακριβείς αλγόριθμους θα δούμε αλγορίθμους που χρησιμοποιούν τεχνικές και μεθόδους αλγορίθμων που επιλύουν το πρόβλημα βέλτιστης κοπής, όπως για παράδειγμα αλγορίθμους διακλάδωσης και δέσμευσης, διακλάδωσης και κοπής κ.α. Η χρήση των ευρετικών αλγορίθμων γίνεται για να επιλύσουν προβλήματα κυρίως μη ντετερμινιστικά πολυωνυμικού χρόνου (NP) προβλήματα δίνοντας γρήγορες λύσεις που μπορεί να είναι και βέλτιστες, αλλά και για τη μείωση της πολυπλοκότητας του χρόνου των προβλημάτων αυτών. Η χρήση των ακριβών αλγορίθμων γίνεται προκειμένου να επιλύσουν προβλήματα με βέλτιστο τρόπο.

2.9.1 Ευρετικοί αλγόριθμοι

Σε αυτήν την υποενότητα θα δούμε ευρετικούς αλγορίθμους που έχουν προταθεί σε διάφορες εργασίες για να πετύχουν την επίλυση του μονοδιάστατου προβλήματος βέλτιστης κοπής.

Οι Yanasse και Limeira [58] χρησιμοποίησαν έναν υβριδικό αλγόριθμο για την αντιμετώπιση του μονοδιάστατου προβλήματος. Πιο συγκεκριμένα, χρησιμοποίησαν έναν υβριδικό για να ληφθεί ένα μειωμένο πλήθος από διαφορετικά μοτίβα για το πρόβλημα. Η διαδικασία που ακολούθησαν είχε να κάνει αρχικά με τη μείωση του προβλήματος και τη λύση του υπολοιπόμενου προβλήματος χρησιμοποιώντας τη δημιουργία μοτίβων που έχουν περιορισμένο πλήθος φύρας, που θα ικανοποιούσαν όμως δύο είδη τουλάχιστον ως προς τις απαιτήσεις τους όταν αυτά τα μοτίβα κόβονται κάθε φορά και συνεχόμενα. Έπειτα εφάρμοσαν τεχνικές για να καταφέρουν να μειώσουν τα πρότυπα με τη χρήση της τοπικής αναζήτησης, όπου ξεκινούσε αρχικά από τη λύση που δημιουργείται. Ωστόσο, έδειξαν ότι το σχέδιο που χρησιμοποίησαν είναι πολύ απλό και ότι για κάθε διάσταση του προβλήματος (δισδιάστατα, τρισδιάστατα) μπορεί να χρησιμοποιηθεί για να επιλύσει το πρόβλημα. Τέλος, οι δοκιμές που έκαναν για τον υπολογισμό του αποτελέσματος έδειξαν ότι το σχήμα που πρότειναν παρείχε διαφορετικές (ή εναλλακτικές) λύσεις για το πρόβλημα των μειώσεων των προτύπων.

Οι Cui και Yang [59] για το μονοδιάστατο πρόβλημα παρουσίασαν έναν ευρετικό αλγόριθμο που χρησιμοποιεί το υπολοιπόμενο μήκος. Ο αλγόριθμος που χρησιμοποίησαν αποτελούνταν από δύο διαδικασίες οι οποίες είναι οι εξής:

1. Διαδικασία που στηρίζεται στον γραμμικό προγραμματισμό για την ικανοποίηση του περισσότερου μέρους για τη ζήτηση των ειδών.

2. Διαδικασία ευρετική για την ικανοποίηση για το υπόλοιπο μέρος της ζήτησης των ειδών.

Στηρίχθηκαν στην ιδέα ότι αυτός ο αλγόριθμος μπορεί να αποδώσει μία σταθερή ισορροπία ανάμεσα στα κέρδη των υπολειμμάτων, τα κόστη των ράβδων της κατανάλωσης και τα κέρδη λόγω της μείωσης των μικρότερων σε αριθμό αποθεμάτων. Τα αποτελέσματα που έβγαλαν χρησιμοποιώντας αυτόν τον αλγόριθμο για τον υπολογισμό τους έδειξαν ότι είναι πολύ ικανοποιητικά, κάτι που σημαίνει ότι ο αλγό-

ριθμος που χρησιμοποίησαν ήταν πολύ αποδοτικός και σε πραγματικά προβλήματα. Τα αποτελέσματα σύγκρισης μεταξύ του αλγορίθμου RSHP που δημιούργησαν και μεταξύ άλλων τριών αλγορίθμων που επιλύουν το πρόβλημα από άλλη πηγή που έχουν τις ονομασίες RGRL1, RGRL2, RGRL3 της πηγής [60] παρουσιάζονται στον παρακάτω. Τα αποτελέσματα αυτά είναι αποτελέσματα της μέσης τιμής όλων των οκτώ προβλημάτων.

Αποτελέσματα σύγκρισης Yaodong Cui και Yuli Yang				
Categories	RSHP	RGRL1	RGRL2	RGRL3
Total bar length	1,11,672	1,11,958	1,11,924	1,11,979
Standard bar count	10.,5	102.6	102.5	102.6
Non-standard bar count	23.6	4.3	4.4	4.5
Total trim loss	9.8	11.5	11.8	12.5
Total leftover length	303.2	587.4	552.8	601.9
Average leftover count	1.16	2.6	2.6	2.8
Average length of leftover	261.5	225.9	212.6	215.0
Average computation time (s)	1.56	22.55	22.74	81.85

Για τα αποτέλεσμα του παραπάνω πίνακα παρατηρήθηκε λοιπόν ότι ο αλγόριθμος που παρουσίασαν είναι όντως πιο αποδοτικός σε σύγκριση με τους άλλους τρεις που χρησιμοποίησαν, καθώς παράγει τις πιο αποδοτικές τιμές για κάθε κατηγορία.

Ο Lee [61] παρουσίασε έναν ευρετικό αλγόριθμο τοπικής αναζήτησης βασισμένο στον ακέραιο γραμμικό προγραμματισμό. Αυτή η αναζήτηση ονομάζεται Crawla. Αυτός ο ευρετικός αλγόριθμος έχει ως λειτουργία την ενσωμάτωση του κύριου προβλήματος και του υποπροβλήματος του παραδείγματος, που δημιουργεί πρότυπα έχοντας ως βάση τις τιμές. Ουσιαστικά, αυτό που καταφέρνει με τη λειτουργία αυτή είναι να έχει ένα μοντέλο ομαδοποιημένο το οποίο θα παράγει μοτίβα κοπής. Ωστόσο, αυτός ο αλγόριθμος έχει ως μειονέκτημα τον χρόνο παραγωγής στηλών, δηλαδή δε δίνει βάση για πόσες στήλες θα δημιουργήσει. Παρόλα αυτά όμως, παρέχει

την εγγύηση ότι οι στήλες που θα δημιουργεί θα παρέχουν μια ακέραια βελτίωση στο γραμμικό πρόγραμμα.

Ο Golfeto [62] πρότεινε έναν αλγόριθμο ο οποίος είχε ως στόχο την ελαχιστοποίηση του πλήθους των υλικών του μονοδιάστατου προβλήματος, αλλά και τη ρύθμισή του. Αυτός ο αλγόριθμος είναι ένας γενετικός αλγόριθμος ο οποίος χρησιμοποιείται ουσιαστικά για να συνδυάσει τα μοτίβα κοπής, δηλαδή τα μήκη της παραγγελίας του πελάτη και την ίδια στιγμή τη συχνότητα αυτών. Αυτό το πετυχαίνει με μία συμβιωτική μέθοδο όπου χρησιμοποιείται σε δύο διαφορετικούς πληθυσμούς, δηλαδή αυτόν της λύσης και του σχεδίου κοπής. Τέλος, τα αποτελέσματα του αλγορίθμου αυτού ήταν πολύ καλά κατά τη σύγκριση που υλοποιήθηκε με άλλους αλγορίθμους.

Οι Cui και Liu [63] πρότειναν μία ευρετική διαδοχική διαδικασία που την ονόμασαν SHP για την επίλυση του μονοδιάστατου προβλήματος βέλτιστης κοπής, όπου έχει ως πρώτο στόχο να κόβεται ένα πλήθος από αντικείμενα που έχουν το ίδιο μήκος με αυτά για να πετύχει να ελαχιστοποιεί το κόστος του ίδιου του αντικειμένου και ως δεύτερο στόχο έχει το να μειωθεί το πλήθος των προτύπων του σχεδίου κοπής. Έτσι, αυτός ο αλγόριθμος, δηλαδή ο SHP, για να παράγει ορισμένα στοιχεία εκτελεί τη δημιουργία του τρέχοντος μοτίβου και συνεχίζει αυτήν τη διαδικασία μέχρι να παραχθούν όλα τα στοιχεία. Ωστόσο, ο αλγόριθμος χρησιμοποιεί τα υποψήφια στοιχεία που έχει παράξει για την παραγωγή υποψήφιων μοτίβων κοπής και έπειτα με τη χρήση μιας στρατηγικής look-ahead κάνει εκτίμηση του κόστους για το κάθε σχέδιο κοπής και τέλος επιλέγει το μικρότερο σε κόστος μοτίβο κοπής. Για αυτόν τον αλγόριθμο, εκτίμησαν ότι βάση των αποτελεσμάτων που παράχθηκαν, ο αλγόριθμος είναι αποτελεσματικός.

2.9.2 Ακριβείς αλγόριθμοι

Σε αυτήν την υποενότητα θα δούμε τη χρήση αλγορίθμων με ακριβή λύση που επιλύουν το πρόβλημα βέλτιστης κοπής από διάφορες πηγές.

Οι Alves et al. [64] για να λύσουν το πρόβλημα της ελαχιστοποίησης των προτύπων παρουσίασαν έναν αλγόριθμο διακλάδωσης και τιμής και περικοπής που επιλύουν αυτό το πρόβλημα ακριβώς. Χρησιμοποίησαν διπλές ανισότητες στον αλγόριθμο διακλάδωσης και τιμής και περικοπής και έδειξαν πως οι ανισότητες αυτές μπορούν μπορούν να χρησιμεύσουν σε συνδυασμό με τη δημιουργία στηλών για να μπορέσει να γίνει ταχύτερη διακλάδωση στους κόμβους του δένδρου. Τα αποτελέσματα που παράχθηκαν από τη διαδικασία υπολογισμών τους από τον αλγόριθμο αυτό έδειξαν, ότι αυτή η τεχνική στο δένδρο διακλάδωσης και δέσμευσης μπορεί να καταλήξει σε σημαντική βελτίωση για τον αλγόριθμο διακλάδωσης και τιμής και κοπής. Τα αποτελέσματα που υπολογίστηκαν προέρχονταν από προβλήματα που δημιούργησε ο Monaci [65] και οι Belov και Scheithauer [66]. Τέλος, κατέληξαν στο γεγονός ότι με τη χρήση των διπλών ανισοτήτων επιλύθηκε το 94 τοις εκατό των προβλημάτων με βέλτιστο τρόπο.

Ο Vance [48] παρουσίασε και σύγκρινε δύο ακριβείς αλγορίθμους για την επίλυση του μονοδιάστατου προβλήματος βέλτιστης κοπής, χρησιμοποιώντας αλγορίθμους διακλάδωσης και τιμής. Πιο συγκεκριμένα, και οι δύο αλγόριθμοι βασίστηκαν σε διατυπώσεις ακέραιου προγραμματισμού του προβλήματος δημιουργίας στηλών αλλά διαφορετικές. Η μία κατέληγε σε ένα κύριο πρόβλημα με δυαδικές μεταβλητές ακέραιες και η άλλη είχε ακέραιες μεταβλητές. Ωστόσο για την επίλυση της γραμμικής χαλάρωσης σε κάθε κόμβο στο δένδρο διακλάδωσης και δέσμευσης, για να ληφθούν βέλτιστες ακέραιες λύσεις, και οι δύο αλγόριθμοι χρησιμοποιούν την τεχνική δημιουργίας στηλών. Ουσιαστικά οι αλγόριθμοι αυτοί έχουν διαφορετικές προσεγγίσεις στη μέθοδο διακλάδωσης και τιμής. Τα αποτελέσματα που παράχθηκαν από τα προβλήματα που επιλέχθηκαν έδειξαν ότι και οι δύο αυτοί αλγόριθμοι πέτυχαν την επίλυση των πραγματικών προβλημάτων. Ωστόσο, λόγω των μικρότερων μεγεθών των γραμμικών χαλαρώσεων που επίλυσαν, η πιο συμπαγής σύνθεση απέδωσε καλύτερα, παρόλο που δεν ήταν τόσο συνεπής ως προς την απόδοσή της στον αριθμό των προβλημάτων που επιλύθηκαν.

