ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΜΕΘΟΔΟΣ ΜΑΘΗΜΑΤΙΚΗΣ ΑΠΟΣΥΝΘΕΣΗΣ DANTZIG-WOLFE

υπό

Γεωργίου Μπάκαβου

Διπλωματική Εργασία

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για
την απόκτηση του Διπλώματος Μηχανολόγου Μηχανικού

Βόλος, 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΜΕΘΟΔΟΣ ΜΑΘΗΜΑΤΙΚΗΣ ΑΠΟΣΥΝΘΕΣΗΣ DANTZIG-WOLFE

υπό

Γεωργίου Μπάκαβου

Διπλωματική Εργασία

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για
την απόκτηση του Διπλώματος Μηχανολόγου Μηχανικού

Βόλος, 2022

© 2022 Γεώργιος Μπάκαβος

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανολόγων Μηχανικών της Πολυτεχνικής Σχολής του Πανεπιστημίου Θεσσαλίας δεν υποδηλώνει αποδοχή των απόψεων του συγγραφέα (Ν. 5343/32 αρ. 202 παρ. 2).

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Γεώργιος Σαχαρίδης Αναπληρωτής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Κωνσταντίνος Αμπουντώλας Αναπληρωτής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Γεώργιος Λυμπερόπουλος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά, θα ήθελα να εκφράσω τις ειλικρινείς ευχαριστίες μου στον επιβλέποντα της διπλωματικής εργασίας μου, Αναπληρωτή Καθηγητή κ. Γεώργιο Σαχαρίδη, για την πολύτιμη βοήθεια και καθοδήγησή του κατά τη διάρκεια της υλοποίησης της εργασίας μου.

Επίσης, είμαι ευγνώμων στα υπόλοιπα μέλη της εξεταστικής επιτροπής της διπλωματικής εργασίας μου, Αναπληρωτή Καθηγητή κ. Κωνσταντίνο Αμπουντώλα και Καθηγητή κ. Γεώργιο Λυμπερόπουλο για την προσεκτική ανάγνωση της εργασίας μου και για τις πολύτιμες υποδείξεις τους. Ιδιαίτερα, οφείλω ευχαριστίες στους υποψήφιους διδάκτορες Καλτή Τιμολέων και Φραγκογιό Αντώνιο για την πολύτιμη βοήθεια και το χρόνο που αφιέρωσαν για την υλοποίηση της εργασίας ιδιαίτερα για τον προγραμματισμό σε περιβάλλον C++/CPLEX.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στους γονείς μου, για την ολόψυχη αγάπη και υποστήριξή τους όλα αυτά τα χρόνια. Αφιερώνω αυτή την εργασία στην μητέρα μου και στον πατέρα μου.

ΜΕΘΟΔΟΣ ΜΑΘΗΜΑΤΙΚΗΣ ΑΠΟΣΥΝΘΕΣΗΣ DANTZIG-WOLFE

Γεώργιος Μπάκαβος

Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας, 2022

Επιβλέπων Καθηγητής: Δρ. Γεώργιος Σαχαρίδης,

Αναπληρωτής Καθηγητής Επιχειρησιακής Έρευνας

Περίληψη

Οι μέθοδοι μαθηματικής αποσύνθεσης, κυρίως μετά το 1960, χρησιμοποιούνται για την επίλυση γραμμικών προβλημάτων μεγάλης κλίμακας στο πεδίο της επιχειρησιακής έρευνας. Μπορούν να χρησιμοποιηθούν σε προβλήματα, όπου οι μέχρι τώρα γνωστοί αλγόριθμοι απαιτούν στη χειρότερη περίπτωση εκθετικό χρόνο για την επίλυσή τους. Πρόκειται για μεθόδους προσέγγισης της κυρτής θήκης (convex hull) του συνόλου των εφικτών λύσεων του προβλήματος βελτιστοποίησης που αντιμετωπίζεται. Το κίνητρο χρήσης τους, είναι η βελτίωση του ορίου γραμμικής χαλάρωσης, κατασκευάζοντας δυναμικά ένα δεύτερο προσεγγιστικό πολύεδρο, που τέμνεται με το αρχικό, για να σχηματιστεί μια καλύτερη προσέγγιση του convex hull. Διακρίνονται σε δύο κύριες κατηγορίες: τις παραδοσιακές και τις ολοκληρωτικές μεθόδους. Οι παραδοσιακές μέθοδοι διαιρούνται με τη σειρά τους σε εσωτερικές (π.χ. μαθηματική αποσύνθεση Dantzig-Wolfe, Lagrange) και σε εξωτερικές μεθόδους (π.χ. μαθηματική αποσύνθεση Benders). Η παρούσα διπλωματική εργασία ασχολείται με την μέθοδο μαθηματικής αποσύνθεσης Dantzig-Wolfe. Αυτή η μέθοδος βασίζεται στην ιδέα της εφαρμογής του θεωρήματος αναπαράστασης Minkowski-Weyl για κυρτά πολύεδρα σε ένα υποσύνολο των περιορισμών του προβλήματος και στη χρήση της μεθόδου δημιουργίας στήλης για την επίλυση του περιορισμένου κύριου προβλήματος. Η διπλωματική εργασία επικεντρώνεται σε τέσσερις στόχους: ο πρώτος αφορά την περιγραφή της μεθόδου αυτής, ο δεύτερος αφορά παραδείγματα εφαρμογής της σε προβλήματα βελτιστοποίησης, ο τρίτος αφορά την περιγραφή μεθόδων επιτάχυνσής της και τέλος ο τέταρτος την ανάπτυξη μίας νέας μεθόδου επιτάχυνσής της. Για την επίτευξη των στόχων, η εργασία διαιρείται σε δύο μέρη. Το πρώτο μέρος είναι το θεωρητικό. Σε αυτή την έκταση, αρχικά η εξεταζόμενη μέθοδος περιγράφεται και στη συνέχεια εφαρμόζεται στο

μονοδιάστατο πρόβλημα κοπής αποθέματος και στο μονοδιάστατο πρόβλημα συσκευασίας κάδου. Επίσης, αναφέρονται τα μειονεκτήματα της κλασικής μεθόδου δημιουργίας στήλης και οι τεχνικές επιτάχυνσής της. Τέλος, ως μία νέα μέθοδος προτείνεται μία προσέγγιση της μεθόδου δημιουργίας πολλαπλών στηλών, όπου σε κάθε επανάληψη, στο περιορισμένο κύριο πρόβλημα προστίθενται $n = k + 1$ στήλες, όπου «1» είναι η μία στήλη που προέρχεται από την κλασική μέθοδο δημιουργίας στήλης και οι υπόλοιπες k στήλες προέρχονται από τροποποιημένο/-α υπο-πρόβλημα/-τα, με χρήση μεθόδων εξομάλυνσης δυϊκών τιμών. Στο δεύτερο μέρος, που είναι το υπολογιστικό μέρος της εργασίας, αναλύεται η μεθοδολογία που ακολουθήθηκε για την εκτέλεση υπολογιστικών πειραμάτων, για την αξιολόγηση της νέας μεθόδου και τη σύγκρισή της με την κλασική μέθοδο δημιουργίας στήλης. Τα αποτελέσματα που προκύπτουν δείχνουν ότι η προτεινόμενη μέθοδος επιτάχυνσης παρουσιάζει πιο βελτιωμένη απόδοση συγκριτικά με την κλασική μέθοδο δημιουργίας στήλης, ιδιαίτερα με την προσθήκη $n = 2$ μέχρι $n = 6$ στηλών, που μπορεί να χρησιμοποιηθεί από την μαθηματική αποσύνθεση Dantzig-Wolfe.

Λέξεις-κλειδιά: Μέθοδοι μαθηματικής αποσύνθεσης, θεώρημα αναπαράστασης Minkowski-Weyl, μέθοδος δημιουργίας στήλης, μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe, τεχνικές σταθεροποίησης, μέθοδος δημιουργίας πολλαπλών στηλών

DANTZIG-WOLFE DECOMPOSITION

Georgios Bakavos

Department of Mechanical Engineering, University of Thessaly, 2022

Supervisor: Dr Georgios Sacharidis

Associate Professor of Operational Research

Abstract

Mathematical decomposition methods, mainly after 1960, are used to solve large-scale linear problems in the field of operations research. They can be used in problems where the so far known algorithms require in the worst-case exponential time to solve them. These are methods of approximating the convex hull of the set of feasible solutions of the optimization problem being addressed. The motivation for using them is to improve the linear relaxation limit by dynamically constructing a second approximate polyhedron, which intersects the original one, to form a better approximation of the convex hull. They are divided into two main categories: traditional and integrative methods. Traditional methods are in turn divided into internal (e.g., Dantzig-Wolfe, Lagrange mathematical decomposition) and external methods (e.g., Benders mathematical decomposition). This thesis deals with the Dantzig-Wolfe mathematical decomposition method. This method is based on the idea of applying the Minkowski-Weyl representation theorem for convex polyhedra to a subset of the problem constraints and using the column generation method to solve the constrained master problem. The thesis focuses on four objectives: the first concerns the description of this method, the second concerns examples of its application in optimization problems, the third concerns the description of methods for its acceleration and finally the fourth concerns the development of a new method for its acceleration. To achieve the objectives, the thesis is divided into two parts. The first part is the theoretical part. In this section, first the considered method is described and then it is applied to the one-dimensional cutting stock problem and the one-dimensional bin packing problem. Also, the disadvantages of the classic column generation method and the techniques to accelerate it are mentioned. Finally, as a new

method, an approach to the multiple column generation method is proposed, where in each iteration, $n = k + 1$ columns are added to the restricted master problem, where "1" is the one column derived from the classical column generation method and the remaining k columns are derived from modified sub-problem(s), using dual smoothing methods. In the second part, which is the computational part of the paper, the methodology followed to perform computational experiments is analyzed, to evaluate the new method and compare it with the classic column generation method. The obtained results show that the proposed acceleration method shows more improved performance compared to the classical column generation method, especially with the addition of $n = 2$ to $n = 6$ columns, which can be used by the Dantzig-Wolfe decomposition.

Key words: (Mathematical) Decomposition Methods, Minkowski-Weyl Representation Theorem, Column Generation, Dantzig-Wolfe Decomposition, Stabilization, Multiple Column Generation

Πίνακας περιεχομένων

Κεφάλαιο 1. Εισαγωγή ... 1

1.1 Κίνητρο και υπόβαθρο ... 1

1.1.1 Οι μέθοδοι μαθηματικής αποσύνθεσης ... 1
1.1.2 Η αρχή της μαθηματικής αποσύνθεσης ... 5
1.1.3 Η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe ... 6
1.1.4 Στόχοι της εργασίας ... 7

1.2 Βιβλιογραφική Ανασκόπηση ... 9

1.3 Οργάνωση Διπλωματικής Εργασίας ... 17

Κεφάλαιο 2. Βασικές έννοιες της θεωρίας πολύεδρων-θεώρημα αναπαράστασης MinKowski-Weyl-μειωμένο κόστος ... 18

2.1 Εισαγωγή ... 18

2.2 Βασικές έννοιες της θεωρίας πολυέδρων ... 18

2.2.1 Πολύεδρο ... 18
2.2.2 Κυρτό σύνολο ... 19
2.2.3 Κυρτή θήκη ... 19
2.2.4 Κυρτό πολύτοπο ... 20
2.2.5 Κυρτός συνδυασμός ... 20
2.2.6 Ακραίο σημείο ... 20
2.2.7 Ακτίνες ... 21
2.2.8 Ακραίες Ακτίνες ... 21
2.2.9 Ιδιότητες κυρτών συνόλων ... 22

2.3 Θεώρημα αναπαράστασης Minkowski-Weyl ... 22

2.4 Η έννοια του μειωμένου κόστους ... 24

2.4.1 Μειωμένο κόστος ... 24
2.4.2 Ερμηνεία του μειωμένου κόστους ... 24

Κεφάλαιο 3. Η μέθοδος δημιουργίας στήλης και η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe ... 26

3.1 Εισαγωγή ... 26

3.2 Η μέθοδος δημιουργίας στήλης ... 26

3.2.1 Ιστορικά στοιχεία-εφαρμογές της μεθόδου δημιουργίας στήλης ... 26
3.2.2 Περιγραφή της κλασικής μεθόδου δημιουργίας στήλης ... 27
3.2.3 Ο αλγόριθμος της κλασικής μεθόδου δημιουργίας στήλης ... 29
3.2.4 Όρια στην μέθοδο δημιουργίας στήλης ... 30

3.3 Η αρχή της μαθηματικής αποσύνθεσης Dantzig-Wolfe ... 31

3.3.1 Διατύπωση του κύριου προβλήματος ... 32
3.3.2 Επίλυση του κύριου προβλήματος με τη μέθοδο δημιουργίας στήλης ... 33
3.3.3 Διατύπωση του υπο-προβλήματος ... 34
3.3.4 Επίλυση του υπο-προβλήματος ... 35
3.3.5 Ο αλγόριθμος μαθηματικής αποσύνθεσης Dantzig-Wolfe ... 36
3.3.6 Η γωνιακή δομή του πίνακα περιορισμών ... 37

3.4 Διατύπωση της αποσύνθεσης Dantzig-Wolfe για προβλήματα ακέραιου γραμμικού προγραμματισμού ... 41

3.4.1 Εισαγωγή-ορισμός αρχικού προβλήματος-ορισμός πολυέδρων ... 41
3.4.2 Το θεώρημα αναπαράστασης Minkowski-Weyl για προβλήματα ακέραιου γραμμικού προγραμματισμού ... 42

3.4.3 Διατύπωση του κύριου προβλήματος ... 43
3.4.4 Εφαρμογή της μεθόδου δημιουργίας στήλης ... 46
3.4.5 Σημείωση για την ειδική περίπτωση του δυαδικού ακέραιου γραμμικού προγραμματισμού ... 50
3.4.6 Γραφική και οικονομική ερμηνεία της μαθηματικής αποσύνθεσης Dantzig-Wolfe ... 53