Οι Scheithauer et al. [67] παρουσίασαν μία ακριβή λύση για το μονοδιάστατο

πρόβλημα βέλτιστης κοπής, όπου χρησιμοποίησαν την τεχνική δημιουργίας στήλης σε συνδυασμό με τη μέθοδο επιπέδου κοπής. Ουσιαστικά ο αλγόριθμος επιπέδου κοπής κατά τον συνδυασμό του, χρησιμοποιήθηκε στο πρόβλημα με έναν πίνακα περιορισμού έμμεσα. Παρατηρήθηκε ωστόσο από τα αποτελέσματα πως ο αλγόριθμος αυτός ανταγωνίζεται τον αλγόριθμο διακλάδωσης και τιμής.

Κεφάλαιο 3

Ανάλυση κωδίκων και αλγόριθμοι αναζήτησης

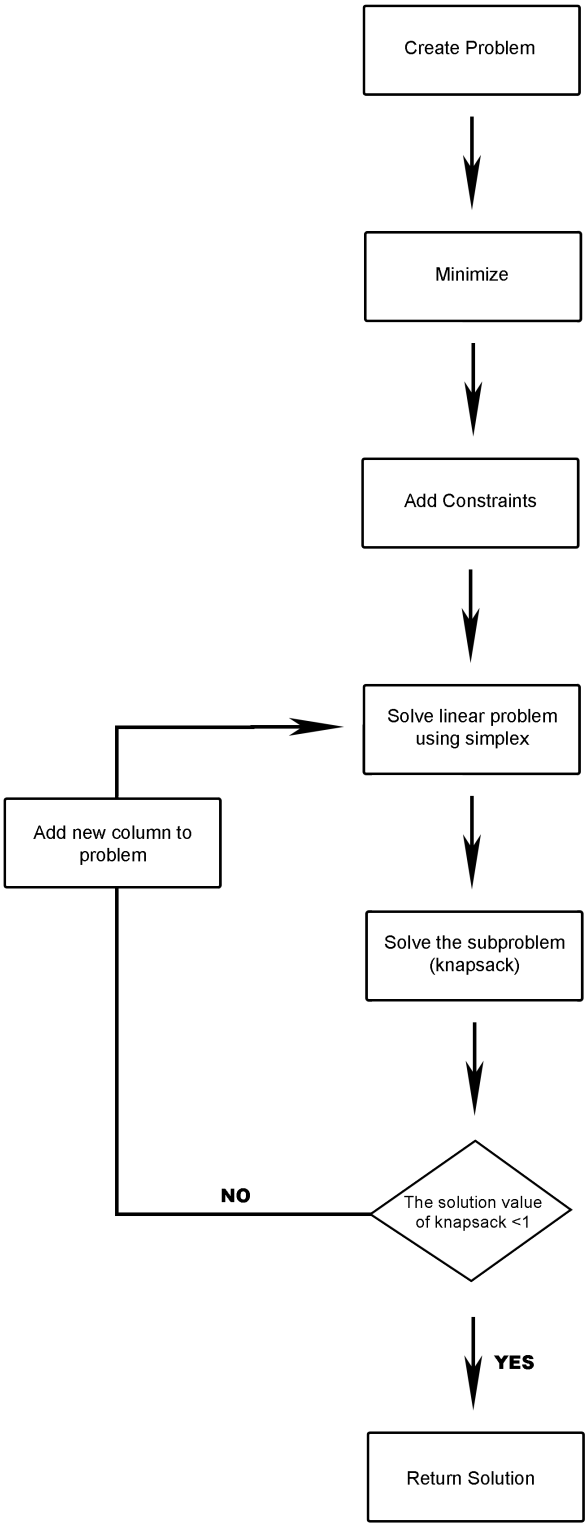
Σε αυτό το κεφάλαιο θα γίνει ανάλυση των κωδίκων που χρησιμοποιήσαμε για την επίλυση του μονοδιάστατου προβλήματος βέλτιστης κοπής. Οι αλγόριθμοι που χρησιμοποιήθηκαν για την επίλυση του προβλήματος είναι ένας αλγόριθμος για την επίτευξη της γραμμικής χαλάρωσης (linear relaxation algorithm) και ένας ακριβής αλγόριθμος, όπου εμπεριέχει τον συνδυασμό μεθόδων, τεχνικών, προβλημάτων και αλγορίθμων που συζητήθηκαν στα προηγούμενα κεφάλαια για την επίλυση του προβλήματος. Για όλα τα ζητούμενα συνολικά υλοποιήθηκαν δεκαοκτώ κλάσεις (cpp αρχεία) για την επίλυση του προβλήματος. Αρχικά λοιπόν, θα γίνει ανάλυση του κώδικα επίλυσης της γραμμικής χαλάρωσης και αμέσως μετά θα ακολουθήσει η ανάλυση του ακριβή αλγόριθμου. Ακόμη, θα ακολουθήσουν διαγράμματα με τα βήματα που ακολούθησαν οι αλγόριθμοι αυτοί για την επίλυση του προβλήματος έτσι ώστε να γίνουν πιο κατανοητοί. Ωστόσο, θα γίνει αναφορά σε μεθόδους (ή αλγορίθμους) αναζήτησης που χρησιμοποιήθηκαν για να βοηθήσουν τη διαδικασία επίτευξης της λύσης σε παρακάτω υποενότητες.

3.1 Αλγόριθμος γραμμικής χαλάρωσης

Για την επίτευξη της γραμμικής χαλάρωσης του προβλήματος (κλάση Cutting StockLinearRelaxation), αρχικά λαμβάνει τα δεδομένα του προβλήματος (κλάση CuttingStockInstance) και τα αποθηκεύσει σε μεταβλητές, δηλαδή τον αριθμό των μηκών της παραγγελίας του πελάτη, τα μήκη της παραγγελίας και το μήκος σταθερού στοκ που έχει η επιχείρηση στο αποθεμά της. Στη συνέχεια δημιουργεί το γραμμικό πρόβλημα μέσω συναρτήσεων του λύτη GLPK και επιλύει το πρόβλημα ελαχιστοποίησης που δημιούργησε. Ο πίνακας $A = a[i, j]$ που δημιουργείται αντιπροσωπεύει τα μοτίβα κοπής. Κάθε γραμμή A_j αντιπροσωπεύει ένα μοτίβο κοπής, όπου το $a[i, j]$ είναι ο αριθμός των κομματιών από το μήκος στοκ $w[i]$ που παράχθηκε από το μοτίβο κοπής j. Ο πίνακας A παρουσιάζεται σε γραμμική μορφή χρησιμοποιώντας πίνακες. Ουσιαστικά ο πίνακας A θα ξεκινήσει με ένα εφικτό το σύνολο m στηλών, που θα έχει τη μορφή ενός πίνακα ταυτότητας, κάτι που σημαίνει ότι για κάθε μοτίβο κοπής j από το 1 μέχρι το m θα παράγεται μόνο μία μπάρα μήκους $w[j]$, δηλαδή αυτό σημαίνει ότι για κάθε $a[j, j] = 1$ τα υπόλοιπα μήκη θα παράγουν μηδέν μπάρες. Αυτό στην ουσία γίνεται για δούμε αν κάποιο μήκος παραγγελίας είναι ακριβώς ίσο με το μήκος στοκ της επιχείρησης και μετά προσθέτει το μήκος αυτό (δηλαδή το μοτίβο κοπής) στη λύση αφού πρώτα δημιουργήσει ένα βαθύ αντίγραφο (κλώνο) του μοτίβου χρησιμοποιώντας μία συνάρτηση της κλάσης CuttingStockPattern. Χρησιμοποιεί τον αλγόριθμο simplex για να λύσει το πρόβλημα. Έπειτα, λαμβάνει τις δυϊκές μεταβλητές μέσω μιας συνάρτησης του GLPK που θα χρειαστεί για το υποπρόβλημα που θα λύσει. Ακολουθεί η διαδικασία όπου κάθε στήλη μοτίβου κοπής A_j θα παράγεται με την επίλυση του ακέραιου προβλήματος του σακιδίου. Αυτό το πετυχαίνει, δημιουργώντας αρχικά το πρόβλημα του σακιδίου μέσω της κλάσης Knapsack και έπειτα μέσω μιας της συνάρτησης της κλάσης IntegerKnapsackSolver επιλύει το υποπρόβλημα αυτό μέσω δυναμικού προγραμματισμού, αποθηκεύει τη λύση για κάθε μοτίβο στη μεταβλητή λύσης (solution). Έπειτα μέσω μιας συνάρτησης της κλάσης FloatingPointComparer, ελέγχει αν η τιμή της λύσης του προβλήματος του σακιδίου είναι μικρότερη του 1 αφού πρώτα πάρει τη βέλτιστη τιμή της λύσης του υποπροβλήματος. Αν ναι, τότε σημαίνει ότι όλα τα μειωμένα κόστη είναι μεγαλύτερα ή ίσα του μηδενός, άρα δεν υπάρχει περίπτωση

να βελτιστοποιηθεί άλλο το σύστημα και συνεχίζει για το επόμενο μοτίβο. Σε άλλη περίπτωση όμως, που δεν ισχύει το παραπάνω, θα προσθέσει ένα νέο μοτίβο, δηλαδή μία νέα στήλη A_j (δηλαδή λύση στο πρόβλημα του σακιδίου) στο στιγμιότυπο (instance), και θα χρησιμοποιήσει ξανά τον αλγόριθμο simplex στην επόμενη επανάληψη. Τέλος, όταν τελειώσει το υποπρόβλημα για όλες τις τιμές τότε το πρόγραμμα φορτώνει τη λύση μέσω συνάρτησης του GLPK και τις παίρνει, αφού πρώτα σταματήσει τον μετρητή για το υποπρόβλημα και για το συνολικό πρόβλημα. Έπειτα, θα επιστρέψει τη λύση, δηλαδή τη μεταβλητή solution. Στο Σχήμα 3.1 βλέπουμε το διάγραμμα με τη λειτουργία του κώδικα της γραμμικής χαλάρωσης που είδαμε παραπάνω.

Σχήμα 3.1: Διάγραμμα αλγορίθμου γραμμικής χαλάρωσης (Linear Relaxation Diagram)

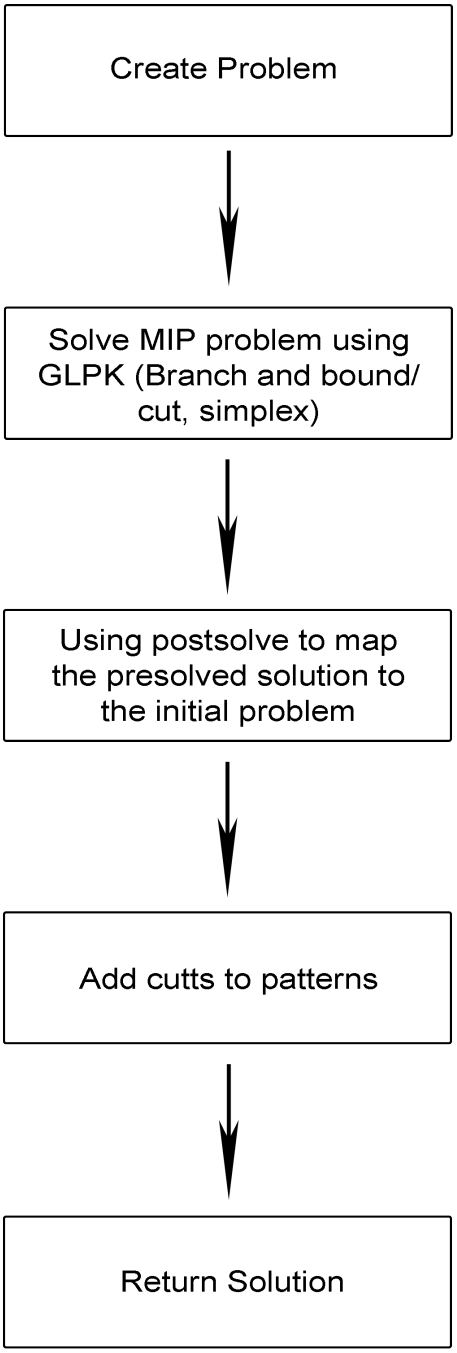


3.2 Ακριβής αλγόριθμος

Για την επίλυση του προβλήματος με τη χρήση του ακριβή αλγορίθμου, υλοποιήθηκε η κλάση CuttingStockExactIntegerSolver, όπου αρχικά μέσω της κλάσης MathProgInstace λαμβάνει τα δεδομένα του προβλήματος που δίνεται. Μετά, μέσω συναρτήσεων δημιουργεί το πρόβλημα μικτού ακέραιου προγραμματισμού (MIP). Αμέσως μετά, δημιουργεί και αρχικοποιεί μία παράμετρο ελέγχου από συναρτήσεις του GLPK. Αυτή η παράμετρος ελέγχου είναι μία παράμετρος βελτιστοποίησης ακεραίων που χρησιμοποιείται από τον βελτιστοποιητή. Στη συνέχεια, καθορίζει τη λειτουργία backtracking διακλάδωσης και κοπής, δηλαδή εδώ, καθορίζει με ποια αναζήτηση θα προχωρήσει για την εύρεση βέλτιστων τιμών του προβλήματος και την αποθηκεύει σε μια μεταβλητή της παραμέτρου ελέγχου. Ωστόσο, γίνεται η πρόσθεση άλλης μίας μεταβλητής όπου αυτή τη φορά θα καθορίζει την ώρα που μπορεί η μέθοδος να εκτελείται για να βρει τη λύση. Επιπρόσθετα, μέσω συναρτήσεων του GLPK φορτώνει τις πληροφορίες για τη δημιουργία του μικτού ακέραιου προβλήματος από το μοντέλο, εκτελεί τη μέθοδο simplex για να λύσει το πρόβλημα, και αμέσως μετά εκτελείται μια συνάρτηση του GLPK όπου βάζει σε εκκίνηση τη διακλάδωση και περικοπή. Ακόμη, αντιγράφει τη λύση από το μικτό ακέραιο πρόβλημα στον μεταφραστή και στη συνέχεια εκτελεί όλες τις υπόλοιπες εντολές μοντέλου, οι οποίες ακολουθούν τη δήλωση επίλυσης. Έπειτα, δημιουργεί τη μεταβλητή λύσης, χρησιμοποιεί μία συνάρτηση για πάρει την τιμή της λύσης του μικτού ακέραιου προβλήματος και στη συνέχεια με μία συνάρτηση τη θέτει σε μία μεταβλητή. Ακόμη, χρησιμοποιεί μία συνάρτηση για να πάρει τον τρέχοντα αριθμό στηλών από το πρόβλημα και τα αποθηκεύει σε μία μεταβλητή columns, παίρνει τα δεδομένα της παραγγελίας και τα αποθηκεύει στο m και χρησιμοποιεί μία μεταβλητή n = columns/(1+m). Ακόμη, προσθέτει έναν κλώνο του μοτίβου για κάθε μοτίβο χρησιμοποιώντας μία συνάρτηση της κλάσης CuttingStockPattern, μετά τσεκάρει αν βρέθηκε μια ακέραια εφικτή λύση για κάθε j από 1 έως n στο μικτό ακέραιο πρόβλημα, με τη χρήση μιας συνάρτησης του GLPK που μπορεί επίσης να ανακτήσει αυτές τις τιμές και και κρατάει τη λύση για το κάθε ένα. Ωστόσο, εκτελεί το ίδιο πράγμα αυτή τη φορά για την n+1 στήλη όμως για i από 1 έως m, εκτελεί τη συνάρτηση κοπής και αποθηκεύει τη λύση. Τέλος, επιστρέφει τη λύση. Στο Σχήμα 3.2 διακρίνουμε τη λειτουργία του

κώδικα ακριβούς λύσης που αναλύσαμε παραπάνω.

Σχήμα 3.2: Διάγραμμα ακριβή αλγόριθμου (Exact Diagram)



3.3 Αλγόριθμοι αναζήτησης

Σε αυτήν την υποενότητα θα δούμε τους αλγορίθμους αναζήτησης που χρησιμοποιούνται στη μέθοδο διακλάδωσης και περικοπής στον ακριβή αλγόριθμο που είδαμε παραπάνω προκειμένου να καθορίσουν τον τρόπο με τον οποίο θα γίνει η αναζήτηση για την καλύτερη λύση. Σημειώνουμε, ότι οι αναζητήσεις αυτές επιλέγονται από τη μεταβλητή backtracking που είδαμε. Η μέθοδος backtracking είναι μία μέθοδος, όπου βασίζεται στο να συνδυάσει τα πλεονεκτήματα εύρεσης καλών ακέραιων εφικτών λύσεων και απόδειξης εύρεσης καλύτερης λύσης από το ήδη υπάρχον κατώτερο όριο [68]. Έτσι, για τους αλγορίθμους επιλέχθηκαν τέσσερις τρόποι αναζήτησης που είναι οι εξής:

1. Αναζήτηση με το καλύτερο τοπικό όριο.
2. Αναζήτηση με την καλύτερη ευρετική προβολή.
3. Αναζήτηση πρώτα κατά βάθος.
4. Αναζήτηση πρώτα κατά πλάτος.

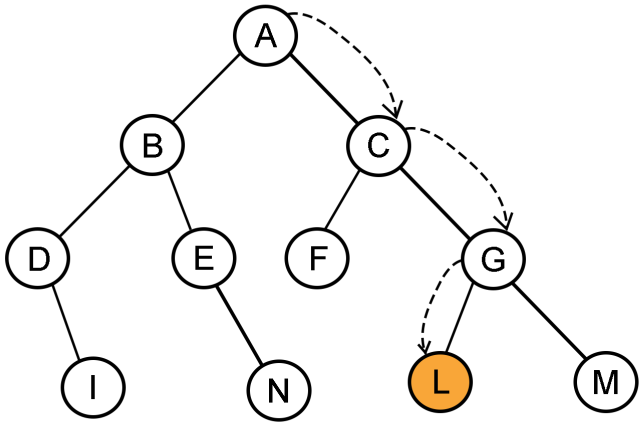
Η διαφορετική επιλογή αυτών των τρόπων αναζητήσεων παίζουν πολύ σημαντικό ρόλο στην πορεία επίλυσης του προβλήματος, καθώς ο κάθε τρόπος εκτελεί την αναζήτηση με διαφορετικό τρόπο, κάτι που σημαίνει ότι το αποτέλεσμα δεν είναι το ίδιο. Ωστόσο, αυτό είναι κάτι που θα δούμε στο επόμενο κεφάλαιο, δηλαδή αυτό της σύγκρισης.

3.3.1 Αλγόριθμος καλύτερου τοπικού ορίου (Best local bound)

Σε αυτήν την υποενότητα θα δούμε τον αλγόριθμο καλύτερου τοπικού ορίου και θα αναλύσουμε και τη λειτουργία του. Για τον αλγόριθμο αναζήτησης καλύτερου ορίου για προβλήματα μικτού ακέραιου προγραμματισμού σε δένδρο διακλάδωσης και δέσμευσης, η λειτουργία του είναι να διαλέγει το ενεργό υποπρόβλημα και να επιστρέφει τον καλύτερο κόμβο. Ουσιαστικά για να το πετύχει αυτό η λειτουργία του είναι αρχικά να διαλέγει το υποπρόβλημα με τον καλύτερο κόμβο, να δηλώνει δύο δείκτες (pointer), έναν για τον κόμβο και έναν ακόμη για να αποθηκεύει που είναι ο καλύτερος κόμβος. Η λειτουργία του εφαρμόζεται και στην περίπτωση της ελαχιστοποίησης αλλά και της μεγιστοποίησης. Για την περίπτωση της ελαχιστοποίησης, αρχικά ξεκινάει θέτοντας ένα όριο (μεταβλητή) δίνοντάς του τη μέγιστη τιμή και έπειτα με βάση αυτό το όριο πηγαίνει στον επόμενο κόμβο με την καλύτερη τιμή για αρχή (node από το head) και έπειτα τσεκάρει με βάση το όριο που έχει αν είναι μεγαλύτερο από το όριο του κόμβου που βρίσκεται εκείνη τη στιγμή. Αν ναι, τότε αντικαθιστά την τιμή του ορίου που έχει ήδη με την τιμή του ορίου το κόμβου που ελέγχει, και μετά παίρνει την απόλυτη τιμή του ορίου και την προσθέτη στην τιμή της μεταβλητής eps. Η eps είναι ένας μικρός θετικός αριθμός, όπου στα περισσότερα παραδείγματα παίρνει την τιμή του 10^{-9} και χρησιμοποιείται για να παραβλέψει τα xi_i, δηλαδή τους συντελεστές επιρροής περιορισμών. Η πράξη που γίνεται δηλαδή, είναι η $eps = 0.001 * (1.0 + fabs(bound))$, όπου το fabs όπως βλέπουμε είναι μία συνάρτηση που επιστρέφει την απόλυτη τιμή του ορίσματος, δηλαδή της μεταβλητής bound. Έπειτα, μόλις τελειώσει με όλους τους κόμβους, ελέγχει πάλι από την αρχή για κάθε όριο ξεκινώντας με το όριο του head αν είναι μικρότερο ή ίσο της τιμής του εαυτού του σε συνδυασμό με τη μεταβλητή eps. Αν ναι τότε προχωρά σε έναν ακόμη έλεγχο, όπου ελέγχει αν από τον καλύτερο κόμβο που έχει μέχρι στιγμής το άθροισμα των ακέραιων αδυναμιών είναι μεγαλύτερο από του ίδιου αθροίσματος του κόμβου και κρατάει στη μεταβλητή best, την τιμή της μεταβλητής του κόμβου του ορίου και συνεχίζει στον επόμενο κόμβο. Αλλιώς, κάνει έλεγχο αν η βέλτιστη αντικειμενική τιμή του καλύτερου κόμβου μέχρι στιγμής είναι μεγαλύτερη της αντίστοιχης τιμής της γραμμικής χαλάρωσης του κόμβου, που είναι εκείνη τη στιγμή. Αν ναι, κρατάει στη μεταβλητή best την τιμή της μεταβλητής του κόμβου του ορίου και συνεχίζει στον επόμενο κόμβο, αν όχι απλά πάει στον επόμενο κόμβο.

Μόλις τελειώσει η διαδικασία, θα έχει τον κόμβο με την καλύτερη τιμή ορίου και θα την επιστρέψει. Για την περίπτωση της μεγιστοποίησης ακολουθεί η ακριβώς ίδια διαδικασία με την περίπτωση της ελαχιστοποίησης που είδαμε παραπάνω, όμως με τη διαφορά ότι το αρχικό όριο παίρνει την ελάχιστη τιμή και στον τελευταίο έλεγχο αλλάζει η ανισότητα για να βρίσκει το καλύτερο όριο από μικρότερο ή ίσο σε μεγαλύτερο ή ίσο για να παίρνει το όριο και να συνεχίζει τον έλεγχο. Η μέθοδος αναζήτησης έχει τη μορφή του Σχήματος 3.3.

Σχήμα 3.3: Μέθοδος αναζήτησης καλύτερου τοπικού ορίου (Best Local Bound)

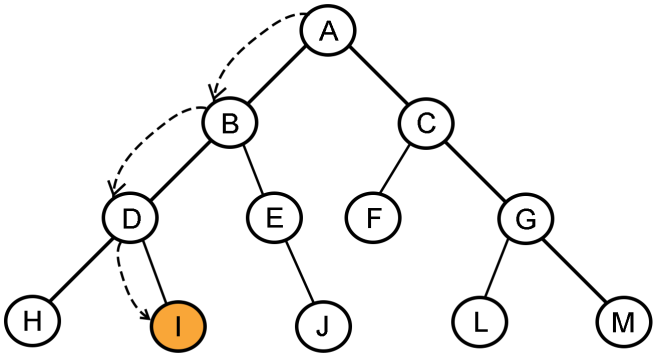


Ουσιαστικά για την αναζήτησή της, η μέθοδος καλύτερου τοπικού ορίου βλέπουμε ότι θα ξεκινήσει από την ρίζα A και θα βλέπει ποιο όριο των επόμενων κόμβων είναι καλύτερο, δηλαδή έχει καλύτερη τιμή για να μπορεί να επιλέξει, και επιλέγει αυτόν. Έπειτα, θα αντικαταστήσει την τιμή του ορίου του A με την τιμή του ορίου του κόμβου που θα επιλέξει και στη συνέχεια θα επαναλαμβάνει την ίδια διαδικασία μέχρι κάποιος κόμβος να μην έχει παιδιά για να ακολουθήσει η διαδικασία της οπισθοχώρησης. Άρα, την καλύτερη τιμή του ορίου για το Σχήμα 3.3 την βρίσκει με το μονοπάτι A->C->G->L.