Κεφάλαιο 4. Παραδείγματα εφαρμογής της μεθόδου μαθηματικής αποσύνθεσης Dantzig-Wolfe σε προβλήματα της Επιχειρησιακής Έρευνας ... 58

4.1 Εισαγωγή ... 58

4.2 Το πρόβλημα κοπής αποθέματος ... 59

4.2.1 Ταξινόμηση του προβλήματος κοπής αποθέματος και οι αντίστοιχες εφαρμογές ... 60
4.2.2 Το μονοδιάστατο πρόβλημα κοπής αποθέματος ... 60
4.2.3 Διατύπωση του μονοδιάστατου προβλήματος κοπής αποθέματος ... 61
4.2.4 Εφαρμογή της μαθηματικής αποσύνθεσης Dantzig-Wolfe στο μονοδιάστατο πρόβλημα κοπής αποθέματος ... 63

4.3 Το πρόβλημα συσκευασίας κάδου ... 68

4.3.1 Είδη του προβλήματος συσκευασίας κάδου και αντίστοιχες εφαρμογές ... 68
4.3.2 Το μονοδιάστατο πρόβλημα συσκευασίας κάδου ... 69
4.3.3 Εφαρμογή της μαθηματικής αποσύνθεσης Dantzig-Wolfe στο μονοδιάστατο πρόβλημα συσκευασίας κάδου ... 72

Κεφάλαιο 5. Βελτίωση της απόδοσης της μεθόδου μαθηματικής αποσύνθεσης Dantzig-Wolfe ... 77

5.1 Εισαγωγή ... 77

5.2 Τεχνικές σταθεροποίησης της μεθόδου δημιουργίας στήλης ... 78

5.2.1 Μέθοδος Boxstep ... 79
5.2.2 Πολυεδρικοί όροι ποινής ... 81
5.2.3 Μέθοδοι δέσμης (Τετραγωνικοί όροι ποινής) ... 84
5.2.4 Κυρτοί συνδυασμοί με προηγούμενες δυϊκές λύσεις ... 85

5.3 Η μέθοδος δημιουργίας πολλαπλών στηλών ... 86

Κεφάλαιο 6. Υπολογιστικά πειράματα ... 90

6.1 Εισαγωγή ... 90

6.2 Σημειώσεις που αφορούν την εκτέλεση των υπολογιστικών πειραμάτων ... 90

6.3 Περιγραφή της βιβλιοθήκης BPPLIB ... 91

6.4 Τα διαθέσιμα παραδείγματα δοκιμής της βιβλιοθήκης BPPLIB ... 92

6.4.1 Η μορφή των παραδειγμάτων δοκιμής ... 92
6.4.2 Παραδείγματα δοκιμής και η αντίστοιχη βιβλιογραφία ... 92

6.5 Περιγραφή του λογισμικού ... 94

6.5.1 Συνοπτική περιγραφή του λογισμικού ... 94
6.5.2 Περιγραφή των υπολογιστικών προγραμμάτων ... 94
6.5.3 Πληροφορίες για το IBM ILOG CPLEX Optimization Studio® ... 96

6.6 Εισαγωγή στο περιβάλλον του MS Visual Studio - σύνδεση με τις βιβλιοθήκες βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio® ... 98

6.6.1 Δημιουργία αρχείου .cpp στο MS Visual Studio ... 98
6.6.2 Σύνδεση του Visual Studio με τις βιβλιοθήκες βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio® ... 101
6.6.3 Βασικές συναρτήσεις του IBM ILOG CPLEX Optimization Studio® ... 105

6.6.4 Βασική ροή εργασίας για την επίλυση ενός προβλήματος μέσω C++ σε συνδυασμό με τις βιβλιοθήκες βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio® 107

Κεφάλαιο 7. Αριθμητικά Αποτελέσματα 108

7.1 Περιγραφή των συμβολισμών 108

7.2 Αριθμητικά αποτελέσματα της πρώτης φάσης υπολογιστικών πειραμάτων 109

7.2.1 Αριθμητικά αποτελέσματα της πρώτης φάσης υπολογιστικών πειραμάτων για το 1D-CSP 109

7.2.2 Σχολιασμός των αποτελεσμάτων των υπολογιστικών πειραμάτων για το 1D-CSP 119

7.2.3 Αριθμητικά αποτελέσματα των υπολογιστικών πειραμάτων για το 1D-BPP 126

7.2.4 Σχολιασμός των αριθμητικών αποτελεσμάτων των υπολογιστικών πειραμάτων για το 1D-BPP 133

Κεφάλαιο 8. Συμπεράσματα-προτάσεις 138

8.1 Συμπεράσματα της διπλωματικής εργασίας 138

8.2 Προτάσεις για μελλοντική εργασία 139

Βιβλιογραφία 140

Παράρτημα Α. Κατάλογος ακρωνύμων 150

Παράρτημα Β. Κώδικες σε Wolfram Mathematica® που χρησιμοποιήθηκαν για τη δημιουργία εικόνων του 2ου Κεφαλαίου 151

Παράρτημα Β.1. Κώδικας για την δημιουργία της Εικόνα 2-1 151

Παράρτημα Β.2. Κώδικας για την δημιουργία της Εικόνα 2-3 (α) 151

Παράρτημα Β.3. Κώδικας για την δημιουργία της Εικόνα 2-3 (β) 151

Παράρτημα Γ. Παράδειγμα πηγαίου κώδικα που επιλύει ένα πρόβλημα γραμμικού προγραμματισμού με τη χρήση της γλώσσας προγραμματισμού C++ και των βιβλιοθηκών βελτιστοποίησης του ILOG Optimization Studio® 152

Παράρτημα Δ. Δείγμα πηγαίου κώδικα που υλοποιεί την προτεινόμενη μέθοδο δημιουργίας πολλαπλών στηλών, όπου επιλύει κάθε παράδειγμα 3 φορές, μία για κάθε τιμή της παραμέτρου a $(0.3, 0.6 \,\&\, 0.8)$, προσθέτοντας 2 στήλες στο RMP του 1D-CSP σε κάθε επανάληψη 154

Παράρτημα Ε. Δείγμα πηγαίου κώδικα που υλοποιεί την προτεινόμενη μέθοδο δημιουργίας πολλαπλών στηλών, όπου επιλύει κάθε παράδειγμα 3 φορές, μία για κάθε τιμή της παραμέτρου a $(0.3, 0.6 \,\&\, 0.8)$, προσθέτοντας 10 στήλες στο RMP του 1D-BPP σε κάθε επανάληψη 167

Κατάλογος Πινάκων

Πίνακας 7-1: Περιγραφή των συμβολισμών που χρησιμοποιούνται στους πίνακες των αριθμητικών αποτελεσμάτων..... 108

Πίνακας 7-2: Αριθμητικά αποτελέσματα από τη επίλυση των 10 παραδειγμάτων του συνόλου «HARD10» των Scholl, R. Klein, and C. Jürgens [109] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 109

Πίνακας 7-3: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAA125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a)..... 110

Πίνακας 7-4: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAB125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a)..... 111

Πίνακας 7-5: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBA125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a)..... 112

Πίνακας 7-6: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBB125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a)..... 113

Πίνακας 7-7: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAA250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a)..... 114

Πίνακας 7-8: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAB250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP

σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a)..... 115
Πίνακας 7-9: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBA250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a)..... 116
Πίνακας 7-10: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBB250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 117
Πίνακας 7-11: Συγκεντρωτικός πίνακας των αριθμητικών αποτελεσμάτων των παραδειγμάτων δοκιμής του 1D-CSP που χρησιμοποιήθηκαν, με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 118
Πίνακας 7-12: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 2 στηλών στο RMP του 1D-CSP σε κάθε επανάληψη και με την βέλτιστη τιμή της παραμέτρου a για κάθε σύνολο παραδειγμάτων). 123
Πίνακας 7-13: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 6 στηλών στο RMP του 1D-CSP σε κάθε επανάληψη και με την βέλτιστη τιμή της παραμέτρου a για κάθε σύνολο παραδειγμάτων). 124
Πίνακας 7-14: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 10 στηλών στο RMP του 1D-CSP σε κάθε επανάληψη και με την βέλτιστη τιμή της παραμέτρου a για κάθε σύνολο παραδειγμάτων). 125
Πίνακας 7-15: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «t60» του E. Falkenauer [108] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 126

Πίνακας 7-16: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «t120» του E. Falkenauer [108] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 127

Πίνακας 7-17: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «t249» του E. Falkenauer [108] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 128

Πίνακας 7-18: Αριθμητικά αποτελέσματα από την επίλυση των 100 παραδειγμάτων του συνόλου «Schwerin 1» των P. Schwerin & G. Wäscher [111] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 129

Πίνακας 7-19: Συγκεντρωτικός πίνακας των αριθμητικών αποτελεσμάτων των παραδειγμάτων δοκιμής του 1D-BPP που χρησιμοποιήθηκαν, με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου a). 132

Πίνακας 7-20: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 2 στηλών στο RMP του 1D-BPP σε κάθε επανάληψη και με την βέλτιστη τιμή της παραμέτρου a για κάθε σύνολο παραδειγμάτων). 136

Πίνακας 7-21: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 6 στηλών στο RMP του 1D-BPP σε κάθε επανάληψη και με την βέλτιστη τιμή της παραμέτρου a για κάθε σύνολο παραδειγμάτων). 136

Πίνακας 7-22: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 10 στηλών στο RMP του 1D-BPP σε κάθε επανάληψη και με την βέλτιστη τιμή της παραμέτρου a για κάθε σύνολο παραδειγμάτων). 137

Κατάλογος Εικόνων

Εικόνα 1-1: Σύγκριση της απόδοσης αλγορίθμων ως προς την υπολογιστική πολυπλοκότητά τους [3]. 2

Εικόνα 1-2: Κατηγορίες των μεθόδων μαθηματικής αποσύνθεσης και παραδείγματα που αντιστοιχούν σε αυτές. 5

Εικόνα 1-3: (α) George Bernard Dantzig (1914-2005) και (β) Philip Starr Wolfe (1927-2016).. 6

Εικόνα 1-4: Διάγραμμα ροής των στόχων της διπλωματικής εργασίας..... 9

Εικόνα 1-5: Αριθμός δημοσιευμένων εγγράφων ανά έτος με τίτλο τους «μαθηματική αποσύνθεση Dantzig-Wolfe», 1964-2022 (Scopus) [46]. 13

Εικόνα 1-6: Αριθμός δημοσιευμένων εγγράφων ανά έτος με τίτλο τους «μαθηματική αποσύνθεση Dantzig-Wolfe», 1985-2022 (Web of Science) [47]..... 13

Εικόνα 1-7: Ποσοστά δημοσιευμένων εγγράφων που αφορούν την μαθηματική αποσύνθεση Dantzig-Wolfe ανά επιστημονικό τομέα, 1964-2022 (Scopus) [46]. 14

Εικόνα 1-8: Αριθμός δημοσιευμένων εγγράφων που αφορούν την μαθηματική αποσύνθεση Dantzig-Wolfe ανά επιστημονικό τομέα (Web of Science) [47]. 14

Εικόνα 1-9: Αριθμός δημοσιευμένων εγγράφων ανά συγγραφέα που αφορούν την μαθηματική αποσύνθεση Dantzig-Wolfe, 1964-2022 (Scopus) [46]..... 15

Εικόνα 1-10: Αριθμός δημοσιευμένων εγγράφων ανά έτος με τίτλο «σταθεροποιημένη μέθοδος δημιουργίας στήλης», 1999-2022 (Scopus) [46]. 15

Εικόνα 1-11: Αριθμός δημοσιευμένων εγγράφων που αφορούν την σταθεροποιημένη μέθοδος δημιουργίας στήλης ανά έτος, 1999-2021 (Web of Science) [47]. 16

Εικόνα 1-12: Συχνότητα εφαρμογής της σταθεροποιημένης μεθόδου δημιουργίας στήλης σε επιστημονικούς τομείς (Web of Science) [47]. 16

Εικόνα 2-1:Παράδειγμα ενός πολύεδρου: Δωδεκάεδρο. 19

Εικόνα 2-2: (α) ένα παράδειγμα μιας κυρτής περιοχής (convex region) και (β) ένα παράδειγμα μιας μη-κυρτής περιοχής (non–convex region). 19

Εικόνα 2-3: Η Κυρτή θήκη (convex hull) (α) ενός 2D τυχαίου συνόλου (δημιουργήθηκαν 50 τυχαία σημεία σε 2D επίπεδο από τυχαίους πραγματικούς αριθμούς στο διάστημα [-1,1]) και (β) ενός 3D τυχαίου συνόλου (δημιουργήθηκαν 50 τυχαία σημεία σε επίπεδο 3D από τυχαίους πραγματικούς αριθμούς στο διάστημα [-1,1]). 20

Εικόνα 2-4: Το σημείο w δεν είναι ακραίο σημείο γιατί $w \in u, v$. Το σημείο x είναι ακραίο σημείο αν $x = \lambda y + 1 - \lambda z \; \mu\varepsilon \; 0 < \lambda < 1 \; \gamma\iota\alpha \; y, z \neq x \; \tau\acute{o}\tau\varepsilon \; \varepsilon\acute{\iota}\tau\varepsilon \; y \notin P \; \varepsilon\acute{\iota}\tau\varepsilon \; z \notin P$..... 21