3.3.2 Αλγόριθμος καλύτερης ευρετικής προβολής (Best Projection Heuristic)

Για τον αλγόριθμο αναζήτησης καλύτερης ευρετικής προβολής για προβλήματα μικτού ακέραιου προγραμματισμού σε δένδρο διακλάδωσης και δέσμευσης, η λειτουργία του είναι να διαλέγει το ενεργό υποπρόβλημα και να επιστρέφει την καλύτερη προβολή κόμβου. Η λειτουργία του αρχικά ξεκινάει δηλώνοντας δύο δείκτες, ο ένας root και ο άλλος node, έπειτα προσθέτει δείκτη στη ρίζα με τον αριθμό ένα και στη συνέχεια υπολογίζει την υποβάθμιση της αντικειμενικής συνάρτησης ανά μονάδα του αθροίσματος των ακεραίων ανέφικτων αριθμών. Αυτό υπολογίζεται αφαιρώντας την αντικειμενική τιμή της συνάρτησης με την τιμή του ορίου του κόμβου της ρίζας και έπειτα διαιρώντας το αποτέλεσμα αυτό με τον αριθμό του αθροίσματος των ακέραιων μη εφικτών αριθμών, αποθηκεύοντας το αποτέλεσμα σε μια μεταβλητή deg. Ακόμη, δίνει στη μεταβλητή best τον μέγιστο αριθμό και εισέργεται στη λίστα των ενεργών υποπροβλημάτων. Έπειτα για όλους τους κόμβους με τη σειρά αποθηκεύει το γινόμενο του αθροίσματος του ορίου που βρίσκεται σε εκείνο τον κόμβο με το deg με το άθροισμα των ακέραιων μη εφικτών αριθμών και το αποθηκεύει σε μια μεταβλητή obj. Έπειτα επιλέγει το υποπρόβλημα που έχει την καλύτερη εκτιμώμενη βέλτιστη αντικειμενική λύση και επιστρέφει τον κόμβο αυτού που είναι ο καλύτερος. Η μέθοδος αναζήτησης έχει τη μορφή του Σχήματος 3.4.

Σχήμα 3.4: Μέθοδος αναζήτησης καλύτερης ευρετικής προβολής (Best Projection Heuristic)



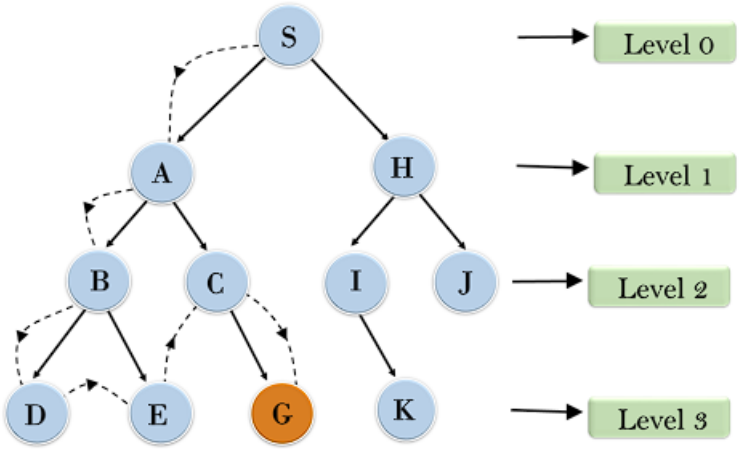
Ουσιαστικά, αυτή η αναζήτηση όπως βλέπουμε ξεκινάει από την ρίζα A και προσπαθεί να βρει τη βέλτιστη αντικειμενική τιμή και να την επιστρέψει. Άρα η καλύτερη τιμή βρίσκεται από το μονοπάτι A->B->D->I την οποία θα την επιστρέψει

μαζί με τον κόμβο στον οποίο βρίσκεται.

3.3.3 Αλγόριθμος αναζήτησης πρώτα κατά βάθος (Depth first search)

Σε αυτήν την υποενότητα θα δούμε τον τρίτο αλγόριθμο αναζήτησης που χρησιμοποιείται για αναζήτηση στο δένδρο αναζήτησης για προβλήματα, όπου είναι ο αλγόριθμος κατά βάθος. Η λειτουργία του αλγορίθμου κατά βάθος, είναι της μορφής όπως μπορούμε να δούμε στο Σχήμα 3.5 [69].

Σχήμα 3.5: Μέθοδος αναζήτησης πρώτα κατά βάθος (Depth First Search)



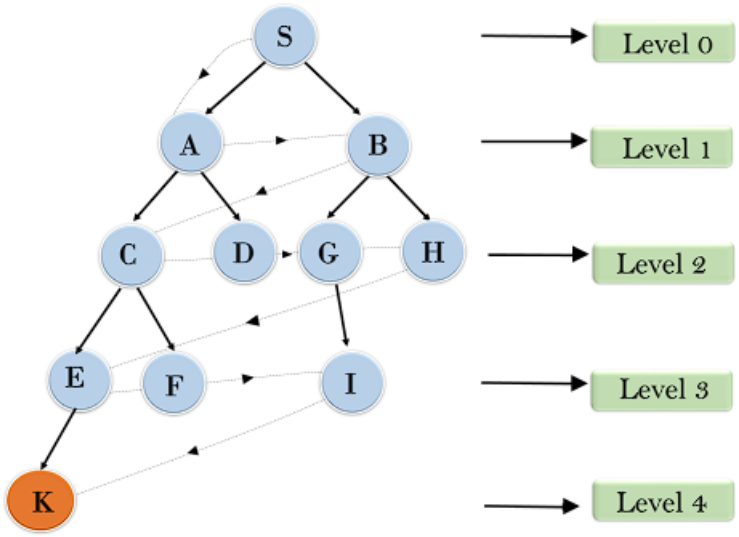
Ουσιαστικά, η αναζήτηση κατά βάθος χρησιμοποιείται για την αναζήτηση μιας δομής δεδομένων ενός δένδρου, για έναν κόμβο που θα ικανοποιήσει μια δεδομένη ισότητα και θα επιστρέψει την καλύτερη λύση, δηλαδή την καλύτερη τιμή. Η διαδικασία αυτή, όπως βλέπουμε στην εικόνα παραπάνω, αφού πρώτα έχει επιλέξει ένα ενεργό υποπρόβλημα από τη λίστα ενεργών υποπροβλημάτων για το μικτό ακέραιο πρόβλημα, ξεκινάει από τη ρίζα δηλαδή το S ενός δένδρου όπως βλέπουμε στο σχήμα και εξερευνά όλους τους κόμβους του παρόντος βάθους πριν προχωρήσει σε επόμενους κόμβους στο επόμενο βάθος. Αυτή η αναζήτηση, χρησιμοποιεί συνήθως μια ουρά για την παρακολούθηση των θυγατρικών κόμβων που συναντιούνται αλλά δεν εξερευνήθηκαν ακόμη. Ουσιαστικά η διαδικασία που θα ακολουθήσει για την αναζήτηση των κόμβων στο δένδρο της παραπάνω εικόνας είναι η S->A->B->D->E->C->G->H->I->K->J. Η διαδρομή όταν θα φτάσει στο E και δει ότι δεν μπορεί να προχωρήσει άλλο προς τα κάτω σε άλλους κόμβους γιατί δεν υπάρχουν θα πραγματοποιήσει μία οπισθοχώρηση (backtracking), δηλαδή θα πάει προς τα πίσω στο

δένδρο. Έπειτα, θα διασχίσει τον κόμβο C και μετά G και μετά θα συνεχίσει κρατώντας πάντα την καλύτερη τιμή που έχει βρει μέχρι εκείνη τη χρονική στιγμή. Τέλος, εάν τερματιστεί το πρόγραμμα θα παρουσιάσει την καλύτερη τιμή ως καλύτερη λύση που βρήκε, όπου για το παράδειγμά μας είναι η τιμή του G.

3.3.4 Αλγόριθμος αναζήτησης πρώτα κατά πλάτος (Breadth first search)

Ο τέταρτος και τελευταίος αλγόριθμος που χρησιμοποιείται για την αναζήτηση και θα δούμε τη λειτουργία του σε αυτήν την υποενότητα είναι ο αλγόριθμος αναζήτησης πρώτα κατά βάθος. Η αναζήτηση κατά πλάτος είναι μία πολύ γνωστή μέθοδος αναζήτησης που χρησιμοποιείται για την αναζήτηση μιας δομής δεδομένων ενός δένδρου. τη μορφή αυτής της αναζήτησης μπορούμε να τη δούμε στο Σχήμα 3.6 [69].

Σχήμα 3.6: Μέθοδος αναζήτησης πρώτα κατά πλάτος (Breadth First Search)



Η λειτουργία του στα προβλήματα μικτού ακέραιου προγραμματισμού είναι η ίδια λειτουργία όπως και στα υπόλοιπα προβλήματα. Για το μικτό ακέραιο πρόβλημα ξεκινάει επιλέγοντας ένα υποπρόβλημα από τη λίστα ενεργών υποπροβλημάτων και στη συνέχεια εκτελεί αναζήτηση κατά πλάτος. Ουσιαστικά ξεκινάει από το S και συνεχίζει με την εξής σειρά σύμφωνα με τους κόμβους του παραπάνω σχήματος: S->A->B->C->D->G->H->E->F->I->K. Ουσιαστικά αυτή η αναζήτηση ελέγχει πρώτα όλα τα παιδιά της ρίζας ξεκινώντας πάντα από τα αριστερά προς τα δεξιά, δηλαδή πρώτα ελέγχει όλους τους κόμβους του βάθους που βρίσκεται, και έπειτα προχωράει στο επόμενο βάθος ελέγχοντας κατ'επανάληψη την ίδια διαδικασία ξεκινώντας από τα αριστερά. Από κόμβο σε κόμβο ελέγχει για την καλύτερη τιμή, αν τη βρει, την κρατάει και συνεχίζει τη διαδικασία μέχρι να τελειώσει την αναζήτηση ή μέχρι να σταματήσει η προγραμματισμένη χρονική διαδικασία αναζήτησης από

το πρόγραμμα. Στην περίπτωση του παραπάνω σχήματος βλέπουμε ότι θα εκτελέσει όλη τη διαδικασία δεδομένου ότι το δένδρο είναι μικρό και θα καταλήξει όπως βλέπουμε ότι το K περιέχει την καλύτερη τιμή οπότε θα την επιστρέψει.

Κεφάλαιο 4

Αποτελέσματα αλγορίθμων και σύγκριση

Σε αυτό το κεφάλαιο θα δούμε και θα αναλύσουμε τα αποτελέσματα που παράχθηκαν μέσω των κωδίκων που χρησιμοποιήσαμε για την επίλυση των προβλημάτων που εισάγαμε. Η διαδικασία εκτέλεσης των κωδίκων αυτών πραγματοποιήθηκε στο λειτουργικό σύστημα Ubuntu και για τη δημιουργία τους χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++. Συνολικά επιλύθηκαν 111 προβλήματα, τα οποία επιλέχθηκαν ανάμεσα σε μονοδιάστατα προβλήματα των:

- ANI_AI_CSP (6 προβλήματα)
- Falkenauer_CSP (33 προβλήματα)
- Hard28_CSP (3 προβλήματα)
- Irnich_CSP (2 προβλήματα)
- Randomly_Generated_CSP (7 προβλήματα)
- Scholl_CSP (43 προβλήματα)
- Schwerin_CSP (7 προβλήματα)
- Waescher_CSP (10 προβλήματα)

Τα προβλήματα είναι διαφορετικών μεγεθών ως προς τα μήκη (length), ζήτησης και απαιτούμενων κομματιών παραγγελίας πελάτη αλλά και διαφορετικό σταθερό στοκ της επιχείρησης που έχει στο απόθεμα για κάθε ένα πρόβλημα. Τα προβλήματα αυτά υπάρχουν και μπορούμε να τα βρούμε στο http://or.dei.unibo.it/library/bpplib. Ο ακριβής αλγόριθμος εκτελέστηκε τέσσερις διαφορετικές φορές για όλα τα αποτελέσματα καθώς χρησιμοποιθηκαν τέσσερις διαφορετικοί τρόποι αναζήτησης. Για

την επίλυση των αρχείων των προβλημάτων χρησιμοποιήθηκε ένα όριο χρόνου (time limit) μιας ώρας, για την παραγωγή του αποτελέσματος. Δηλαδή, αν ο αλγόριθμος για παράδειγμα δεν επίλυε το πρόβλημα εντός του χρονικού ορίου, τότε έβγαζε ως αποτέλεσμα τη λύση που είχε μέχρι το διάστημα της χρονικής στιγμής που εκτελέστηκε, αν βέβαια παράχθηκε λύση.

Ο πρώτος πίνακας που θα δούμε περιέχει τα δεδομένα των προβλημάτων, ο δεύτερος περιέχει τα αποτελέσματα των στοκ που παράχθηκαν για να ικανοποιηθεί το κάθε πρόβλημα και ο τρίτος τα αποτελέσματα των χρόνων που η κάθε μέθοδος χρησιμοποίησε για να πετύχει την επίλυση του προβλήματος. Για τον δεύτερο και τρίτο πίνακα θα ακολουθήσει και η σύγκριση των αποτελεσμάτων, για να δούμε ποιος τελικά είναι πιο αποδοτικός αλγόριθμος ως προς την παραγωγή και τον χρόνο.