Εικόνα 2-5: Τα ακραία σημεία και οι ακραίες ακτίνες ενός μη φραγμένου πολύεδρου [54]. 22
Εικόνα 2-6: (α) Hermann Minkowski (1864-1909) και (β) Hermann Klaus Hugo Weyl (1885-1955)..... 23
Εικόνα 3-1: Διάγραμμα ροής του κλασικού αλγόριθμου δημιουργίας στήλης [70]..... 30
Εικόνα 3-2: Παράδειγμα απεικόνισης των «εύκολων» και των «δύσκολων» περιορισμών. . 32
Εικόνα 3-3: Γωνιακή δομή όπου Connecting constraints είναι το μπλοκ των περιορισμών σύνδεσης και καθένα από τα μπλοκ Subproblem $1,2 \dots h$*, ορίζει ένα διαφορετικό υπο-πρόβλημα* [75]..... 38
Εικόνα 3-4: Λειτουργία του αλγόριθμου μαθηματικής αποσύνθεσης Dantzig-Wolfe. Σε κάθε επανάληψη, οι δυϊκές τιμές που σχετίζονται με τους περιορισμούς σύνδεσης προκύπτουν από την επίλυση του RMP. Αυτές οι τιμές χρησιμοποιούνται από τα h *υπο-προβλήματα για τον υπολογισμό μιας ενημερωμένης λύσης που βελτιώνει τη συνολική αντικειμενική συνάρτηση* [76]. 41
Εικόνα 3-5: $PLP = x: Ax \geq b, Dx \geq b$ είναι η εφικτή περιοχή της γραμμικής χαλάρωσης του ILP (3-57)-(3-60). $PDW = x: Ax \geq b, x \in conv(X)$ είναι το πολύεδρο της μαθηματικής αποσύνθεσης Dantzig-Wolfe στο ILP. Στην μαθηματική αποσύνθεση Dantzig-Wolfe επαναδιατυπώνεται το $conv\ X$ όλων των ακέραιων λύσεων που είναι εφικτές για το υποσύστημα των περιορισμών $Dx \geq b$, χρησιμοποιώντας τα ακραία σημεία του $x1, x2, x3, \dots$. Η τομή του $conv\ X$ με τους περιορισμούς που απομένουν, δίνουν το PDW που είναι πιο σφικτή προσέγγιση απ' ότι η γραμμική χαλάρωση. 44
Εικόνα 3-6: Παραδείγματα (α) προσέγγισης κυρτότητας σε ένα φραγμένο IP και (β) η προσέγγισης διακριτοποίησης της μαθηματικής αποσύνθεσης Dantzig-Wolfe σε ένα φραγμένο IP [79]. Τα κόκκινου χώματος σημεία είναι ακέραια σημεία. 45
Εικόνα 3-7: Εφικτές λύσεις για (α) το ακέραιο πρόβλημα (IP), (β) τη γραμμική χαλάρωσή του (LP), (γ) της μαθηματικής αποσύνθεσης Dantzig-Wolfe (DWD) και (δ) κέρδος της DWD έναντι της γραμμικής χαλάρωσης (LP) [85]. 56
Εικόνα 3-8: Επικοινωνία μεταξύ του κύριου προβλήματος (MP) και των h υπο-προβλημάτων $SP1, SP2, \dots, SPh$ [17]. 57
Εικόνα 4-1: Αριθμός δημοσιεύσεων σε συνάρτηση με το έτος έκδοσης για το πρόβλημα κοπής αποθέματος, 1970-2022 (Scopus) [46]. 58
Εικόνα 4-2: Αριθμός δημοσιεύσεων σε συνάρτηση με το έτος έκδοσης για το πρόβλημα συσκευασίας κάδου, 1972-2022 (Scopus) [46]. 59

Εικόνα 4-3: (α) απόθεμα κυλίνδρων λαμαρίνας και (β) απόθεμα μεταλλικών ράβδων στη βιομηχανία [87]. 59
Εικόνα 4-4: Μονοδιάστατο πρόβλημα κοπής αποθέματος με έναν τύπο αποθέματος [88]. 61
Εικόνα 4-5: Προσθήκη στηλών στο περιορισμένο κύριο πρόβλημα του μονοδιάστατου προβλήματος κοπής αποθέματος. Όπου, $\lambda p *$ είναι η βέλτιστη τιμή της μεταβλητής λp στην πρώτη επανάληψη, $\lambda p **$ είναι η βέλτιστη τιμή της μεταβλητής λp στην δεύτερη επανάληψη. Οι συντελεστές aip λαμβάνονται από την επίλυση του προβλήματος σακιδίου. 68
Εικόνα 4-6: Παράδειγμα της διαδικασίας επίλυσης ενός 1D-BPP 15 στοιχείων με μήκη μεταξύ 1 και 6 μονάδων και κάδους χωρητικότητας 10 μονάδων. Αυτή τη στιγμή, δέκα αντικείμενα έχουν συσκευαστεί σε πέντε κάδους (δύο κλειστοί και τρεις ανοιχτοί) και πέντε αντικείμενα αναμένουν να συσκευαστούν [98]. 69
Εικόνα 4-7: Προσθήκη στηλών στο περιορισμένο κύριο πρόβλημα του μονοδιάστατου προβλήματος κοπής αποθέματος. Όπου, $\lambda p *$ είναι η βέλτιστη τιμή της μεταβλητής λp στην πρώτη επανάληψη, $\lambda p **$ είναι η βέλτιστη τιμή της μεταβλητής λp στην δεύτερη επανάληψη. Οι συντελεστές aip λαμβάνονται από την επίλυση του προβλήματος σακιδίου. 76
Εικόνα 5-1: Μειονεκτήματα της μεθόδου δημιουργίας στήλης [99]. *Ο κατακόρυφος άξονας αντιπροσωπεύει τις τιμές του περιορισμένου κύριου προβλήματος, ο οριζόντιος άξονας αντιπροσωπεύει τον αριθμό των επαναλήψεων, η κόκκινή καμπύλη αντιπροσωπεύει τη μεταβολή της τιμής του περιορισμένου κύριου προβλήματος και τέλος, η μπλε καμπύλη αντιπροσωπεύει μεταβολή της τιμής του δυϊκού ορίου.* 78
Εικόνα 5-2: Οπτικοποίηση της μεθόδου Boxstep. Όπου $\pi 1, \pi 2$ είναι οι νέες βέλτιστες δυϊκες λύσεις που επιτυγχάνονται στα όρια του πλαισίου και ορίζουν την κατεύθυνση στην οποία θα μεταφερθεί το πλαίσιο και $\pi *$ είναι η βέλτιστη δυϊκή λύση που επιτυγχάνεται στο εσωτερικό του πλαισίου [17]. 81
Εικόνα 5-3: Ποινή σύμφωνα με Du Merle et al [17]. 83
Εικόνα 5-4: Περιοχή εμπιστοσύνης σύμφωνα με Du Merle et al [103]. 84
Εικόνα 5-5: Παραδείγματα συναρτήσεων ποινής: (α) Boxstep, (β), (γ), (δ) τρεις γραμμικές τμηματικές συναρτήσεις ποινής και (ε) η τετραγωνική ποινή της μεθόδου δέσμης (Bundle Method). Όπου S είναι μία συνάρτηση ποινής που αυξάνεται καθώς η δυϊκή μεταβλητή π απομακρύνεται από το κέντρο σταθερότητας π. 85

Εικόνα 6-1: Βήμα 1 δημιουργίας αρχείου.cpp 98
Εικόνα 6-2: Βήμα 2 δημιουργίας αρχείου.cpp 99
Εικόνα 6-3: Βήμα 3 δημιουργίας αρχείου.cpp 99
Εικόνα 6-4: Βήμα 4 δημιουργίας αρχείου.cpp 100
Εικόνα 6-5: Βήμα 5 δημιουργίας αρχείου.cpp. 100
Εικόνα 6-6: Βήμα 1 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® 101
Εικόνα 6-7: Βήμα 2 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® 102
Εικόνα 6-8: Βήμα 3 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® 103
Εικόνα 6-9: Βήμα 4 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® 103
Εικόνα 6-10: Βήμα 5 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® 104
Εικόνα 6-11: Βήμα 6 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® 105
Εικόνα 6-12: Η βασική ροή εργασίας για την επίλυση ενός προβλήματος μέσω C++ και των βιβλιοθηκών βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio®. 107
Εικόνα 7-1: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) csAA125, (β) csAB125, (γ) csBA125, (δ) csBB125 και (ε) HARD10, με την προσθήκη 2 στηλών στο RMP σε κάθε επανάληψη, για 3 διαφορετικές τιμές της παραμέτρου επίπεδου εξομάλυνσης α. 120
Εικόνα 7-2: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) csAA250, (β) csAB250, (γ) csBA250 και (δ) csBB250, με την προσθήκη 2 στηλών στο RMP σε κάθε επανάληψη, για 3 διαφορετικές τιμές της παραμέτρου επίπεδου εξομάλυνσης α. 121
Εικόνα 7-3: Μέσος χρόνος επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) csAA125, (β) csAB125, (γ) csBA125, (δ) csBB125, (ε) HARD10, (στ) csAA250, (ζ) csAB250, (η) csBA250 και (ϑ) csBB250, με την κλασική μέθοδο δημιουργίας στήλης (CCG) και της μεθόδου δημιουργίας πολλαπλών στηλών, με την προσθήκη 2 (MCG_2), 6 (MCG_6) και 10

(MCG_10) στηλών στο RMP σε κάθε επανάληψη αντίστοιχα, για την αντίστοιχη τιμή α. 123

Εικόνα 7-4: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) t60, (β) Schwerin1, (γ) t120 και (δ) t249, για 3 διαφορετικές τιμές της παραμέτρου επίπεδου εξομάλυνσης α. 134

Εικόνα 7-5: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) t60, (β) Schwerin1, (γ) t120 και (δ) t249 με την κλασική μέθοδο δημιουργίας στήλης (CCG) και της μεθόδου δημιουργίας πολλαπλών στηλών, με την προσθήκη 2 (MCG_2), 6 (MCG_6) και 10 (MCG_10) στηλών στο RMP σε κάθε επανάληψη αντίστοιχα, για την τιμή της παραμέτρου α. 135

Κεφάλαιο 1. Εισαγωγή

Το κεφάλαιο αυτό αποτελείται από τρεις ενότητες. Στην πρώτη ενότητα παρουσιάζονται πληροφορίες εισαγωγικού χαρακτήρα που δίνουν το κίνητρο και το υπόβαθρο αυτής της διπλωματικής εργασίας. Στην δεύτερη ενότητα, γίνεται μία σύντομη ανασκόπηση της βιβλιογραφίας. Τέλος, στην τρίτη ενότητα, περιγράφεται η οργάνωση της διπλωματικής εργασίας.

1.1 Κίνητρο και υπόβαθρο

1.1.1 Οι μέθοδοι μαθηματικής αποσύνθεσης

Τα προβλήματα μικτού ακέραιου γραμμικού προγραμματισμού (Mixed Integer Linear Programming, MILP), είναι τα πιο συνηθισμένα σε πραγματικές εφαρμογές της επιχειρησιακής έρευνας [1]. Περιλαμβάνουν την ελαχιστοποίηση (ή τη μεγιστοποίηση) της τιμής κάποιας γραμμικής συνάρτησης πάνω σε μία εφικτή πολυεδρική περιοχή, υποκειμενική σε περιορισμούς ακεραιότητας σε ορισμένες από τις μεταβλητές και μπορεί να οριστεί ως εξής:

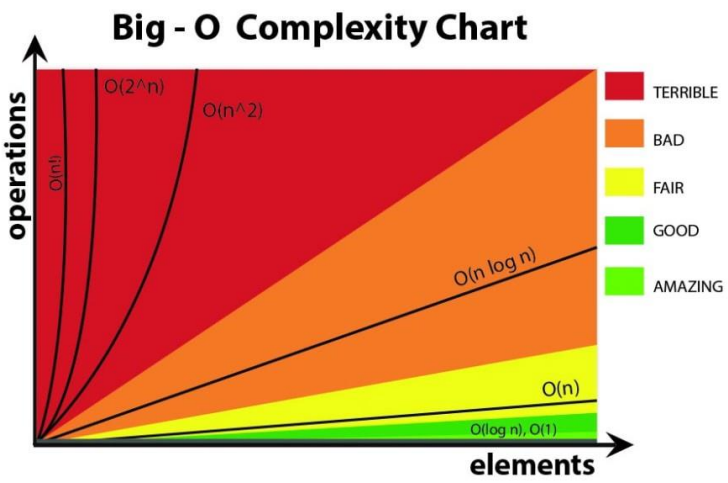
$$\min_{x \in \mathbb{R}^n} \{c^T x | Ax \geq b, x_i \in \mathbb{Z}, \forall i \in I\} \qquad (1\text{-}1)$$

Όπου, $c \in \mathbb{Q}^n$ είναι το διάνυσμα κόστους, $A \in \mathbb{Q}^{m \times n}$είναι ο πίνακας των περιορισμών, $b \in \mathbb{Q}^m$ είναι το δεξί μέλος των περιορισμών και $I \subseteq \{1,2, \dots, n\}$ είναι το σύνολο δεικτών των μεταβλητών που περιορίζονται σε ακέραιες τιμές.

Υπάρχουν δύο σημαντικές ειδικές περιπτώσεις:

- Αν $I = \{1,2, \dots, n\}$, το πρόβλημα **(1-1)** αναφέρεται ως καθαρό πρόγραμμα ακέριου γραμμικού προγραμματισμού (Integer Linear Program, ILP).
- Αν $I = \emptyset$, το πρόβλημα **(1-1)** αναφέρεται ως γραμμικό πρόγραμμα (Linear Program, LP).

Η επίλυση ενός προβλήματος MILP, είναι γνωστό ότι γενικά είναι ένα πρόβλημα NP-hard. Όλοι οι γνωστοί αλγόριθμοι που επιλύουν αυτά τα προβλήματα στην χειρότερη περίπτωση απαιτούν εκθετικό χρόνο επίλυσης [2].



Εικόνα 1-1: Σύγκριση της απόδοσης αλγορίθμων ως προς την υπολογιστική πολυπλοκότητά τους [3].

Ένας από τους πιο επιτυχημένους αλγόριθμους που έχουν αναπτυχθεί για την επίλυση αυτών των προβλημάτων είναι η μέθοδος διακλάδωσης και ορίου (Branch and Bound method) [4],[5], [6]. Οι περισσότερες διαδικασίες οριοθέτησης των προβλημάτων MILP βασίζονται στην επαναληπτική κατασκευή και βελτίωση πολυεδρικών προσεγγίσεων της κυρτής θήκης (στο εξής convex hull) των εφικτών λύσεων του προβλήματος **(1-1)**, που συμβολίζεται με $\mathcal{P}$. Η επίλυση ενός προβλήματος βελτιστοποίησης πάνω σε μια τέτοια πολυεδρική προσέγγιση, που περιέχει πλήρως το $\mathcal{P}$, παράγει ένα όριο που μπορεί να χρησιμοποιηθεί για την εφαρμογή ενός αλγόριθμου Branch and Bound. Η αποτελεσματικότητα της διαδικασίας οριοθέτησης εξαρτάται σε μεγάλο βαθμό σχετικά με το πόσο καλά μπορεί να προσεγγιστεί το convex hull, $\mathcal{P}$. Η πιο απλή προσέγγιση του $\mathcal{P}$ είναι η συνεχής προσέγγιση (continuous approximation), που αποτελείται απλώς από τους γραμμικούς περιορισμούς που υπάρχουν στην αρχική διατύπωση. Ωστόσο, το όριο που προκύπτει από αυτήν την προσέγγιση είναι συχνά ασθενές.

Για την λήψη ισχυρότερων της γραμμικής χαλάρωσης ορίων, μπορούν να χρησιμοποιηθούν οι μέθοδοι μαθηματικής αποσύνθεσης (mathematical decomposition methods) [5].

Βασική ιδέα αυτών των μεθόδων, είναι η εκμετάλλευση της διασπάσιμης δομής ενός προβλήματος βελτιστοποίησης, ώστε η λύση του να μπορεί να μετατραπεί σε λύση αρκετών μικρότερων προβλημάτων (υπο-προβλήματα), που μπορούν να επιλυθούν ευκολότερα [7], [8], [9]. Οι μέθοδοι αυτοί διασπούν τη συμμετρία αναδιατυπώνοντας το πρόβλημα [5] και βελτιώνουν την προσέγγιση του $\mathcal{P}$ δημιουργώντας δυναμικά πρόσθετες πολυεδρικές πληροφορίες [4].

Σύμφωνα με τον M. Galati [4], οι μέθοδοι μαθηματικής αποσύνθεσης διακρίνονται σε δύο κύριες κατηγορίες:

- Τις παραδοσιακές δυναμικές διαδικασίες (Traditional dynamic procedures) και
- Τις ολοκληρωτικές μεθόδους (integrated methods)

1.1.1.1 Παραδοσιακές δυναμικές διαδικασίες

Η κατηγορία των παραδοσιακών δυναμικών διαδικασιών, θεωρεί την τομή ενός πολύεδρου που έχει συμπαγή περιγραφή με μια που δημιουργείται σιωπηρά με την επίλυση ενός βοηθητικού προβλήματος. Αυτό γίνεται γιατί το δεύτερο πολύεδρο έχει περιγραφή που είναι εκθετικού μεγέθους και επομένως δεν μπορεί να οριστεί ρητά αποτελεσματικά. Η κατηγορία αυτή, χωρίζεται σε δύο περαιτέρω κατηγορίες: (α) τις μεθόδους επιπέδου κοπής (cutting plane methods) και (β) τις παραδοσιακές μεθόδους δημιουργίας στήλης (traditional column generation methods) [4].

Από τη μία πλευρά, οι μέθοδοι επιπέδου κοπής βελτιώνουν την προσέγγιση δημιουργώντας δυναμικά ημι-χώρους (half-spaces) που περιέχουν το $\mathcal{P}$, αλλά όχι τη συνεχή προσέγγιση (π.χ. έγκυρες ανισότητες-valid inequalities). Αυτοί οι ημι-χώροι, στη συνέχεια τέμνονται με την τρέχουσα προσέγγιση, με αποτέλεσμα να την βελτιώνουν. Με αυτήν την προσέγγιση, οι έγκυρες ανισότητες δημιουργούνται με την επίλυση ενός σχετικού προβλήματος διαχωρισμού (separation problem). Γενικά, η προσθήκη κάθε έγκυρης ανισότητας δημιουργεί ένα δυνητικά βελτιωμένο όριο. Επειδή αυτές οι μέθοδοι δημιουργούν δυναμικά ένα μέρος της περιγραφής του τελικού προσεγγιστικού πολυέδρου ως τομή του ημι-χώρου (μία εξωτερική αναπαράσταση), αναφέρονται ως μέθοδοι εξωτερικής προσέγγισης (outer approximation methods) [4]. Ένα παράδειγμα αυτών των μεθόδων αποτελεί η αποσύνθεση Benders [10].

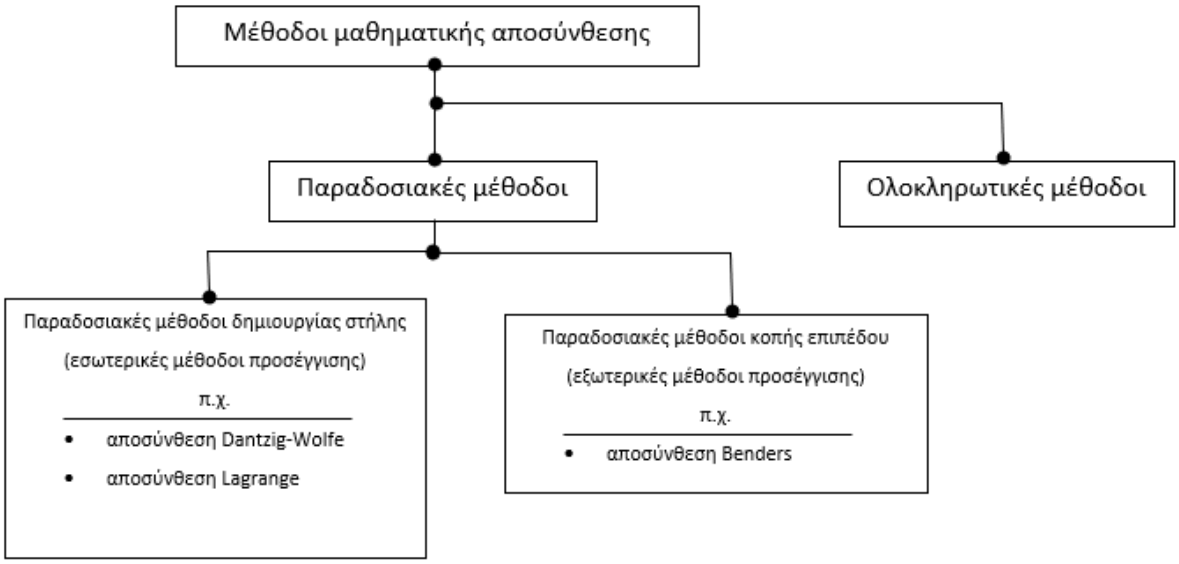
Από την άλλη πλευρά, οι παραδοσιακές μέθοδοι δημιουργίας στήλης βελτιώνουν την προσέγγιση δημιουργώντας δυναμικά τα ακραία σημεία ενός πολύεδρου, που περιέχει το $\mathcal{P}$

και στη συνέχεια τέμνεται με τη συνεχή προσέγγιση (όπως και στη μέθοδο επιπέδου κοπής), για να προκύψει ένα τελικό προσεγγιστικό πολύεδρο. Στην περίπτωση αυτή, κάθε διαδοχικό ακραίο σημείο δημιουργείται με τη λύση ενός σχετικού προβλήματος βελτιστοποίησης σε κάθε βήμα. Επειδή οι μέθοδοι αυτοί δημιουργούν δυναμικά ένα μέρος της περιγραφής του προσεγγιστικού πολυέδρου ως το convex hull ενός πεπερασμένου συνόλου (μια εσωτερική αναπαράσταση), αναφέρονται ως εσωτερικές μέθοδοι προσέγγισης (inner approximation methods). Χαρακτηριστικά παραδείγματα αυτής της κατηγορίας αποτελούν η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe και η χαλάρωση Lagrange (Lagrangian relaxation) [4].

Οι δύο παραπάνω κατηγορίες (οι εσωτερικές και οι εξωτερικές μέθοδοι) λειτουργούν κατά προσέγγιση εναλλάσσοντας μια διαδικασία υπολογισμού της λύσης και των οριοθετημένων πληροφοριών (κύριο πρόβλημα) και μια διαδικασία για την αύξηση της τρέχουσας προσέγγισης (υπο-πρόβλημα). Ωστόσο, οι δύο αυτές προσεγγίσεις, διαφέρουν σημαντικά. Οι εξωτερικές μέθοδοι αφενός, απαιτούν από το κύριο πρόβλημα να παράγει αρχικές πληροφορίες λύσης, οι οποίες στη συνέχεια αποτελούν την είσοδο του υπο-προβλήματος (ένα πρόβλημα διαχωρισμού). Οι εσωτερικές μέθοδοι αφετέρου, απαιτούν δυϊκές πληροφορίες οι οποίες στη συνέχεια χρησιμοποιούνται ως είσοδος του υπο-προβλήματος (πρόβλημα βελτιστοποίησης). Επομένως, οι δύο προσεγγίσεις μπορούν να θεωρηθούν ως δυϊκές μεταξύ τους [4], [11].

1.1.1.2 Ολοκληρωτικές μέθοδοι

Στις ολοκληρωτικές μεθόδους, χρησιμοποιούνται ταυτόχρονα τεχνικές εσωτερικής και εξωτερικής προσέγγισης [4].



Εικόνα 1-2: Κατηγορίες των μεθόδων μαθηματικής αποσύνθεσης και παραδείγματα που αντιστοιχούν σε αυτές.

1.1.2 Η αρχή της μαθηματικής αποσύνθεσης

Ας υποτεθεί ότι το **(1-1)** είναι ένα γραμμικό ακέραιο πρόβλημα. Η πιο συχνά χρησιμοποιούμενη μέθοδος οριοθέτησης προβλημάτων γραμμικού ακέριου προγραμματισμού είναι η επίλυση της χαλάρωσης γραμμικού προγραμματισμού που λαμβάνεται αφαιρώντας την απαίτηση ακεραιότητας από τη διατύπωση ILP. Το όριο γραμμικής χαλάρωσης είναι:

$$z_{ILP} = \min_{x \in \mathbb{R}^n}\{c^T x | Ax \geq b\} = \min_{x \in \mathcal{S}}\{c^T x\} \qquad (1\text{-}2)$$

και προκύπτει με την επίλυση ενός γραμμικού προγράμματος με την αρχική αντικειμενική συνάρτηση c πάνω από το πολύεδρο $\mathcal{S}$. Είναι φανερό ότι $z_{LP} \leq z_{ILP}$ και $\mathcal{P} \subseteq \mathcal{S}$. Αυτή η χαλάρωση LP είναι συνήθως πολύ πιο εύκολο να λυθεί σε σχέση με το αρχικό ILP, αλλά το z_{LP} μπορεί γενικά να είναι αυθαίρετα πολύ μακριά από το z_{ILP}. Επομένως, πρέπει να εξεταστούν πιο αποτελεσματικότερες διαδικασίες.

Μια περιγραφή ενός ILP που αναπαρίσταται με έναν πολυωνυμικό αριθμό μεταβλητών και περιορισμών ονομάζεται συμπαγής. Στις περισσότερες περιπτώσεις, η περιγραφή του $\mathcal{S}$ είναι συμπαγής, μπορεί να αναπαρασταθεί ρητά και το όριο υπολογίζεται χρησιμοποιώντας έναν τυπικό αλγόριθμο γραμμικού προγραμματισμού. Για να βελτιωθεί το

όριο LP, οι μέθοδοι μαθηματικής αποσύνθεσης κατασκευάζουν ένα δεύτερο προσεγγιστικό πολύεδρο $\mathcal{Q}$, το οποίο τέμνεται με το πολύεδρο $\mathcal{S}$, για να σχηματιστεί μια καλύτερη προσέγγιση. Σε αντίθεση με το $\mathcal{S}$, το πολύεδρο $\mathcal{Q}$ έχει συνήθως εκθετικό μέγεθος και επομένως πρέπει να δημιουργηθούν δυναμικά τμήματα της περιγραφής του. Αυτό επιτυγχάνεται είτε με κάποια μέθοδο επιπέδου κοπής (εξωτερική προσέγγιση) π.χ. αποσύνθεση Benders είτε με κάποια μέθοδο δημιουργίας στηλών (εσωτερική προσέγγιση) π.χ. αποσύνθεση Dantzig-Wolfe ή χαλάρωση Lagrange [4].

Από τις παραπάνω μεθόδους, στην παρούσα εργασία, εξετάζεται η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe.

1.1.3 Η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe

Η μαθηματική αποσύνθεση Dantzig-Wolfe (Dantzig-Wolfe Decomposition, DWD), είναι μία μέθοδος αποσύνθεσης προβλημάτων μαθηματικού προγραμματισμού μεγάλης κλίμακας [12]. Αναπτύχθηκε από τους δύο Αμερικανούς μαθηματικούς George Bernard Dantzig [**Εικόνα 1-3 (α)**] [13] και Philip Starr Wolfe [**Εικόνα 1-3 (β)**] [14]. Η μέθοδος, δημοσιεύτηκε το 1960 [15] και το 1961 [16] για προβλήματα γραμμικού προγραμματισμού. Η βασική ιδέα της μεθόδου είναι η εφαρμογή του θεωρήματος αναπαράστασης Minkowski-Weyl για κυρτά πολύεδρα σε ένα υποσύνολο των περιορισμών του προβλήματος, που πρέπει να επιλυθεί και η χρήση της μεθόδου δημιουργίας στήλης για την βελτίωση του ελέγχου γραμμικών προβλημάτων μεγάλης κλίμακας.