Παρακάτω βλέπουμε τον Πίνακα 4.1 που περιέχει τα δεδομένα του προβλήματος. Περιέχει τέσσερις στήλες, όπου η πρώτη στήλη δηλαδή το ''No'' αντιστοιχεί στο νούμερο του προβλήματος, στη δεύτερη το όνομα του προβλήματος που αντιστοιχεί σε πιο πρόβλημα επιλύουμε, στην τρίτη ο αριθμός απαιτούμενων μηκών, όπου αντιστοιχεί στον αριθμό παραγγελίας μηκών του πελάτη και στην τέταρτη το σταθερό στοκ, όπου αντιστοιχεί στο σταθερό μήκος στοκ που έχει η επιχείρηση.

Πίνακας 4.1: Πίνακας Δεδομένων

No	**Όνομα Προβλήματος**	**Αριθμός Απαιτούμενων Στοκ**	**Σταθερό Στοκ**
1	201_2500_DI_21	175	2456
2	201_2500_DI_24	171	2032
3	201_2500_DI_25	177	2464
4	201_2500_DI_26	170	2068
5	201_2500_DI_27	168	1940
6	201_2500_DI_28	167	1956
7	BPP_200_125_0.1_0.7_5	71	125
8	BPP_200_125_0.2_0.8_3	70	125
9	BPP_50_50_0.1_0.7_0	23	50
10	BPP_50_50_0.1_0.7_2	26	50
11	BPP_50_50_0.1_0.7_5	24	50
12	BPP_50_50_0.1_0.7_6	25	50
13	BPP_50_50_0.1_0.7_8	25	50
14	csAA125_1	125	500000
15	csAA125_3	125	500000
16	Falkenauer_t120_01	85	1000
17	Falkenauer_t120_06	86	1000
18	Falkenauer_t120_08	86	1000
19	Falkenauer_t120_09	85	1000
20	Falkenauer_t60_06	51	1000
21	Falkenauer_t60_08	49	1000
			Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.1 – Συνέχεια της προηγούμενης Σελίδας

No	Όνομα Προβλήματος	Αριθμός Απαιτούμενων Στοκ	Σταθερό Στοκ
22	Falkenauer_t60_10	53	1000
23	Falkenauer_t60_12	53	1000
24	Falkenauer_t60_14	50	1000
25	Falkenauer_t60_17	48	1000
26	Falkenauer_t60_18	50	1000
27	Falkenauer_u1000_09	81	150
28	Falkenauer_u1000_10	81	150
29	Falkenauer_u1000_13	81	150
30	Falkenauer_u1000_17	85	150
31	Falkenauer_u120_03	68	150
32	Falkenauer_u120_05	61	150
33	Falkenauer_u120_07	64	150
34	Falkenauer_u120_08	67	150
35	Falkenauer_u120_10	64	150
36	Falkenauer_u120_12	63	150
37	Falkenauer_u120_13	62	150
38	Falkenauer_u120_14	61	150
39	Falkenauer_u120_18	66	150
40	Falkenauer_u500_04	80	150
41	Falkenauer_u500_07	80	150
42	Falkenauer_u500_09	81	150
43	Falkenauer_u500_10	81	150
44	Falkenauer_u500_11	81	150
45	Falkenauer_u500_14	81	150
46	Falkenauer_u500_15	80	150
47	Falkenauer_u500_17	81	150
48	Falkenauer_u500_19	81	150
49	HARD0	199	100000

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.1 – Συνέχεια της προηγούμενης Σελίδας

No	Όνομα Προβλήματος	Αριθμός Απαιτούμενων Στοκ	Σταθερό Στοκ
50	Hard28_BPP181	157	1000
51	Hard28_BPP359	164	1000
52	Hard28_BPP531	175	1000
53	HARD3	197	100000
54	HARD6	199	100000
55	HARD7	200	100000
56	N1C1W1_B	37	100
57	N1C1W1_C	41	100
58	N1C1W1_E	40	100
59	N1C1W1_G	41	100
60	N1C2W1_R	42	120
61	N1W1B1R0	41	1000
62	N1W1B1R2	42	1000
63	N1W1B1R3	40	1000
64	N1W1B1R5	41	1000
65	N1W1B1R7	45	1000
66	N1W1B1R9	41	1000
67	N1W1B2R1	45	1000
68	N1W1B2R6	49	1000
69	N1W1B3R0	48	1000
70	N1W1B3R6	48	1000
71	N1W1B3R9	49	1000
72	N1W2B1R0	35	1000
73	N1W2B1R1	35	1000
74	N1W2B1R2	38	1000
75	N1W2B1R3	38	1000
76	N1W2B1R4	34	1000
77	N1W2B1R5	40	1000

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.1 – Συνέχεια της προηγούμενης Σελίδας

No	Όνομα Προβλήματος	Αριθμός Απαιτούμενων Στοκ	Σταθερό Στοκ
78	N1W2B1R6	38	1000
79	N1W2B1R7	42	1000
80	N1W2B1R8	31	1000
81	N1W2B1R9	37	1000
82	N1W2B2R0	46	1000
83	N1W2B2R1	43	1000
84	N1W2B2R2	48	1000
85	N1W2B2R3	45	1000
86	N1W2B2R4	44	1000
87	N1W2B2R5	47	1000
88	N1W2B2R6	47	1000
89	N1W2B2R7	43	1000
90	N1W2B3R1	45	1000
91	N1W2B3R3	46	1000
92	N1W2B3R6	44	1000
93	N1W2B3R9	47	1000
94	N2W1B3R8	93	1000
95	Schwerin1_BPP1	46	1000
96	Schwerin1_BPP10	41	1000
97	Schwerin1_BPP3	44	1000
98	Schwerin1_BPP4	42	1000
99	Schwerin1_BPP5	44	1000
100	Schwerin1_BPP8	44	1000
101	Schwerin2_BPP26	44	1000
102	Waescher_TEST0005	57	10000
103	Waescher_TEST0014	47	10000
104	Waescher_TEST0022	33	10000
105	Waescher_TEST0044	56	10000

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.1 – Συνέχεια της προηγούμενης Σελίδας

No	Όνομα Προβλήματος	Αριθμός Απαιτούμενων Στοκ	Σταθερό Στοκ
106	Waescher_TEST0049	43	10000
107	Waescher_TEST0054	56	10000
108	Waescher_TEST0055A	52	10000
109	Waescher_TEST0065	35	10000
110	Waescher_TEST0082	48	10000
111	Waescher_TEST0095	63	10000

Ο Πίνακας 4.2 περιέχει τα αποτελέσματα παραγωγής στοκ, δηλαδή στοκ σταθερού μήκους που παράχθηκαν από τη επιχείρηση για να καλύψει την παραγγελία του πελάτη. Περιέχει έξι στήλες, όπου η πρώτη στήλη αντιστοιχεί στον αριθμό του προβλήματος που είδαμε στον πίνακα των δεδομένων, η δεύτερη αντιστοιχεί στο αποτέλεσμα της λύσης του αλγορίθμου γραμμικής χαλάρωσης (LINEAR) και οι υπόλοιπες τέσσερις, στις διαφορετικές μεθόδους αναζήτησης που ο ακριβής αλγόριθμος επίλυσε το πρόβλημα. Η τρίτη στήλη ουσιαστικά αντιστοιχεί στη μέθοδο καλύτερου τοπικού ορίου (BLB), η τέταρτη στην καλύτερη ευρετική προβολή (BPH), η πέμπτη στην αναζήτηση κατά βάθος (DFS) και η έκτη στην κατά βάθος (BFS).

Πίνακας 4.2: Πίνακας Αποτελεσμάτων Παραγωγής Στοκ

No	**LINEAR**	**BLB**	**BPH**	**DFS**	**BFS**
1	65	71	–	–	69
2	65	–	–	–	69
3	65	68	68	68	68
4	65	71	–	–	70
5	65	69	68	70	68
6	65	71	–	72	70
7	80.224	84	84	84	84
8	102.5	109	111	–	111
9	23	24	–	–	23
10	20.5	22	22	23	21
11	18.32	19	19	19	19
12	19.64	21	21	21	20
13	21.36	22	22	–	22
14	526.09	–	–	–	–
15	521.23	–	–	–	–
16	40	45	–	–	45
17	40	–	45	–	45
18	40	–	–	–	46

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.2 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR	BLB	BPH	DFS	BFS
19	40	–	–	–	46
20	20	–	–	–	22
21	20	21	21	22	22
22	20	–	–	–	22
23	20	21	–	–	23
24	20	22	–	–	22
25	20	21	–	–	22
26	20	22	–	–	22
27	396.92	–	–	–	–
28	399.34	–	–	–	–
29	395.27	–	–	–	–
30	403.8	–	–	–	–
31	48.62	50	–	–	51
32	47.48	49	49	50	49
33	48.65	51	51	53	51
34	49.91	52	–	–	52
35	51.28	–	–	–	54
36	47.86	–	–	–	50
37	48.01	50	–	–	50
38	49.16	51	51	52	51
39	48.38	–	–	–	51
40	205.11	214	–	–	–
41	203.98	–	–	–	–
42	201.06	–	210	–	210
43	199.06	–	209	209	209
44	199.42	–	–	–	–
45	203.02	–	224	–	–
46	200.13	–	–	–	–

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.2 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR	BLB	BPH	DFS	BFS
47	197.42	–	–	–	–
48	195.63	206	–	–	206
49	55.00	60	60	–	59
50	71.99	82	–	–	77
51	74.99	82	81	83	82
52	83	90	–	–	–
53	54.92	59	59	–	60
54	56.09	60	–	–	60
55	54.24	59	59	–	60
56	30.5	31	–	–	31
57	20	21	21	–	21
58	26	–	30	29	26
59	25	28	28	28	26
60	23	–	–	–	23
61	17.41	19	19	20	19
62	18.30	–	19	20	19
63	17.20	19	19	19	18
64	16.66	18	–	–	18
65	16.66	18	18	19	18
66	16.66	18	18	18	18
67	16.34	18	18	–	17
68	16.25	17	17	18	17
69	16.14	17	17	17	17
70	15.04	16	16	19	16
71	16.40	17	17	17	17
72	10.43	11	11	–	11
73	10.28	11	11	12	11
74	10.03	11	11	11	11

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.2 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR	BLB	BPH	DFS	BFS
75	10.39	11	11	11	11
76	10.53	11	–	12	11
77	9.88	10	10	11	11
78	10.06	11	11	11	11
79	10.06	11	11	–	11
80	9.95	11	11	11	11
81	10.09	11	11	11	11
82	9.77	10	10	11	10
83	10.4	11	11	12	11
84	9.23	10	10	10	10
85	10.08	11	11	–	11
86	9.74	10	10	10	10
87	9.90	10	10	11	10
88	9.51	10	10	10	10
89	9.75	10	10	10	10
90	10.14	11	11	11	11
91	10.15	11	11	11	11
92	10.19	11	11	11	11
93	9.552	10	10	10	10
94	33.46	36	37	–	35
95	17.53	19	19	–	19
96	17.56	–	–	–	19
97	17.60	19	19	19	19
98	17.74	–	–	–	19
99	17.59	19	19	19	19
100	17.80	19	19	19	19
101	21.22	22	23	23	22
102	27.99	–	–	–	29

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.2 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR	BLB	BPH	DFS	BFS
103	22.99	24	24	24	24
104	13.99	–	–	–	15
105	13.99	15	15	15	15
106	10.99	12	12	12	12
107	13.99	–	–	–	15
108	14.99	16	16	–	16
109	14.99	16	16	16	16
110	23.98	25	25	–	25
111	15.99	17	17	–	17
Average	**73.55**	**36.89**	**37.29**	**29.23**	**38.61**
Common Av	**20.33**	**24.26**	**24.65**	**24.89**	**23.93**

Στον πίνακα των αποτελεσμάτων για τα στοκ που παράχθηκαν παρατηρούμε ότι, πολλά από τα αποτελέσματα δεν είχαν λύση και συγκεκριμένα είναι τα αποτελέσματα που έχουν ως λύση την παύλα (-). Παρατηρούμε επίσης, ότι ο LINEAR παρήγαγε λύση για όλα τα προβλήματα, ωστόσο δε χρησιμοποιείται σε πραγματικά προβλήματα. Για τα αποτελέσματα του ακριβή αλγόριθμου με τις τέσσερις διαφορετικές μεθόδους αναζήτησης στο χρονικό διάστημα της μίας ώρας παρατηρούμε ότι:

- Η μέθοδος BLB έδωσε λύση σε 81 προβλήματα.
- Η μέθοδος BPH έδωσε λύση σε 77 προβλήματα.
- Η μέθοδος DFS έδωσε λύση σε 53 προβλήματα.
- Η μέθοδος BFS έδωσε λύση σε 97 προβλήματα.