(α) (β)

Εικόνα 1-3: (α) George Bernard Dantzig (1914-2005) και (β) Philip Starr Wolfe (1927-2016).

Το κίνητρο της χρήσης της αρχής αποσύνθεσης Dantzig-Wolfe είναι να ληφθεί ένα καλύτερο κάτω όριο για ένα δέντρο branch-and-bound παρά να εφαρμοστεί γραμμική χαλάρωση στο αρχικό πρόβλημα. Γενικά, τα μοντέλα που προκύπτουν από την μαθηματική αποσύνθεση Dantzig-Wolfe [17]:

- Γίνονται πιο εύκολα ως προς τον χειρισμό τους, καθώς χρησιμοποιείται η μέθοδος δημιουργίας στήλης.
- Είναι κατάλληλα για την αντιμετώπιση μη γραμμικών περιορισμών: αντιμετωπίζονται από ένα υπο-πρόβλημα δυναμικού προγραμματισμού.
- Μπορεί να είναι ισχυρότερα: λαμβάνονται ισχυρότερα όρια από εκείνα της γραμμικής χαλάρωσης.

Άλλος ένας επιπρόσθετος λόγος χρήσης της μαθηματικής αποσύνθεσης Dantzig-Wolfe είναι η συμβολή της στην εξοικονόμηση υπολογιστικής μνήμης. Η κύρια ιδιότητα της μεθόδου να υλοποιείται παράλληλα, την καθιστά ιδιαίτερα ελκυστική: Κατά τη φάση εφαρμογής της μεθόδου δημιουργίας στήλης, επιλύεται ένα σύνολο ανεξάρτητων, γραμμικών, μικρών, υπο-προβλημάτων, που μπορούν εύκολα να επιλυθούν παράλληλα, ιδιαίτερα με τη χρήση λογισμικών [18].

1.1.4 Στόχοι της εργασίας

Οι στόχοι της παρούσας διπλωματικής εργασίας είναι οι ακόλουθοι:

- Η μελέτη της μεθόδου μαθηματικής αποσύνθεσης Dantzig-Wolfe.
- Η εφαρμογή της σε δύο προβλήματα της επιχειρησιακής έρευνας.
- Η περιγραφή μεθόδων επιτάχυνσής της.
- Η ανάπτυξη μίας νέας μεθόδου επιτάχυνσής της.

Πιο συγκεκριμένα για κάθε στόχο αναφέρονται τα αντίστοιχα σχόλια:

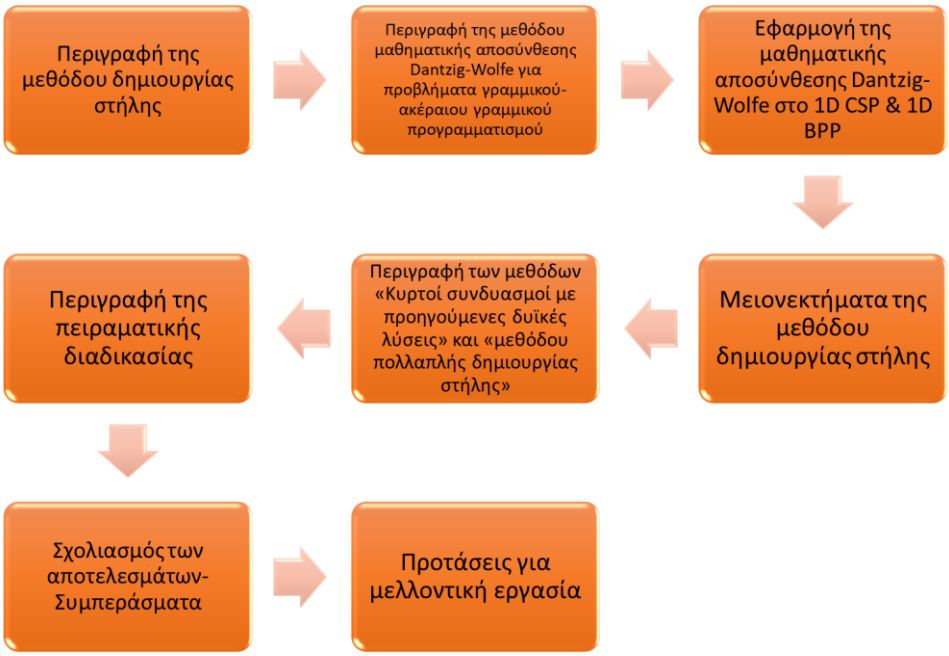
Για την μελέτη της DWD αρχικά αναπτύσσεται το θεώρημα αναπαράστασης Minkowski-Weyl και η μέθοδος δημιουργίας στήλης. Στη συνέχεια η DWD περιγράφεται για προβλήματα γραμμικού, ακέραιου γραμμικού προγραμματισμού με ειδική δομή και τέλος για προβλήματα δυαδικού ακέραιου γραμμικού προγραμματισμού.

Η DWD εφαρμόζεται σε δύο δημοφιλή και στενά συνδεδεμένα προβλήματα βελτιστοποίησης, όπου διακρίνεται καθαρά η διαδικασία δημιουργίας στήλης. Αυτά τα

προβλήματα είναι: (α) το μονοδιάστατο πρόβλημα κοπής αποθέματος (1 Dimensional Cutting Ctock Problem, 1D-CSP) και (β) το μονοδιάστατο πρόβλημα συσκευασίας κάδου (1 Dimensional Bin Packing Problem, 1D-BPP).

Η επιτάχυνση DWD, γίνεται μέσω επιτάχυνσης της μεθόδου δημιουργίας στήλης. Η μέθοδος δημιουργίας στήλης που χρησιμοποιείται, είναι μία επιτυχημένη τεχνική επίλυσης προβλημάτων μεγάλης κλίμακας αλλά έχει το σημαντικό μειονέκτημα αργής σύγκλισης. Η κυριότερη αιτία είναι η ασταθής συμπεριφορά των δυϊκών λύσεων. Έχει καταβληθεί μεγάλη προσπάθεια μελέτης και εύρεσης τεχνικών βελτίωσης της απόδοσής της. Υπάρχουν διάφορες τεχνικές επιτάχυνσης της μεθόδου δημιουργίας στήλης. Στην παρούσα εργασία προτείνεται μία νέα προσέγγιση της μεθόδου δημιουργίας πολλαπλών στηλών (Multiple Column Generation, MCG) – όπου οι στήλες που προστίθενται στο περιορισμένο κύριο πρόβλημα προέρχονται α) μία στήλη από την επίλυση του κλασικού υπο-προβλήματος και β) από ένα τροποποιημένο υπο-πρόβλημα του οποίου οι δυϊκές λύσεις είναι γραμμικός συνδυασμός προηγούμενων δυϊκών λύσεων.

Για την σύγκριση, την αξιολόγηση και την εξαγωγή συμπερασμάτων που αφορούν την απόδοση της κλασικής μεθόδου δημιουργίας στήλης και της προτεινόμενης μεθόδου, αναπτύχθηκαν κώδικες σε γλώσσα προγραμματισμού C++ που χρησιμοποιούν τις βιβλιοθήκες βελτιστοποίησης του λογισμικού βελτιστοποίησης CPLEX IBM ILOG CPLEX Optimization Studio®, ενός δημοφιλούς υπολογιστικού πακέτου βελτιστοποίησης τόσο στον ακαδημαϊκό όσο και στον βιομηχανικό χώρο [19]. Τα παραδείγματα δοκιμών (benchmarks), προέρχονται από την βιβλιοθήκη BPPLIB και είναι διαθέσιμα στην ιστοσελίδα [20].



Εικόνα 1-4: Διάγραμμα ροής των στόχων της διπλωματικής εργασίας.

1.2 Βιβλιογραφική Ανασκόπηση

Υπάρχουν αρκετές μελέτες που αφορούν την αποσύνθεση Dantzig-Wolfe. Το 1960 [15] και το 1961 [16], οι G. Dantzig και P. Wolfe πρότειναν την αρχή της αποσύνθεσης Dantzig-Wolfe, που αφορά τη χρήση της σε προβλήματα γραμμικού προγραμματισμού. Από τότε υπάρχουν αρκετοί ερευνητές που προσπάθησαν να την εφαρμόσουν σε πρακτικά προβλήματα διάφορων επιστημονικών τομέων. Η πρώτη πρακτική εφαρμογή της, αναφέρθηκε από τους E.M.L. Beale, P.A.B. Hughes και R.E. Small, το 1965 και αφορούσε την εφαρμογή της στη βιομηχανία πετρελαίου [21]. Ο αλγόριθμος υλοποιήθηκε ως εσωτερική επέκταση στο εμπορικό λογισμικό μαθηματικού προγραμματισμού LP/90/94. Το λογισμικό είχε θεωρητική χωρητικότητα 100 blocks, 10^5 περιορισμούς, 1000 περιορισμούς σε οποιοδήποτε block και έναν απεριόριστο αριθμό μεταβλητών. Το καλύτερο αποτέλεσμα της προσπάθειάς τους επιτεύχθηκε σε ένα πρόβλημα 450 περιορισμών, όπου βρέθηκε ότι η μέθοδος αποσύνθεσης Dantzig-Wolfe απέδωσε. Το 1974 οι H.P. Williams και A.C. Redwood [22], υλοποίησαν την μέθοδο μαθηματικής αποσύνθεσης Dantzig-Wolfe για ένα πρόβλημα στη βιομηχανία τροφίμων, με 1800 περιορισμούς και 3200 μεταβλητές. Το λογισμικό που χρησιμοποιήθηκε ήταν το MPSX της IBM. Τη δεκαετία του 1980, οι Ho J. και Sundarraj R., ανέπτυξαν μία υλοποίηση της αποσύνθεσης Dantzig-Wolfe στη γλώσσα προγραμματισμού FORTRAN, με το όνομα DECOMP που δημοσιοποιήθηκε τελικά το 1989 [23]. Το 2001 ο Tebboth J. στην

διδακτορική του διατριβή, υλοποίησε μια υπολογιστική αξιολόγηση της αποσύνθεσης Dantzig-Wolfe [18]. Η εργασία του αναλύει τα αποτελέσματα με μεγάλη λεπτομέρεια και προσφέρει πληροφορίες για τις κατηγορίες προβλημάτων που μπορεί να εφαρμοστεί η συγκεκριμένη μέθοδος αποσύνθεσης. Αν και υπάρχει εγχειρίδιο χρήσης για το λογισμικό που ανέπτυξε στο παράρτημα της εργασίας του, ωστόσο δεν έχει δημοσιοποιηθεί. Το 2013, ο Rios J., πραγματοποίησε μία παράλληλη υλοποίηση της αποσύνθεσης Dantzig-Wolfe σε γλώσσα προγραμματισμού C για ένα πρόβλημα διαχείρισης της εναέριας κυκλοφορίας [24].

Επιπλέον εφαρμογές της αποσύνθεσης Dantzig-Wolfe σε προβλήματα της επιχειρησιακής έρευνας, αναφέρονται στις εργασίες των François Vanderbeck και Laurence A. Wolsey [25], Gerald Gamrath και Marco E. Lübbecke [26]. Οι François Vanderbeck και Martin W.P. Savelsbergh [27] παρουσιάζουν μία αναδιατύπωση της αποσύνθεσης Dantzig-Wolfe σε προβλήματα μικτού ακέραιου προγραμματισμού και την εφαρμόζουν στο περιορισμένο πρόβλημα μεγέθους παρτίδας πολλαπλών στοιχείων (capacitated multi-item lot-sizing problem). Ο Raf Jans [28] ταξινομεί τις αναδιατυπώσεις Dantzig-Wolfe σε εννέα κατηγορίες για προβλήματα δυαδικού ακέραιου γραμμικού προγραμματισμού και περιγράφει πότε είναι δυνατή η άμεση επιβολή δυαδικών περιορισμών στο κύριο πρόβλημα. Οι Brian Kallehauge, Jesper Larsen, Oli B.G. Madsen και Marius M. Solomon [29] χρησιμοποιούν την αποσύνθεση Dantzig-Wolfe για την επίλυση του προβλήματος δρομολόγησης στόλου οχημάτων με περιορισμένα χρονικά παράθυρα (Vehicle Routing Problem with Time Windows, VRPTW).

Άλλες αναφορές για εφαρμογές της αποσύνθεσης Dantzig-Wolfe σε προβλήματα της επιχειρησιακής έρευνας είναι:

- Εφαρμογή στο πρόβλημα της χωροθέτησης εγκαταστάσεων (facility location), όπως αυτή των Tao Wu, Zhongshun Shi, Zhe Liang, Xiaoning Zhang και Canrong Zhang [30], οι οποίοι αποδεικνύουν ότι τα υποπροβλήματα του κοστολόγησης της αποσύνθεσης, σχετίζονται με προβλήματα μεγέθους παρτίδας χωρίς χωρητικότητα με την ιδιότητα Wagner-Whitin. Αυτή η ιδιότητα χρησιμοποιείται για βελτίωση της απόδοσης της μεθόδου δημιουργίας στήλης της αποσύνθεσης Dantzig-Wolfe.
- Εφαρμογή στο πρόβλημα καταμερισμού-χρονοπρογραμματισμού εργασιών (Job Shop Scheduling ή Job Shop Problem, JSP), όπως αυτή των Sylvie Gélinas και François Soumis [31], οι οποίοι παρουσιάζουν μια διατύπωση για το πρόβλημα αυτό με βάση την αποσύνθεση Dantzig-Wolfe με ένα υπο-πρόβλημα

για κάθε μηχανή. Κάθε υπο-πρόβλημα είναι ένα πρόβλημα αλληλουχίας σε ένα μόνο μηχάνημα με χρονικά παράθυρα.