Ωστόσο, μπορούμε να διακρίνουμε και τις δύο τελευταίες σειρές του πίνακα όπου έχουν υπολογιστεί και οι μέσοι όροι των προβλημάτων συνολικά. Η σειρά που περιέχει το Average μας δείχνει, τον μέσο όρο των αποτελεσμάτων των προβλημάτων

που έχουν λύση με κάθε μία μέθοδο. Ο LINEAR έχει τον καλύτερο μέσο όρο με 73.55 με επίλυση όλων των προβλημάτων. Ο BLB έχει περίπου 36.89 με παραγωγή σε 81 προβλήματα έναντι 111, ο BPH έχει περίπου 37.29 με 77 έναντι 111, ο BFS έχει περίπου 38.61 με 97 σε 111 και ο DFS έλυσε μόνο 53 προβλήματα έναντι 111 με μέσο όρο περίπου 29.23. Συνεπώς, παρατηρούμε ότι σταδιακά στα προβλήματα ο BFS έχει καλύτερο μέσο όρο σε σχέση με τους άλλους τρεις, καθώς ο μέσος όρος του κυμαίνεται κοντά στων υπολοίπων και παράχθηκε λύση σε πολύ περισσότερα προβλήματα από ότι οι υπόλοιποι. Ακόμη, υπήρξαν και δέκα προβλήματα, στα οποία καμία μέθοδος δεν βρήκε λύση στο διάστημα της μίας ώρας και είναι τα προβλήματα (14,15,27,28,29,30,41,44,46,47). Ο λόγος για τον οποίο καμία μέθοδος δεν απέδωσε σε αυτά τα προβλήματα μπορεί να είναι είτε γιατί τα προβλήματα αυτά ήταν πολύ μεγάλα σε αριθμό μηκών, όπως είναι τα προβλήματα (14,15), είτε διότι ο αριθμός των απαιτούμενων κομματιών για τα μήκη ήταν μεγάλος όπως στα προβλήματα (27,28,29,30,41,44,46,47). Συνεπώς, βλέπουμε ότι η σημασία του χρονικού διαστήματος είναι πολύ σημαντική για την παραγωγή λύσης, καθώς στην περίπτωση που το χρονικό διάστημα θα ήταν μεγαλύτερο, πολύ πιθανόν οι μέθοδοι αυτοί να απέδιδαν καλύτερα και ως προς μία καλύτερη λύση στα προβλήματα που επίλυσαν αλλά και στα προβλήματα που δεν παρείχαν λύση.

Ακόμη, υπολογίστηκε και ο μέσος όρος της παραγωγής λύσεων για τα κοινά τους προβλήματα, όπου βλέπουμε τη σειρά Common Av. Θεωρούμε ότι η καλύτερη λύση στα προβλήματα αυτά προστίθεται σε όποια μέθοδο έχει την καλύτερη λύση, δηλαδή αν σε ένα πρόβλημα ο BLB έχει λύση 18, ο BPH έχει λύση 19, ο DFS έχει λύση 20 και ο BFS έχει λύση 18, τότε για τον BLP και το BFS μετράει ότι βρήκαν την καλύτερη λύση για το πρόβλημα. Στα κοινά τους προβλήματα αυτά, τα οποία σε αριθμό είναι 46, παρατηρούμε ότι η καλύτερη μέθοδος είναι αυτής του BFS με περίπου 23.93 μέσο όρο και καλύτερη λύση στα 44 από τα 46 προβλήματα. Ακολουθεί ο BLB με 24.26 και καλύτερη λύση σε 39 από 46, ο BPH με 24.65 και καλύτερη λύση σε 40 από 46, και τέλος ο DFS με 24.89 και καλύτερη λύση σε 26 από τα 46. Άρα, και στα κοινά προβλήματα τους, βλέπουμε ότι ο μέσος όρος του BFS είναι ο καλύτερος.

Συνεπώς, για τον ακριβή αλγόριθμο, η μέθοδος πρώτα κατά πλάτος επίλυσε τον μεγαλύτερο αριθμό προβλημάτων σε σχέση με τις υπόλοιπες μεθόδους, άρα

καταλήγουμε στο συμπέρασμα ότι είναι ο πιο αποδοτικός για την παραγωγή αποτελέσματος και ο λιγότερος αποδοτικός φαίνεται ότι είναι η μέθοδος κατά βάθος.

Ο Πίνακας 4.3 περιέχει τα αποτελέσματα των χρόνων όπου, η κάθε μέθοδος χρειάστηκε για την παραγωγή αποτελεσμάτων και τη μη παραγωγή αποτελεσμάτων. Ωστόσο, παρά το γεγονός ότι σε κάποια προβλήματα, κάποιες μέθοδοι δεν έβγαλαν αποτέλεσμα στο χρονικό όριο που δόθηκε ο χρόνος έχει κρατηθεί στο μέγιστο, δηλαδή ο χρόνος μετράει για τη σύγκριση των μεθόδων. Ακόμη, ο πίνακας περιέχει και αυτός έξι στήλες, μία για κάθε μία μέθοδο, ακριβώς όπως και ο πίνακας παραγωγής αποτελεσμάτων. Ο πίνακας των αποτελεσμάτων χρόνου λοιπόν έχει την πρώτη στήλη, όπου αντιστοιχεί στον αριθμό του προβλήματος που είδαμε στον πίνακα των δεδομένων, η δεύτερη αντιστοιχεί που αντιστοιχεί στον χρόνο επίλυσης του αλγορίθμου γραμμικής χαλάρωσης (LINEAR_T) και οι υπόλοιπες τέσσερις στις διαφορετικές μεθόδους αναζήτησης που ο ακριβής αλγόριθμος επίλυσε το πρόβλημα. Η τρίτη στήλη ουσιαστικά αντιστοιχεί στον χρόνο της μεθόδου καλύτερου τοπικού ορίου (BLB_T), η τέταρτη στην καλύτερη ευρετική προβολή(BPH_T), η πέμπτη στην αναζήτηση κατά βάθος(DFS_T) και η έκτη στην κατά βάθος(BFS_T).

Πίνακας 4.3: Πίνακας Αποτελεσμάτων Χρόνου

No	**LINEAR_T**	**BLB_T**	**BPH_T**	**DFS_T**	**BFS_T**
1	11.59	3,600	3,600	3,600	3,600
2	9.66	3,600	3,600	3,600	3,600
3	10.49	3,600	3,600	3,600	3,600
4	9.829	3,600	3,600	3,600	3,600
5	8.74	3,600	3,600	3,600	3,600
6	8.23	3,600	3,600	3,600	3,600
7	0.08	3,600	3,600	3,600	3,600
8	0.06	3,600	3,600	3,600	3,600
9	0.005	3,600	3,600	3,600	3,600
10	0.00	3,600	3,600	3,600	3,600
11	0.005	1.083	1.06	1,002.88	0.80
12	0.006	3,600	3,600	3,600	22.92
13	0.007	0.26	0.27	3,600	2.95
Συνεχίζεται στην επόμενη σελίδα					

Πίνακας 4.3 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR_T	BLB_T	BPH_T	DFS_T	BFS_T
14	504.94	3,600	3,600	3,600	3,600
15	582.53	3,600	3,600	3,600	3,600
16	0.72	3,600	3,600	3,600	3,600
17	0.6287	3,600	3,600	3,600	3,600
18	0.61	3,600	3,600	3,600	3,600
19	0.65	3,600	3,600	3,600	3,600
20	0.20	3,600	3,600	3,600	3,600
21	0.23	3,600	3,600	3,600	3,600
22	0.25	3,600	3,600	3,600	3,600
23	0.28	3,600	3,600	3,600	3,600
24	0.22	3,600	3,600	3,600	3,600
25	0.20	3,600	3,600	3,600	3,600
26	0.21	3,600	3,600	3,600	3,600
27	0.10	3,600	3,600	3,600	3,600
28	0.10	3,600	3,600	3,600	3,600
29	0.10	3,600	3,600	3,600	3,600
30	0.10	3,600	3,600	3,600	3,600
31	0.10	3,600	3,600	3,600	3,600
32	0.07	3,600	3,600	3,600	3,600
33	0.08	3,600	3,600	3,600	3,600
34	0.10	3,600	3,600	3,600	3,600
35	0.09	3,600	3,600	3,600	3,600
36	0.06	3,600	3,600	3,600	3,600
37	0.07	3,600	3,600	3,600	3,600
38	0.08	3,600	3,600	3,600	3,600
39	0.09	3,600	3,600	3,600	3,600
40	0.10	3,600	3,600	3,600	3,600
41	0.11	3,600	3,600	3,600	3,600

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.3 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR_T	BLB_T	BPH_T	DFS_T	BFS_T
42	0.11	3,600	3,600	3,600	3,600
43	0.10	3,600	3,600	3,600	3,600
44	0.11	3,600	3,600	3,600	3,600
45	0.10	3,600	3,600	3,600	3,600
46	0.11	3,600	3,600	3,600	3,600
47	0.12	3,600	3,600	3,600	3,600
48	0.09	3,600	3,600	3,600	3,600
49	421,00	3,600	3,600	3,600	3,600
50	3.27	3,600	3,600	3,600	3,600
51	2.92	3,600	3,600	3,600	3,600
52	3.37	3,600	3,600	3,600	3,600
53	422.93	3,600	3,600	3,600	3,600
54	387.51	3,600	3,600	3,600	3,600
55	419.31	3,600	3,600	3,600	3,600
56	0.012	3,600	3,600	3,600	3,600
57	0.02	3,600	3,600	3,600	3,600
58	0.02	3,600	3,600	3,600	3,600
59	0.02	3,600	3,600	3,600	3,600
60	0.02	3,600	3,600	3,600	3,600
61	0.13	3,600	3,600	3,600	3,600
62	0.11	3,600	3,600	3,600	3,600
63	0.15	3,600	3,600	3,600	3,600
64	0.11	3,600	3,600	3,600	3,600
65	0.19	3,600	3,600	3,600	3,600
66	0.11	3,600	3,600	3,600	3,600
67	0.16	3,600	3,600	3,600	531.94
68	0.25	69.40	65.54	3,600	87.99
69	0.28	0.96	0.95	3,142.72	2.34

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.3 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR_T	BLB_T	BPH_T	DFS_T	BFS_T
70	0.26	0.53	1.43	3,600	0.97
71	0.30	6.86	7.18	49.99	24.42
72	0.07	1,796	1385.61	3,600	5.08
73	0.10	1.40	1.39	3,600	5.98
74	0.10	0.88	0.89	25.00	2.11
75	0.08	0.55	0.63	8.55	6.93
76	0.08	10.36	3,600	3,600	11.90
77	0.11	426.67	384.83	3,600	3,600
78	0.09	1.11	1.17	30.66	0.97
79	0.11	2.05	2.01	3,600	0.71
80	0.06	3,600	3,600	3,600	3,600
81	0.10	0.33	0.34	9,81	0.50
82	0.15	485.93	449.60	3,600	1.23
83	0.12	0.50	0.50	3,600	1.93
84	0.17	0.62	0.63	2.12	4.92
85	0.13	1.15	1.13	3,600	2.81
86	0.15	0.11	0.12	0.13	0.12
87	0.16	1.22	1.29	3,600	26.14
88	0.15	0.15	0.16	0.17	0.16
89	0.14	0.62	0.61	2.78	1.03
90	0.17	34.25	33.73	2.52	0.28
91	0.17	1.45	1.33	1.26	1.76
92	0.18	0.38	0.35	510.26	1.16
93	0.20	0.56	0.57	1.34	3.33
94	0.97	3,600	3,600	3,600	3,600
95	0.11	3,600	3,600	3,600	3,600
96	0.08	3,600	3,600	3,600	3,600
97	0.11	3,600	3,600	3,600	3,600

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 4.3 – Συνέχεια της προηγούμενης Σελίδας

No	LINEAR_T	BLB_T	BPH_T	DFS_T	BFS_T
98	0.09	3,600	3,600	3,600	3,600
99	0.10	3,600	3,600	3,600	3,600
100	0.10	3,600	3,600	3,600	3,600
101	0.10	3,600	3,600	3,600	3,600
102	5.47	3,600	3,600	3,600	3,600
103	2.90	3,600	3,600	3,600	3,600
104	1.42	3,600	3,600	3,600	3,600
105	2.89	3,600	3,600	3,600	3,600
106	1.33	3,600	3,600	3,600	3,600
107	2.12	3,600	3,600	3,600	3,600
108	2.22	3,600	3,600	3,600	3,600
109	1.87	3,600	3,600	3,600	3,600
110	2.27	3,600	3,600	3,600	3,600
111	3.10	3,600	3,600	3,600	3,600
Average T	**33.53**	**2,861.97**	**2,898.85**	**3,307.99**	**2,794.24**
Common A.T	**20.33**	**24.26**	**24.65**	**24.45**	**23.93**

Για τους χρόνους που βλέπουμε στον πίνακα των χρονικών αποτελεσμάτων παρατηρούμε ότι, όλοι οι μέθοδοι για τα περισσότερα προβλήματα χρειάστηκαν όλο το χρονικό όριο που τους δόθηκε εκτός του γραμμικού. Πιο συγκεκριμένα, ο BLB χρησιμοποίησε ολόκληρο το χρονικό όριο σε 85 προβλήματα από τα 111 συνολικά, το ίδιο και ο BPH, ενώ ο DFS σε 86 και ο BFS σε 74. Ακόμη, υπολογίστηκε και ο μέσος όρος των χρόνων για τις μεθόδους αυτές, όπως μπορούμε να δούμε στην τελευταία σειρά στο Average T. Παρατηρούμε ότι, για τον LINEAR ο μέσος χρόνος επίλυσης όλων των προβλημάτων είναι 33. 53 περίπου, για τον BLB είναι 2,861.97(47 λεπτά περίπου), για τον BPH είναι 2,898.85 (48 λεπτά περίπου), για τον DFS είναι 3,307.99 (55 λεπτά περίπου) και για τον BFS είναι 2,794.24 (46 λεπτά περίπου). Άρα, τον καλύτερο μέσο χρόνο επίλυσης των προβλημάτων τον έχει ο BFS με 46 λεπτά περίπου, μετά ακολουθεί ο BLB με 47 λεπτά, έπειτα ο BPH με 48 λεπτά και

τέλος ο DFS με 55 λεπτά. Ωστόσο, υπολογίστηκε και ο χρόνος επίλυσης των κοινών τους προβλημάτων, δηλαδή τα 46 προβλήματα που αναφέραμε στον προηγούμενο πίνακα που παρείχαν λύση όπως βλέπουμε, στην τελευταία σειρά του πίνακα στο Common Av.T. Παρατηρούμε και πάλι πως η μέθοδος αναζήτησης BFS είναι καλύτερη των υπολοίπων με χρόνο επίλυσης 23.93 δευτερόλεπτα, έπειτα ακολουθεί ο BLB με 24.26, μετά ο DFS με 24.45 και τέλος ο BPH 24.65. Ακόμη, τα προβλήματα που επιλύθηκαν ακριβώς σε λιγότερο από το προγραμματισμένο όριο του χρόνου που προστέθηκε, δηλαδή το διάστημα της μίας ώρας για κάθε αναζήτηση διαφορετικά είναι:

- BLB επίλυσε 27 προβλήματα έναντι 111, σε λίγοτερο από μία ώρα.
- BPH επίλυσε 24 προβλήματα έναντι 111, σε λίγοτερο από μία ώρα.
- DFS επίλυσε 15 προβλήματα έναντι 111, σε λίγοτερο από μία ώρα.
- BFS επίλυσε 28 προβλήματα έναντι 111, σε λίγοτερο από μία ώρα.

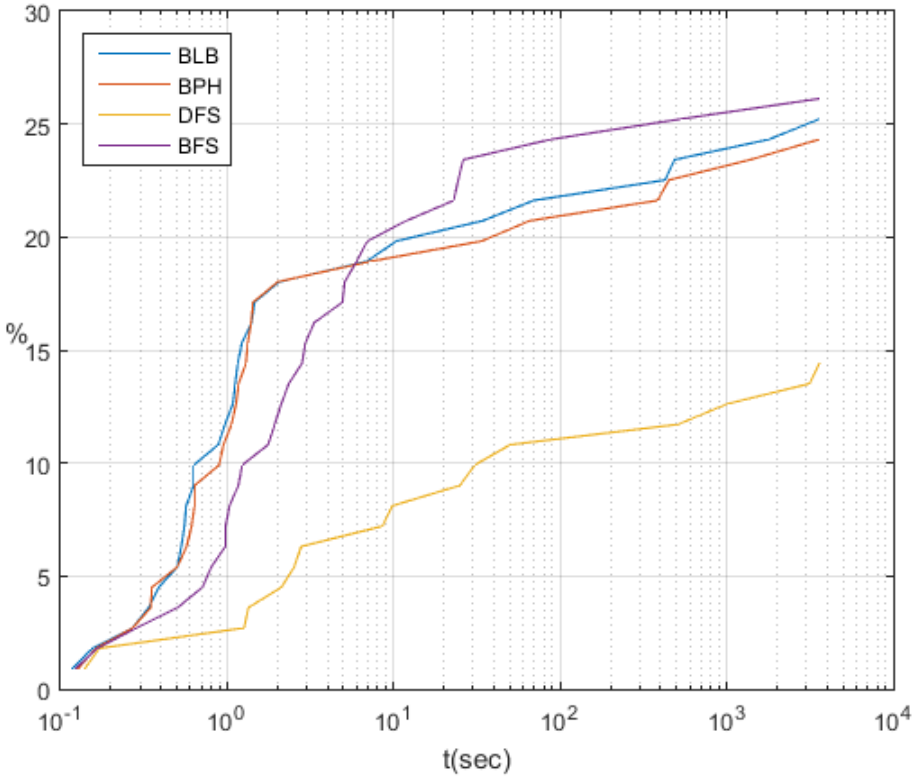
Έπίσης, μπορούμε να παρατηρήσουμε με γνώμονα τις λύσεις των χρόνων των προβλημάτων του LINEAR_T, μιας και είναι οι βέλτιστες μη ακέραιες λύσεις, παρατηρούμε ότι στα περισσότερα προβλήματα και των τεσσάρων μεθόδων αναζήτησης του Πίνακα 4.2 που έχουν φτάσει το όριο των 3,600 δευτερολέπτων για τα ίδια προβλήματα του Πίνακα 4.3, φαίνεται ότι παρόλο που έφτασαν στο τέλος του χρονικού ορίου, οι λύσεις που παράχθηκαν σε σύγκριση με τις βέλτιστες λύσεις που παράγει ο LINEAR είναι πολύ κοντά, άρα μπορεί να είναι και οι βέλτιστες. Ωστόσο, αυτό δεν μπορούν να το αντιληφθούν οι μέθοδοι αναζήτησης καθώς ο στόχος τους είναι να αναζητούν τις βέλτιστες λύσεις και να διακλαδώνουν το δένδρο. Άρα, καταλήγουμε ότι οι μέθοδοι δε θα σταματούσαν την αναζήτηση αν το προγραμματισμένο χρονικό όριο ήταν μεγαλύτερο αν δεν ολοκλήρωναν την αναζήτηση ολικά. Άρα, δεν μπορούμε να πούμε ότι η λύση είναι βέλτιστη, παρόλο που μπορεί να είναι. Ακόμη, οι μέθοδοι αναζήτησης παρατηρούμε ότι, στα περισσότερα αποτελέσματα ή μη αποτελέσματα λύσεων των προβλημάτων που παράχθηκαν στο χρονικό όριο των 3,600 δευτερολέπτων ακριβώς είναι ότι:

- Ο BLB παρήγαγε ακριβώς στο χρονικό όριο σε 84 από τα 111 προβλήματα, δηλαδή περίπου στο **76%**

- Ο BPH παρήγαγε ακριβώς στο χρονικό όριο σε 85 από τα 111 προβλήματα, δηλαδή περίπου στο **77%**
- Ο DFS παρήγαγε ακριβώς στο χρονικό όριο σε 94 από τα 111 προβλήματα, δηλαδή περίπου στο **85%**
- Ο BFS παρήγαγε ακριβώς στο χρονικό όριο σε 83 από τα 111 προβλήματα, δηλαδή περίπου στο **75%**

Παρατηρούμε, ότι στο μεγαλύτερο ποσοστό τους για τα προβλήματα οι αλγόριθμοι αναζήτησης τερμάτισαν ακριβώς στο χρονικό όριο που τους δόθηκε. Ωστόσο, βλέπουμε ότι και για αυτά τα προβλήματα, πάλι η μέθοδος αναζήτησης κατά πλάτος είχε καλύτερο ποσοστό και η μέθοδος κατά βάθος τον χειρότερο. Συνεπώς, για τον ακριβή αλγόριθμο λοιπόν καταλήγουμε ότι η πιο αποτελεσματική μέθοδος ως προς τον χρόνο επίλυσης όλων των μονοδιάστατων προβλημάτων που παρουσιάστηκαν είναι η μέθοδος κατά πλάτος. Στο σχήμα 4.1 παρουσιάζεται το γράφημα ποσοστού ολοκλήρωσης σε βάθος χρόνου των μεθόδων αναζήτησης του ακριβή αλγορίθμου.

Σχήμα 4.1: Γράφημα ποσοστού ολοκλήρωσης σε βάθος χρόνου



Κεφάλαιο 5

Συμπεράσματα

Σε αυτήν τη διπλωματική εργασία, ασχοληθήκαμε με το πρόβλημα βέλτιστης κοπής το οποίο είναι ένα πρόβλημα που έχει εφαρμογές στον τομέα των βιομηχανικών επιχειρήσεων και της επιχειρησιακής έρευνας. Το πρόβλημα αυτό, αφορά την κοπή υλικού σε κομμάτια σταθερού μήκους προκειμένου να επιτευχθεί η παραγγελία μηκών και έχει ως στόχο την ελαχιστοποίηση της φύρας όσο περισσότερο γίνεται (δηλαδή τα απόβλητα να περιοριστούν σε μικρότερο αριθμό), έτσι ώστε να εξασφαλισθεί ένας καλύτερος χρόνος επίτευξης και καλύτερο κόστος για την παραγωγή.

Ακόμη, αναφερθήκαμε στην ιστορική αναδρομή του προβλήματος, δηλαδή πως ξεκίνησε το πρόβλημα και από ποιον, και σε τρόπους επίλυσης του προβλήματος για πραγματικά προβλήματα. Επίσης, αναφερθήκαμε σε γραμμικά μοντέλα (Kantorovich, Gilmore και Gomory κ.α.), μεθόδους (simplex), τεχνικές (τεχνική δημιουργίας στήλης), αλγόριθμους (αλγορίθμους διακλάδωσης και δέσμευσης, αλγορίθμους αναζήτησης κ.α.) και προβλήματα (πρόβλημα του σακιδίου, κύριο πρόβλημα) που από κοινού επιλύουν το πρόβλημα αυτό. Από βιβλιογραφικές πηγές, είδαμε πολλούς διαφορετικούς τρόπους που επιλύεται το πρόβλημα συλλογικά όπως ακριβείς αλγορίθμους, που χρησιμοποιούν πολλές μεθόδους, όπως τη μέθοδο διακλάδωσης και κοπής, διακλάδωσης και τιμής και αρκετές ακόμη και ευρετικούς αλγορίθμους, όπου είδαμε αλγορίθμους γενετικούς και υβριδικούς που επιλύουν το πρόβλημα. Ακόμη, δεν πρέπει να ξεχνάμε ότι το πρόβλημα βέλτιστης κοπής είναι ένα συνδυαστικό πρόβλημα μεγάλης κλίμακας, κάτι που σημαίνει ότι υπάρχει ένας πολύ μεγάλος αριθμός τεχνικών λύσεων, που έχουν αναφερθεί σε άλλες εργασίες όπου η κάθε μια έχει τα δικά της μειονεκτήματα και πλεονεκτήματα στις προσεγ-

γίσεις που παρουσιάζουν [70].