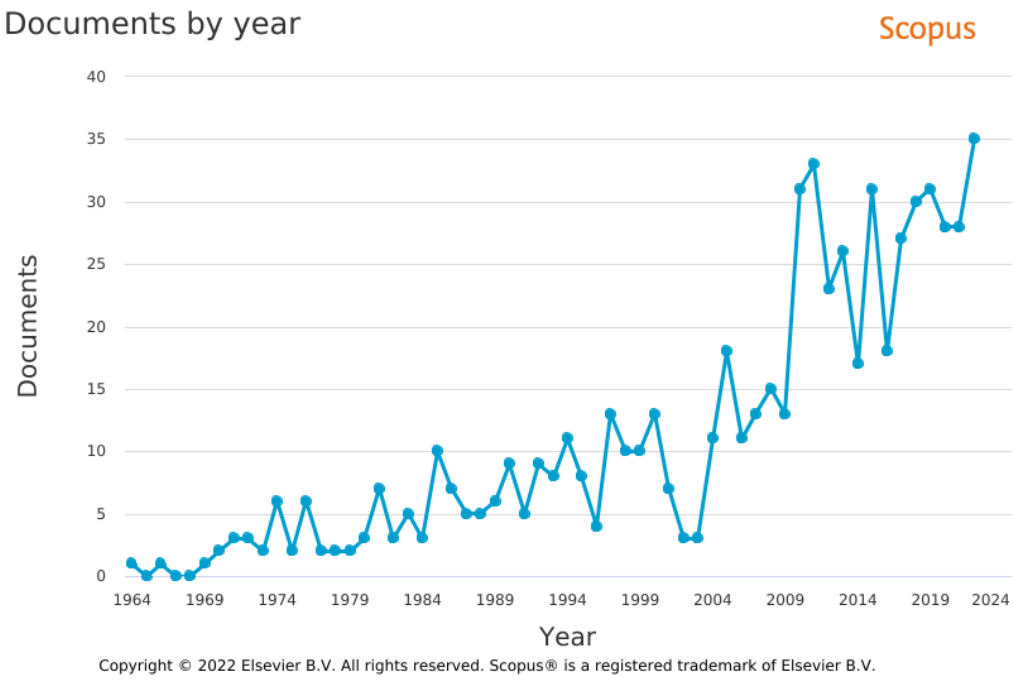
- Εφαρμογή στο πρόβλημα προγραμματισμού των πληρωμάτων αεροπορικών εταιρειών (airline crew pairing), όπως των Guy Desaulniers, Jacques Desrosiers, Michel Gamache και François Soumis [32]. Στη συγκεκριμένη εργασία, εξετάζονται ορισμένα προβλήματα προγραμματισμού του πληρώματος που αντιμετωπίζουν οι αεροπορικές εταιρείες τόσο κατά τη φάση σχεδιασμού όσο και κατά τη φάση λειτουργίας. Λαμβάνοντας υπόψη ένα πρόγραμμα πτήσεων και μια ανάθεση στόλου, η διαδικασία σχεδιασμού συνίσταται πρώτα στην επίλυση ενός προβλήματος δημιουργίας πληρώματος για κάθε στόλο αεροσκαφών και στη συνέχεια στην κατασκευή ενός μηνιαίου προγράμματος εργασίας για κάθε μεμονωμένο μέλος του πληρώματος. Σε επιχειρησιακό επίπεδο, ορισμένα «ζεύγη» πληρωμάτων, πρέπει να τροποποιηθούν προκειμένου να αντισταθμιστούν ορισμένες διακοπές που συμβαίνουν κατά τη διάρκεια των επιχειρήσεων. Το πρόβλημα προγραμματισμού του επιχειρησιακού πληρώματος περιλαμβάνει την πραγματοποίηση αυτών των τροποποιήσεων.
- Προγραμματισμός ροής κυκλοφορίας οχημάτων (Traffic Flow Scheduling) των Joseph Rios και Kevin Ross [33] οι οποίοι εφαρμόζουν την αποσύνθεση Dantzig-ένα πρόβλημα ακέραιου προγραμματισμού και τα υπο-προβλήματα αυτής της αποσύνθεσης επιλύονται παράλληλα μέσω ανεξάρτητων υπολογιστών. Τα πειραματικά αποτελέσματα δείχνουν ότι καθώς αυξάνεται ο αριθμός των υπο-προβλημάτων/πλήθους υπολογιστών, η ποιότητα της λύσης, η σύγκλιση και ο χρόνος εκτέλεσης βελτιώνονται.

Όσον αφορά τις τεχνικές επιτάχυνσης της μεθόδου δημιουργίας στήλης, υπάρχουν αρκετές εργασίες, κυρίως για μεθόδους σταθεροποίησής της. Οι Olivier du Merle, Daniel Villeneuve, Jacques Desrosiers και Pierre Hansen [34], προτείνουν έναν αλγόριθμο σταθεροποίησης των δυϊκών λύσεων, που επιταχύνει τη διαδικασία επίλυσης παραμένοντας εντός του πλαισίου γραμμικού προγραμματισμού. Οι O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot και F. Vanderbeck [35], προτείνουν τη μέθοδο δέσμης (bundle method) ως μία μέθοδο σταθεροποίησης της κλασικής μεθόδου δημιουργίας στήλης, την εφαρμόζουν σε

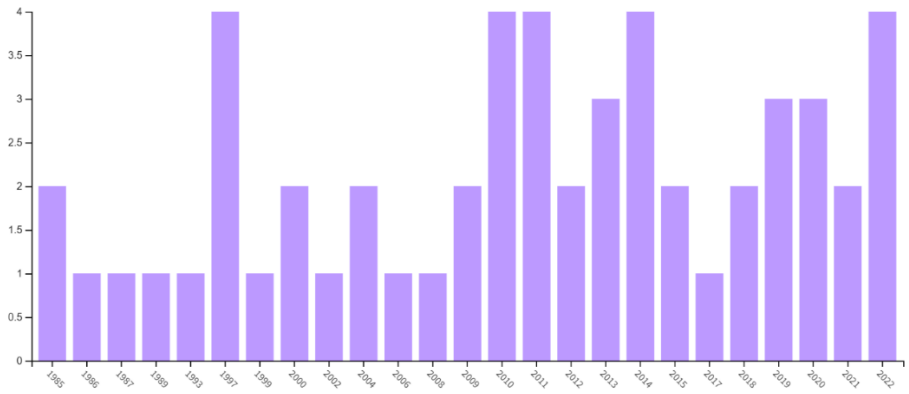
πέντε παραδείγματα (πρόβλημα κοπής του αποθέματος, πρόβλημα χρωματισμού κορυφής, πρόβλημα μεγέθους παρτίδας πολλαπλών ειδών, πρόβλημα δρομολόγησης στόλου οχημάτων με περιορισμένη χωρητικότητα και το πρόβλημα του πλανόδιου πωλητή) και συγκρίνουν τα αποτελέσματα. Ο Marco E. Lübbecke [36], αναφέρει διάφορες τεχνικές που σταθεροποιούν τον κλασικό αλγόριθμο δημιουργίας στήλης (μέθοδοι πολυωνυμικής ποινής, δέσμης, εσωτερικών σημείων). Οι Chungmok Lee και Sungsoo Park [37] συνεχίζοντας το έργο του B. Betrò [38], προτείνουν μία μέθοδο βασισμένη στο κέντρο Chebyshev ως τεχνική σταθεροποίησης του κλασικού αλγόριθμου δημιουργίας στήλης. Σημαντικές αναφορές για την σταθεροποίηση της μεθόδου αποσύνθεσης Dantzig-Wolfe είναι αυτή των Antonio Frangioni και Bernard Gendron [39], που προτείνουν ένα σχήμα μιας σταθεροποιημένης μορφής του αλγορίθμου αποσύνθεσης Dantzig-Wolfe, εφαρμόζουν το νέο σχήμα στο πρόβλημα σχεδιασμού δικτύου με χωρητικότητα πολλαπλών εμπορευμάτων και συγκρίνουν τα υπολογιστικά αποτελέσματα με τη χρήση του λογισμικού IBM ILOG CPLEX Optimization Studio®. Ο Paul Wentges περιγράφει ένα σχήμα σταθεροποίησης του αλγορίθμου αποσύνθεσης Dantzig-Wolfe, τη λεγόμενη σταθεροποιημένη αποσύνθεση Dantzig-Wolfe [40]. Ο P. Neame στην εργασία του [41], προτείνει μία μέθοδο σταθεροποίησης της κλασικής μεθόδου δημιουργίας στήλης, εξομαλύνοντας τις δυϊκές τιμές και την εφαρμόζει στο δυαδικό πρόβλημα κοπής αποθέατος. Ο Phillipe Mahey προτείνει έναν αλγόριθμο υπο-βαθμίδας (subgradient algorithm) και περιγράφει τον τρόπο με τον οποίο οι πληροφορίες που φέρονται μέσω αλυσιδωτών υποβαθμίσεων, βελτιώνουν την απόδοση της μεθόδου αποσύνθεσης Dantzig-Wolfe [42]. Μία από τις πιο ελπιδοφόρες προσεγγίσεις σταθεροποιημένης μεθόδου δημιουργίας στήλης έχει προταθεί από τον V. Carvalho [43]. Η μέθοδος προσθέτει έναν πολυωνυμικό αριθμό δυϊκών κοπών (αρχικές στήλες) στο περιορισμένο κύριο πρόβλημα. Ο Carvalho πρότεινε ένα σύνολο αδύναμων δυϊκών ανισοτήτων για το μονοδιάστατο πρόβλημα κοπής αποθέματος, οι οποίες δεν «κόβουν» καμία βέλτιστη λύση του δυϊκού προβλήματος. Εκτός από τις μεθόδους σταθεροποίησης, ενδιαφέρον, παρουσιάζει η περίπτωση βελτίωσης της απόδοσης της κλασικής μεθόδου δημιουργίας στήλης με εντατικοποίηση και διαφοροποίηση. Βασική ιδέα αυτών των τροποποιημένων μεθόδων, είναι η προσθήκη στο περιορισμένο κύριο πρόβλημα ενός συνόλου στηλών αρνητικού κόστους σε κάθε επανάληψη. Ωστόσο, υπάρχουν ελάχιστες αναφορές που αφορούν αυτές, όπως είναι οι [44], [45].

Όπως είναι φανερό από τα προηγούμενα, υπάρχει έντονο ερευνητικό ενδιαφέρον για την μαθηματική αποσύνθεση Dantzig-Wolfe. Ένα δείγμα μπορεί να οπτικοποιηθεί από μια

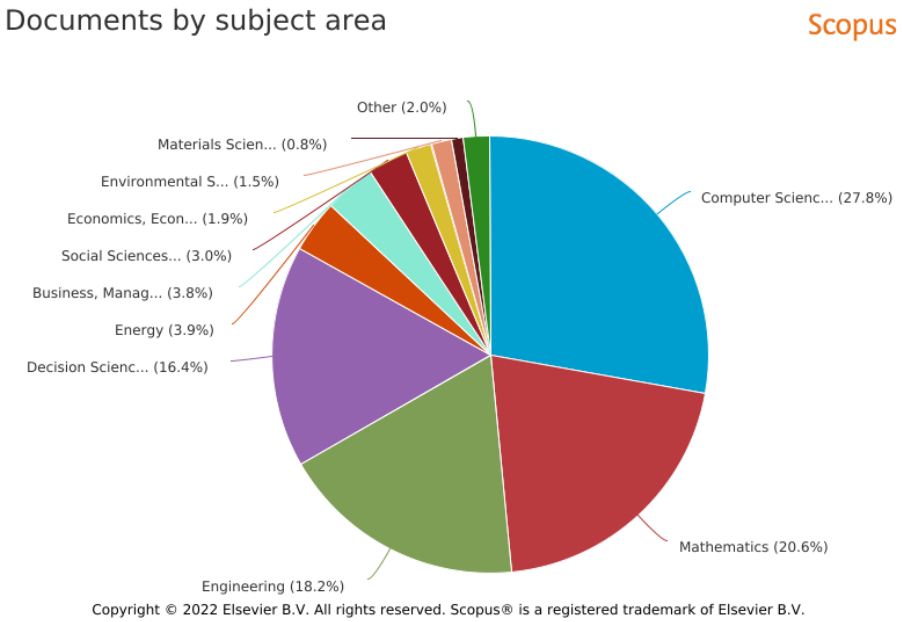
αναζήτηση στις βάσεις βιβλιογραφικών δεδομένων «Scopus» και «Web of Science» για άρθρα που έχουν στον τίτλο τους τον όρο «Dantzig-Wolfe decomposition».



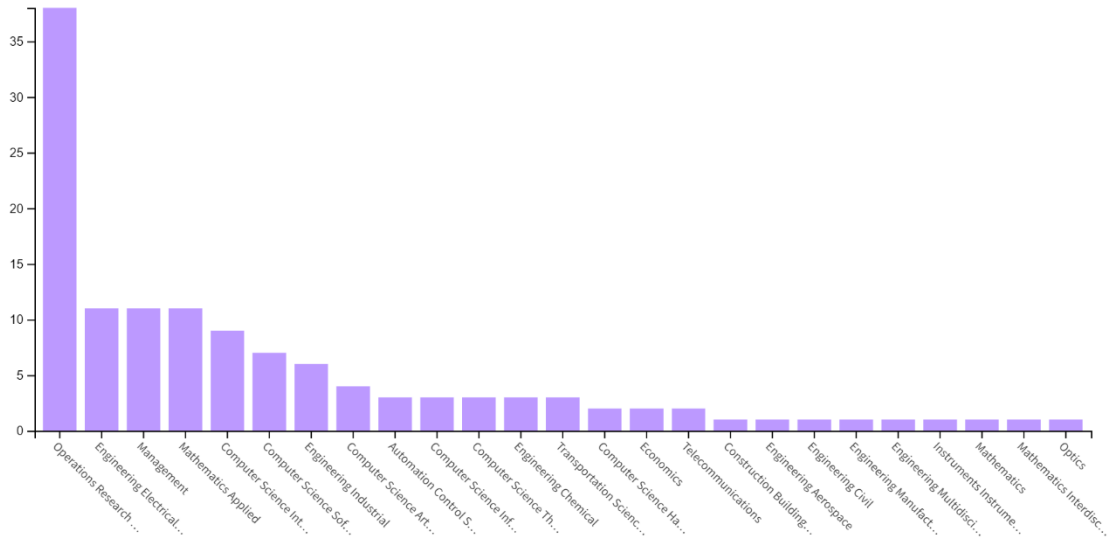
Εικόνα 1-5: Αριθμός δημοσιευμένων εγγράφων ανά έτος με τίτλο τους «μαθηματική αποσύνθεση Dantzig-Wolfe», 1964-2022 (Scopus) [46].



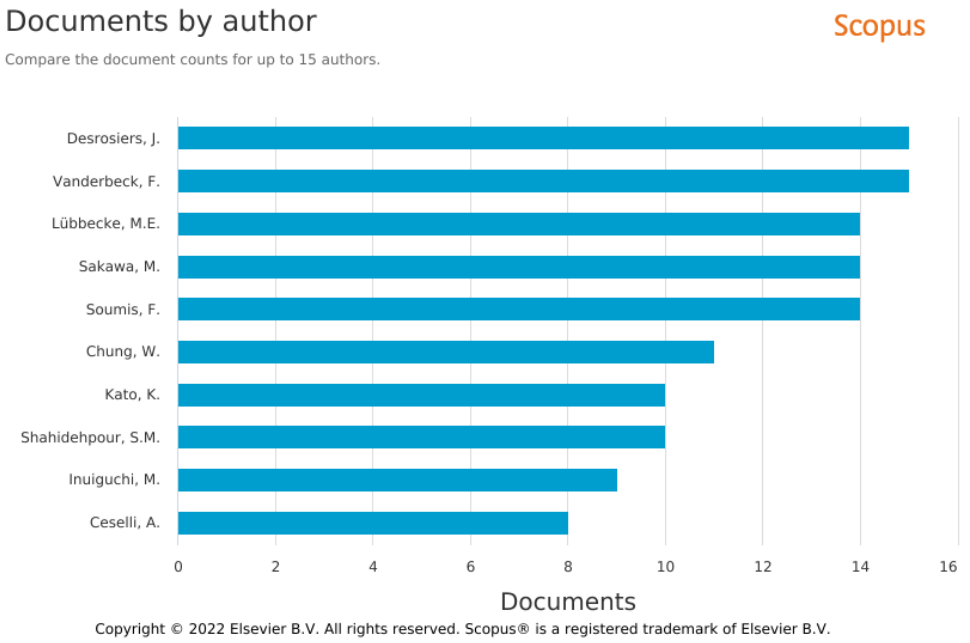
Εικόνα 1-6: Αριθμός δημοσιευμένων εγγράφων ανά έτος με τίτλο τους «μαθηματική αποσύνθεση Dantzig-Wolfe», 1985-2022 (Web of Science) [47].



Εικόνα 1-7: Ποσοστά δημοσιευμένων εγγράφων που αφορούν την μαθηματική αποσύνθεση Dantzig-Wolfe ανά επιστημονικό τομέα, 1964-2022 (Scopus) [46].

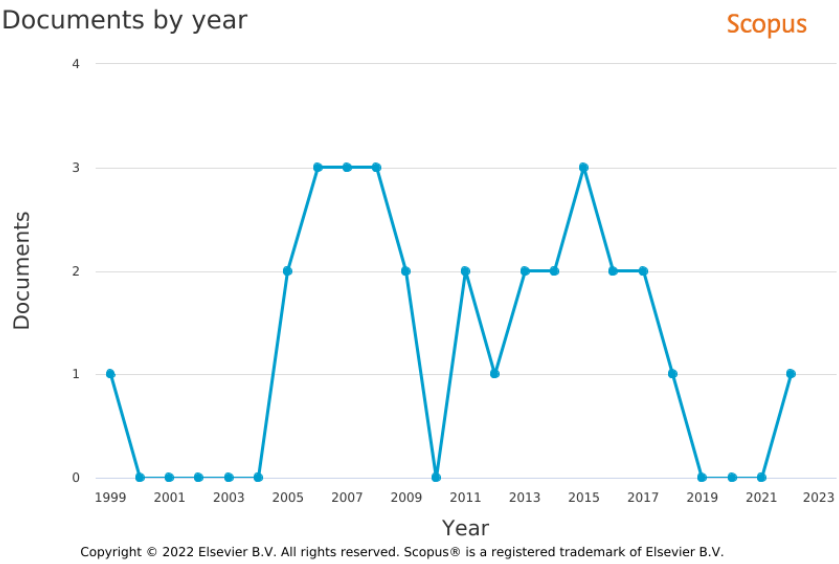


Εικόνα 1-8: Αριθμός δημοσιευμένων εγγράφων που αφορούν την μαθηματική αποσύνθεση Dantzig-Wolfe ανά επιστημονικό τομέα (Web of Science) [47].

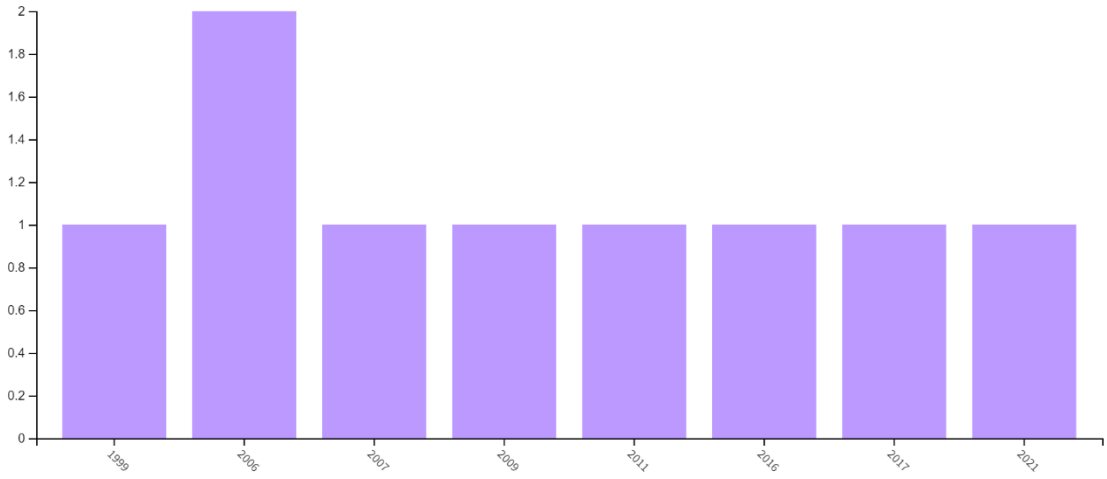


Εικόνα 1-9: Αριθμός δημοσιευμένων εγγράφων ανά συγγραφέα που αφορούν την μαθηματική αποσύνθεση Dantzig-Wolfe, 1964-2022 (Scopus) [46].

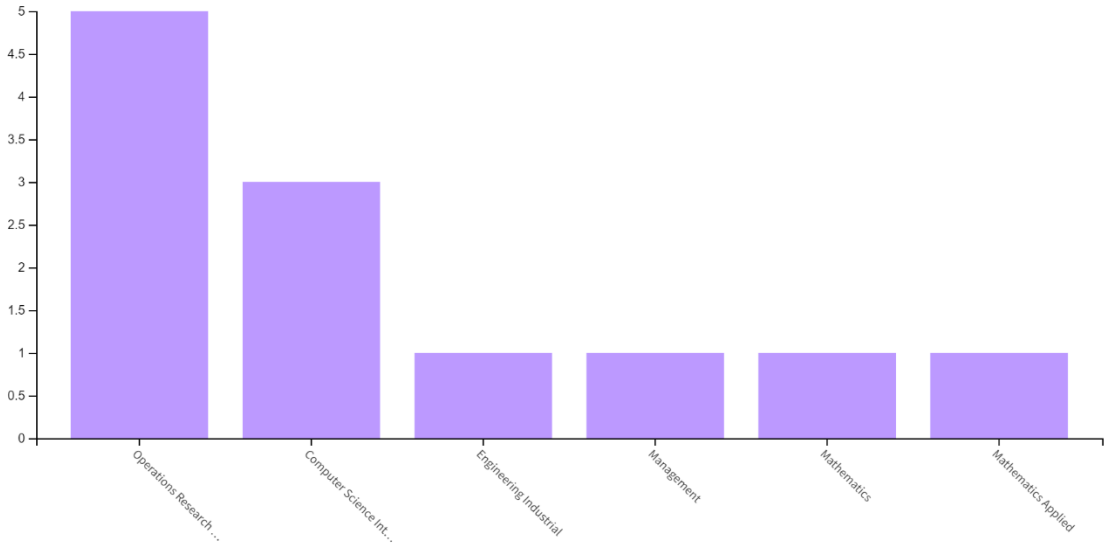
Για τις μεθόδους επιτάχυνσης (κυρίως σταθεροποίησης) της μεθόδου δημιουργίας στήλης που χρησιμοποιεί η αποσύνθεση Dantzig-Wolfe υπάρχει ανά περιόδους ερευνητικό ενδιαφέρον. Ένα δείγμα μπορεί να οπτικοποιηθεί από μια αναζήτηση στις βάσεις βιβλιογραφικών δεδομένων «Scopus» και «Web of Science» για άρθρα που έχουν στον τίτλο τους «stabilized column generation».



Εικόνα 1-10: Αριθμός δημοσιευμένων εγγράφων ανά έτος με τίτλο «σταθεροποιημένη μέθοδος δημιουργίας στήλης», 1999-2022 (Scopus) [46].



Εικόνα 1-11: Αριθμός δημοσιευμένων εγγράφων που αφορούν την σταθεροποιημένη μέθοδος δημιουργίας στήλης ανά έτος, 1999-2021 (Web of Science) [47].



Εικόνα 1-12: Συχνότητα εφαρμογής της σταθεροποιημένης μεθόδου δημιουργίας στήλης σε επιστημονικούς τομείς (Web of Science) [47].

1.3 Οργάνωση Διπλωματικής Εργασίας

Το υπόλοιπο της διπλωματικής εργασίας χωρίζεται σε δύο μέρη: Το θεωρητικό μέρος που αποτελείται από τα Κεφάλαια 2-5 και το υπολογιστικό μέρος που αποτελείται από τα Κεφάλαια 6-7.

Όσον αφορά το θεωρητικό μέρος:

Στο **Κεφάλαιο 2**, γίνεται μία ανασκόπηση κάποιων βασικών εννοιών της θεωρίας πολυέδρων. Στη συνέχεια, περιγράφεται το θεώρημα Minkowski-Weyl και τέλος αναφέρεται η έννοια του μειωμένου κόστους και η ερμηνεία της.

Στο **Κεφάλαιο 3**, περιγράφονται η μέθοδος δημιουργίας στήλης και η αρχή της μαθηματικής αποσύνθεσης Dantzig-Wolfe. Στη συνέχεια, η ιδέα της μαθηματικής αποσύνθεσης Dantzig-Wolfe επεκτείνεται σε προβλήματα ακέραιου γραμμικού προγραμματισμού και σε δυαδικά προβλήματα ακέραιου γραμμικού προγραμματισμού.

Στο **Κεφάλαιο 4**, η μαθηματική αποσύνθεση Dantzig-Wolfe εφαρμόζεται στο μονοδιάστατο πρόβλημα κοπής αποθέματος και στο μονοδιάστατο πρόβλημα συσκευασίας κάδου.

Στο **Κεφάλαιο 5**, αναφέρονται διάφορες τεχνικές επιτάχυνσης της μεθόδου δημιουργίας στήλης. Από αυτές, περιγράφονται κάποιες μέθοδοι ποινής, μέθοδοι εξομάλυνσης δυϊκών μεταβλητών και αναπτύσσεται η μέθοδος πολλαπλής δημιουργίας στήλης.

Όσον αφορά το υπολογιστικό μέρος:

Στο **Κεφάλαιο 6**, περιγράφεται η μεθοδολογία που ακολουθήθηκε για την υλοποίηση υπολογιστικών πειραμάτων. Αρχικά, αναφέρονται κάποιες υποθέσεις που έγιναν για την εκτέλεση των υπολογιστικών πειραμάτων, στη συνέχεια περιγράφονται η βιβλιοθήκη BPPLIB, τα διαθέσιμα δοκιμαστικά προβλήματα και τέλος το λογισμικό που χρησιμοποιήθηκε.

Στο **Κεφάλαιο 7**, παρουσιάζονται τα αριθμητικά αποτελέσματα και σχολιάζεται η απόδοση της κάθε μεθόδου (κλασικής μεθόδου δημιουργίας στήλης, μεθόδου δημιουργίας πολλαπλών στηλών με τη χρήση κυρτών συνδυασμών με προηγούμενες δυϊκές λύσεις, για διάφορες τιμές της παραμέτρου επιπέδου εξομάλυνσης, μεθόδου δημιουργίας πολλαπλών στηλών, για διάφορες τιμές της παραμέτρου επιπέδου εξομάλυνσης και διαφορετικό αριθμό δημιουργούμενων στηλών).

Τέλος, στο **Κεφάλαιο 8**, αναφέρονται τα συμπεράσματα της διπλωματικής εργασίας και δίνονται προτάσεις για μελλοντική εργασία.

Κεφάλαιο 2. Βασικές έννοιες της θεωρίας πολύεδρων-θεώρημα αναπαράστασης MinKowski-Weyl-μειωμένο κόστος

2.1 Εισαγωγή

Στο παρόν Κεφάλαιο, γίνεται μία ανασκόπηση βασικών εννοιών της θεωρίας πολυέδρων, του θεωρήματος Minkowski-Weyl και του μειωμένου κόστους.