Ωστόσο, σε αυτήν την εργασία, χρησιμοποιήθηκαν δύο αλγόριθμοι για την επίλυση του προβλήματος, όπου ο πρώτος είναι ένας αλγόριθμος επίτευξης της γραμμικής χαλάρωσης που όμως δεν μπορεί να λειτουργήσει σε πραγματικά προβλήματα. Ακόμη χρησιμοποιήθηκε και ένας ακριβής αλγόριθμος, όπου εκτελέστηκε τέσσερις φορές, με τέσσερις διαφορετικές μεθόδους αναζήτησης που είναι η μέθοδος καλύτερου τοπικού ορίου, καλύτερης ευρετικής προβολής, μέθοδος κατά βάθος και μέθοδος κατά πλάτος. Σύμφωνα με τα αποτελέσματα που παράχθηκαν και παρουσιάστηκαν σε πίνακες της εργασίας, παρατηρήσαμε και καταλήξαμε έπειτα από τη σύγκριση που πραγματοποιήθηκε στο συμπέρασμα ότι, η πιο αποδοτική μέθοδος αναζήτησης για τα προβλήματα που επιλύθηκαν είναι ο αλγόριθμος κατά πλάτος, καθώς παρείχε τα καλύτερα αποτελέσματα από όλες τις μεθόδους, και ως προς την παραγωγή των αποτελεσμάτων, όπου παρήγαγε για τα περισσότερα προβλήματα λύση και στα περισσότερα την καλύτερη, αλλά και ως προς τον χρόνο, όπου παρείχε τον καλύτερο μέσο όρο επίλυσης των προβλημάτων. Η χειρότερη ωστόσο μέθοδος φάνηκε ότι είναι η αναζήτηση κατά βάθος, ως προς την παραγωγή αποτελέσματος, καθώς παρήγαγε τα λιγότερα αποτελέσματα, όμως ως προς τον χρόνο επίλυσης δεν ήταν η χειρότερη καθώς ήταν αυτής της καλύτερης ευρετικής προβολής.

Τέλος, μπορεί να υπάρξει βελτίωση στο μέλλον για το πρόβλημα αυτό για τις μεθόδους που χρησιμοποιήσαμε, καθώς παρατηρούμε ότι για τους χρόνους του αλγορίθμου με όλες τις μεθόδους αναζήτησης, ο μέσος όρος ήταν μεγάλος και για τα περισσότερα προβλήματα χρησιμοποιήθηκε όλο το χρονικό διάστημα που δόθηκε, κάτι που σημαίνει ότι μπορεί μελλοντικά να υπάρξει βελτίωση στον αλγόριθμο για τη διαδικασία παραγωγής βέλτιστων λύσεων καθώς, αυτή είναι η λύση στο να μειωθεί ο χρόνος επίλυσης του προβλήματος για πιο γρήγορες και αποδοτικές λύσεις.

Βιβλιογραφία

[1] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[2] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, 1990.

[3] V.-D. Cung, M. Hifi, and B. Le Cun, "Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm," *International Transactions in Operational Research*, vol. 7, no. 3, pp. 185–210, 2000.

[4] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Management science*, vol. 6, no. 4, pp. 366–422, 1960.

[5] A. Mobasher and A. Ekici, "Solution approaches for the cutting stock problem with setup cost," *Computers & operations research*, vol. 40, no. 1, pp. 225–235, 2013.

[6] Y. Lirov, "Knowledge based approach to the cutting stock problem," *Mathematical and computer modelling*, vol. 16, no. 1, pp. 107–125, 1992.

[7] D. J. Alem, P. A. Munari, M. N. Arenales, and P. A. V. Ferreira, "On the cutting stock problem under stochastic demand," *Annals of Operations Research*, vol. 179, no. 1, pp. 169–186, 2010.

[8] H. Ben Amor and J. Valério de Carvalho, "Cutting stock problems," in *Column generation*, pp. 131–161, Springer, 2005.

[9] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.

[10] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting stock problem—part ii," *Operations research*, vol. 11, no. 6, pp. 863–888, 1963.

[11] R. W. Haessler and P. E. Sweeney, "Cutting stock problems and solution procedures," *European Journal of Operational Research*, vol. 54, no. 2, pp. 141–150, 1991.

[12] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European journal of operational research*, vol. 183, no. 3, pp. 1109–1130, 2007.

[13] R. Morabito and V. Garcia, "The cutting stock problem in a hardboard industry: A case study," *Computers & Operations Research*, vol. 25, no. 6, pp. 469–485, 1998.

[14] D. Gale, "Linear programming and the simplex method," *Notices of the AMS*, vol. 54, no. 3, pp. 364–369, 2007.

[15] J. V. De Carvalho, "Lp models for bin packing and cutting stock problems," *European Journal of Operational Research*, vol. 141, no. 2, pp. 253–273, 2002.

[16] G. A. Ogunranti and A. E. Oluleye, "Minimizing waste (off-cuts) using cutting stock model: The case of one dimensional cutting stock problem in wood working industry," *Journal of Industrial Engineering and Management*, vol. 9, no. 3, pp. 834–859, 2016.

[17] G. Belov, "Problems, models and algorithms in one-and two-dimensional cutting," 2003.

[18] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete applied mathematics*, vol. 28, no. 1, pp. 59–70, 1990.

[19] M. E. Lübbecke, "Column generation," *Wiley encyclopedia of operations research and management science. Wiley, New York*, pp. 1–14, 2010.

[20] L. R. Ford Jr and D. R. Fulkerson, "A suggested computation for maximal multi-commodity network flows," *Management Science*, vol. 5, no. 1, pp. 97–101, 1958.

[21] A. Farley, "Practical adaptations of the gilmore-gomory approach to cutting stock problems," *Operations-Research-Spektrum*, vol. 10, no. 2, pp. 113–123, 1988.

[22] O. Marcotte, "The cutting stock problem and integer rounding," *Mathematical Programming*, vol. 33, no. 1, pp. 82–92, 1985.

[23] H. M. Salkin and C. A. De Kluyver, "The knapsack problem: a survey," *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127–144, 1975.

[24] J. H. Lorie and L. J. Savage, "Three problems in rationing capital," *The journal of business*, vol. 28, no. 4, pp. 229–239, 1955.

[25] R. Bellman, "Some applications of the theory of dynamic programming—a review," *Journal of the Operations Research Society of America*, vol. 2, no. 3, pp. 275–288, 1954.

[26] R. Bellman, "Notes on the theory of dynamic programming, 5. maximization over discrete sets," tech. rep., RAND CORP SANTA MONICA CA, 1955.

[27] R. Bellman and R. Kalaba, "On the role of dynamic programming in statistical communication theory," *IRE Transactions on Information Theory*, vol. 3, no. 3, pp. 197–203, 1957.

[28] R. Bellman, "Comment on dantzig's paper on discrete variable extremum problems," *Operations Research*, vol. 5, no. 5, pp. 723–724, 1957.

[29] G. B. Dantzig, "Discrete-variable extremum problems," *Operations research*, vol. 5, no. 2, pp. 266–288, 1957.

[30] H. Greenberg, "An algorithm for the computation of knapsack functions," *Journal of Mathematical Analysis and Applications*, vol. 26, no. 1, pp. 159–162, 1969.

[31] J. S. Yormark, "Accelerating greenberg's method for the computation of knapsack functions," *Journal of Mathematical Analysis and Applications*, vol. 49, no. 3, pp. 629–637, 1975.

[32] S. E. Dreyfus and K. L. Prather, *Improved algorithms for knapsack problems*. University of California, Operations Research Center, 1970.

[33] H. Dyckhoff, "A new linear programming approach to the cutting stock problem," *Operations Research*, vol. 29, no. 6, pp. 1092–1104, 1981.

[34] E. L. Johnson, "Modeling and strong linear programs for mixed integer programming," in *Algorithms and model formulations in mathematical programming*, pp. 1–43, Springer, 1989.

[35] H. Stadtler, "A comparison of two optimization procedures for 1-and 1 1/2-dimensional cutting stock problems," *Operations-Research-Spektrum*, vol. 10, no. 2, pp. 97–111, 1988.

[36] M. Rao, "On the cutting stock problem," 1976.

[37] H. Dyckhoff, "Produktionstheoretische fundierung industrieller zuschneideprozesse," *Operations-Research-Spektrum*, vol. 10, no. 2, pp. 77–96, 1988.

[38] F. Vanderbeck and M. W. Savelsbergh, "A generic view of dantzig–wolfe decomposition in mixed integer programming," *Operations Research Letters*, vol. 34, no. 3, pp. 296–306, 2006.

[39] J. Desrosiers and M. E. Lübbecke, "A primer in column generation," in *Column generation*, pp. 1–32, Springer, 2005.

[40] C. Cheng, B. Feiring, and T. Cheng, "The cutting stock problem—a survey," *International Journal of Production Economics*, vol. 36, no. 3, pp. 291–305, 1994.

[41] F. Vanderbeck, "On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Operations Research*, vol. 48, no. 1, pp. 111–128, 2000.

[42] P. Wentges, "Weighted dantzig-wolfe decomposition for linear mixed-integer programming," *International Transactions in Operational Research*, vol. 4, no. 2, pp. 151–162, 1997.

[43] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*, vol. 5. Springer Science & Business Media, 2006.

[44] J. Clausen, "Branch and bound algorithms-principles and examples," *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999.

[45] M. E. Lalami and D. El-Baz, "Gpu implementation of the branch and bound method for knapsack problems," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pp. 1769–1777, IEEE, 2012.

[46] G. Zarpellon, J. Jo, A. Lodi, and Y. Bengio, "Parameterizing branch-and-bound search trees to learn branching policies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 3931–3939, 2021.

[47] G. Scheithauer and J. Terno, "A branch&bound algorithm for solving one-dimensional cutting stock problems exactly," *Applicationes Mathematicae*, vol. 23, no. 2, pp. 151–167, 1995.

[48] P. H. Vance, "Branch-and-price algorithms for the one-dimensional cutting stock problem," *Computational optimization and applications*, vol. 9, no. 3, pp. 211–228, 1998.

[49] F. Vanderbeck, "Computational study of a column generation algorithm for bin packing and cutting stock problems," *Mathematical Programming*, vol. 86, no. 3, pp. 565–594, 1999.

[50] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, no. 1, pp. 65–77, 2002.

[51] G. Belov and G. Scheithauer, "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting," *European journal of operational research*, vol. 171, no. 1, pp. 85–106, 2006.

[52] A. Martin, "General mixed integer programming: Computational issues for branch-and-cut algorithms," in *Computational combinatorial optimization*, pp. 1–25, Springer, 2001.

[53] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[54] D. Feillet, "A tutorial on column generation and branch-and-price for vehicle routing problems," *4or*, vol. 8, no. 4, pp. 407–424, 2010.

[55] J. Desrosiers and M. E. Lübbecke, "Branch-price-and-cut algorithms," *Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Chichester*, pp. 109–131, 2011.

[56] S. Nayak, *Fundamentals of Optimization Techniques with Algorithms*. Academic Press, 2020.

[57] S. Y. Balaman, *Decision-Making for Biomass-Based Production Chains: The Basic Concepts and Methodologies*. Academic Press, 2018.

[58] H. H. Yanasse and M. S. Limeira, "A hybrid heuristic to reduce the number of different patterns in cutting stock problems," *Computers & Operations Research*, vol. 33, no. 9, pp. 2744–2756, 2006.

[59] Y. Cui and Y. Yang, "A heuristic for the one-dimensional cutting stock problem with usable leftover," *European Journal of Operational Research*, vol. 204, no. 2, pp. 245–250, 2010.

[60] A. C. Cherri, M. N. Arenales, and H. H. Yanasse, "The one-dimensional cutting stock problem with usable leftover–a heuristic approach," *European Journal of Operational Research*, vol. 196, no. 3, pp. 897–908, 2009.

[61] J. Lee, "In situ column generation for a cutting-stock problem," *Computers & Operations Research*, vol. 34, no. 8, pp. 2345–2358, 2007.

[62] R. R. Golfeto, A. C. Moretti, and L. L. d. Salles Neto, "A genetic symbiotic algorithm applied to the one-dimensional cutting stock problem," *Pesquisa Operacional*, vol. 29, pp. 365–382, 2009.

[63] Y. Cui and Z. Liu, "C-sets-based sequential heuristic procedure for the one-dimensional cutting stock problem with pattern reduction," *Optimization Methods & Software*, vol. 26, no. 1, pp. 155–167, 2011.

[64] C. Alves and J. V. de Carvalho, "A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1315–1328, 2008.

[65] M. Monaci, "Algorithms for packing and scheduling problems," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, no. 1, pp. 85–87, 2003.

[66] G. Belov and G. Scheithauer, "A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths," *European Journal of Operational Research*, vol. 141, no. 2, pp. 274–294, 2002.

[67] G. Scheithauer, J. Terno, A. Müller, and G. Belov, "Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm," *Journal of the Operational Research Society*, vol. 52, no. 12, pp. 1390–1401, 2001.

[68] K. Kostikas and C. Fragakis, "Genetic programming applied to mixed integer programming," in *European conference on genetic programming*, pp. 113–124, Springer, 2004.

[69] S. Jaiswal, "Uninformed search algorithms." `https://www.javatpoint.com/ai-uninformed-search-algorithms/`, 2019.

[70] B. L. Golden, "Approaches to the cutting stock problem," *AIIE transactions*, vol. 8, no. 2, pp. 265–274, 1976.