2.2 Βασικές έννοιες της θεωρίας πολυέδρων

2.2.1 Πολύεδρο

Τα προβλήματα του γραμμικού προγραμματισμού, ασχολούνται με εφικτά σύνολα στο χώρο $\mathbb{R}^n$ που ορίζονται από γραμμικές εξισώσεις ή ανισώσεις. Τέτοια σύνολα ονομάζονται πολύεδρα [18], [48]–[50].

Πολύεδρο (polyhedron) P ορίζεται ένα σύνολο σημείων που ικανοποιούν έναν πεπερασμένο αριθμό m γραμμικών ανισοτήτων ή εξισώσεων, όπως δηλώνουν οι ακόλουθες σχέσεις [18]:

Ανισοτήτων:

$$P = \{x \in \mathbb{R}^n \mid Ax \geq b\} \tag{2-1}$$

ή

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\} \tag{2-2}$$

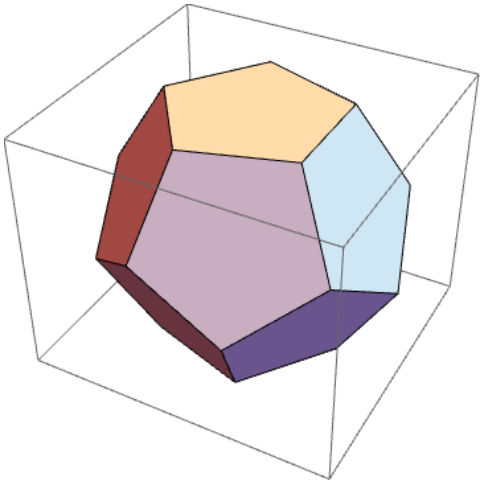
και

Εξισώσεων:

$$P = \{x \in \mathbb{R}^n \mid Ax = b\} \tag{2-3}$$

Όπου, A είναι ένας πίνακας διαστάσεων $m \times n$ και b είναι ένα διάνυσμα διαστάσεων $m \times 1$ στον χώρο $\mathbb{R}^n$.

Στην **Εικόνα 2-1**, παρουσιάζεται ένα παράδειγμα ενός δωδεκάεδρου (χώρος $\mathbb{R}^3$), για τη δημιουργία του οποίου χρησιμοποιήθηκε το υπολογιστικό πακέτο Wolfram Mathematica®. Ο αντίστοιχος κώδικας βρίσκεται στο **Παράρτημα Β.1**.

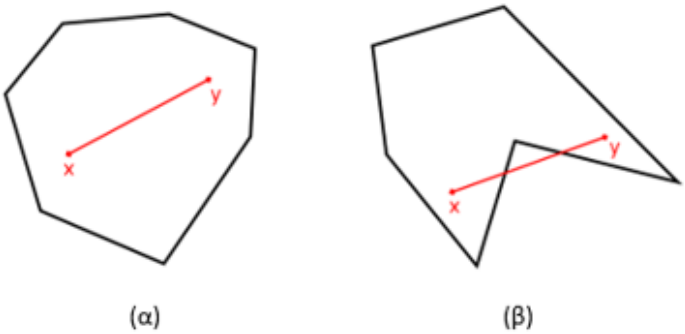


Εικόνα 2-1:Παράδειγμα ενός πολύεδρου: Δωδεκάεδρο.

2.2.2 Κυρτό σύνολο

Ένα σύνολο $S \subseteq \mathbb{R}^n$ ονομάζεται κυρτό (convex set) αν για οποιαδήποτε στοιχεία $x, y \in S$ και για $\lambda \in \mathbb{R}$, με $0 \leq \lambda \leq 1$, το νέο στοιχείο $\lambda x + (1-\lambda)y \in S$ [18], [49].

Το $\lambda x + (1-\lambda)y$ είναι το ευθύγραμμο τμήμα που ενώνει τα δύο σημεία. Στην **Εικόνα 2-2** [51], παρουσιάζεται ένα παράδειγμα μιας κυρτής και μιας μη κυρτής περιοχής.

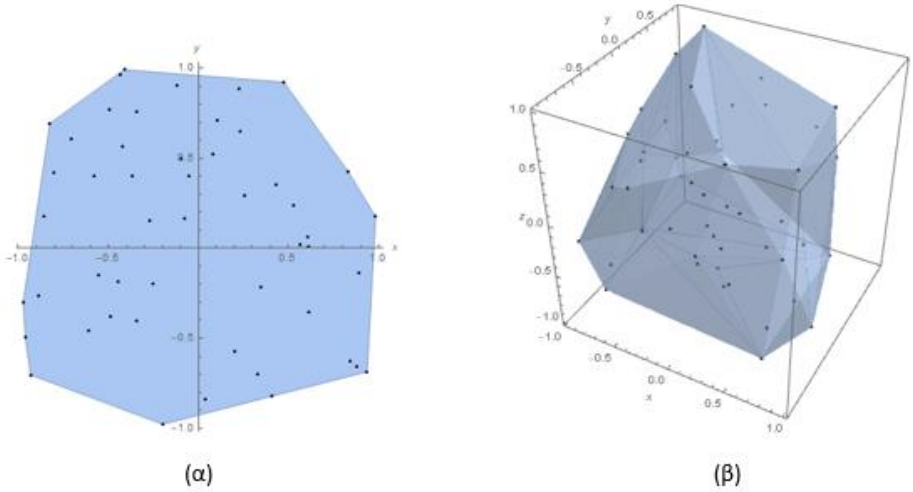


Εικόνα 2-2: (α) ένα παράδειγμα μιας κυρτής περιοχής (convex region) και (β) ένα παράδειγμα μιας μη-κυρτής περιοχής (non–convex region).

2.2.3 Κυρτή θήκη

Η κυρτή θήκη (convex hull) ενός συνόλου S, συμβολίζεται ως $conv\ (S)$ και είναι το σύνολο όλων των σημείων που είναι κυρτοί συνδυασμοί των σημείων του S, [48]–[50].

Στην **Εικόνα 2-3**, παρουσιάζονται δύο παραδείγματα τυχαίων κυρτών θηκών που προέκυψαν με τη χρήση του υπολογιστικού πακέτου Mathematica®. Οι κώδικες, παρουσιάζονται στο **Παράρτημα Β.2** και στο **Παράρτημα Β.3** αντίστοιχα.



Εικόνα 2-3: Η Κυρτή θήκη (convex hull) (α) ενός 2D τυχαίου συνόλου (δημιουργήθηκαν 50 τυχαία σημεία σε 2D επίπεδο από τυχαίους πραγματικούς αριθμούς στο διάστημα [-1,1]) και (β) ενός 3D τυχαίου συνόλου (δημιουργήθηκαν 50 τυχαία σημεία σε επίπεδο 3D από τυχαίους πραγματικούς αριθμούς στο διάστημα [-1,1]).

2.2.4 Κυρτό πολύτοπο

Ένα κυρτό πολύεδρο που είναι φραγμένο ονομάζεται κυρτό πολύτοπο (convex polytope) [18], [48]–[50].

2.2.5 Κυρτός συνδυασμός

Ένα σημείο $x \in \mathbb{R}^n$ είναι ένας κυρτός συνδυασμός (convex combination) ενός συνόλου $S \subseteq \mathbb{R}^n$, αν αυτό το σημείο μπορεί να εκφραστεί ως $x = \sum_i \lambda_i x_i, \lambda_i \geq 0$ για ένα πεπερασμένο υποσύνολο $\{x_1, x_2, \dots, x_k\}$ του συνόλου S, όπου $\sum_i \lambda_i = 1$ [18], [48]–[50].

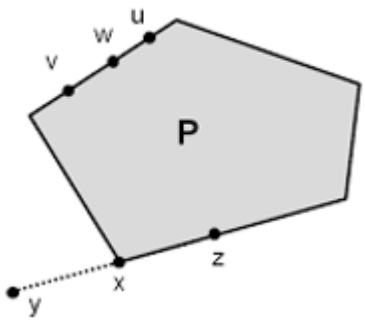
2.2.6 Ακραίο σημείο

Ένα σημείο x λέγεται ακραίο σημείο (extreme point) ενός κυρτού συνόλου S, αν δεν μπορεί να γραφτεί σαν κυρτός συνδυασμός δύο διακεκριμένων σημείων του συνόλου S, [18], [48]–[50]. Δηλαδή:

$$x \text{ ακραίο σημείο του } S \Leftrightarrow \nexists\ (x_1\ \kappa\alpha\iota\ x_2) \in S \text{ τέτοια ώστε:}$$

$$x = \lambda x_1 + (1-\lambda)x_2, \text{ με } \lambda \in \mathbb{R}, 0 < \lambda < 1$$

Επομένως, ακραίο σημείο ενός πολυέδρου είναι κάποιο γωνιακό σημείο του πολυέδρου, που ονομάζεται κορυφή (vertex) του πολυέδρου. Στην **Εικόνα 2-4** [52], δίνεται ένα παράδειγμα αν ένα σημείο είναι ακραίο ή όχι.



Εικόνα 2-4: Το σημείο w δεν είναι ακραίο σημείο γιατί $w \in (u, v)$. Το σημείο x είναι ακραίο σημείο αν $x = \lambda y + (1-\lambda)z$ με $0 < \lambda < 1$ για $y, z \neq x$ τότε είτε $y \notin P$ είτε $z \notin P$.

2.2.7 Ακτίνες

Έστω ένα μη κενό πολύεδρο (non-empty polyhedron) $P = \{x \in \mathbb{R}^n \mid Ax \geq b \text{ ή } Ax \leq b \text{ ή } Ax = b\}$. Ένα διάνυσμα $r \neq 0, r \in \mathbb{R}^n$, ονομάζεται ακτίνα (ray) του πολυέδρου P αν $Ar \geq 0$ ή $Ar \leq 0$ ή $Ar = 0$. Δύο ακτίνες $r\ \kappa\alpha\iota\ s$ του πολυέδρου P είναι διακεκριμένες αν δεν υπάρχει μία σταθερά $\lambda > 0$ τέτοια ώστε $s = \lambda r$. Στην περίπτωση που υπάρχει μία σταθερά $\lambda > 0$ τέτοια ώστε $s = \lambda r$, οι ακτίνες $r\ \kappa\alpha\iota\ s$ ταυτίζονται [18].

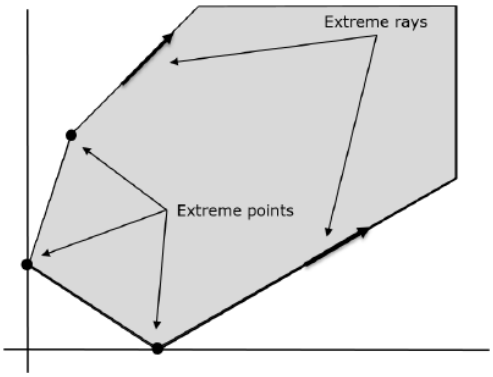
Ένας άλλος ορισμός είναι ο ακόλουθος:

Αν $P^0 = \{r \in \mathbb{R}^n : Ar \leq 0\}$. Κάθε $r \in P^0\{0\}$ είναι ακτίνα του P (είναι ένας κυρτός κώνος) [53].

2.2.8 Ακραίες Ακτίνες

Μία ακτίνα r ενός πολύεδρου P είναι ακραία (extreme ray), αν δεν υπάρχουν δύο ακτίνες $r^1, r^2 \in P^0$ και ένα μ $(r^1 \neq \lambda r^2, \forall\ \lambda > 0\ \kappa\alpha\iota\ 0 < \mu < 1)$ έτσι ώστε: $r = \mu\, r^1 + (1-\mu)r^2$ [18], [53].

Για μία έδρα ενός πολυέδρου P, μία ακτίνα r είναι ακραία αν και μόνο αν το σύνολο $\{x + \lambda r \mid \lambda \geq 0\}$ είναι μία έδρα του πολυέδρου P για μερικά ακραία σημεία [18].



Εικόνα 2-5: Τα ακραία σημεία και οι ακραίες ακτίνες ενός μη φραγμένου πολύεδρου [54].

2.2.9 Ιδιότητες κυρτών συνόλων

Στην ενότητα αυτή, αναφέρονται μερικές βασικές ιδιότητες των κυρτών συνόλων, χωρίς απόδειξη [18], [48]–[50]:

- Ένα πολύεδρο είναι ένα κυρτό σύνολο.
- Η τομή κυρτών συνόλων είναι κυρτό σύνολο.
- Το σύνολο όλων των κυρτών συνδυασμών ενός πεπερασμένου αριθμού σημείων ορίζει το κυρτό πολύεδρο που παράγεται από αυτά τα σημεία.
- Ένα πολύεδρο έχει ένα πεπερασμένο αριθμό ακραίων σημείων και ακραίων ακτίνων.

2.3 Θεώρημα αναπαράστασης Minkowski-Weyl

Το θεώρημα αναπαράστασης Minkowski-Weyl (Minkowski-Weyl Representation Theorem), οφείλει το όνομά του στους δύο Γερμανούς μαθηματικούς Hermann Minkowski [**Εικόνα 2-6 (α)**] [55] και Hermann Klaus Hugo Weyl [**Εικόνα 2-6 (β)**] [56], που διατύπωσαν αυτό το θεώρημα, το 1896 και το 1935 αντίστοιχα [50].

(α) (β)

Εικόνα 2-6: (α) Hermann Minkowski (1864-1909) και (β) Hermann Klaus Hugo Weyl (1885-1955).

Το θεώρημα, διατυπώνεται όπως παρακάτω [49], [50], [57]:

Έστω ότι $P \subseteq \mathbb{R}^n$ είναι ένα πολύεδρο. Τότε υπάρχουν πεπερασμένα σύνολα $Q, R \subseteq \mathbb{R}^n$, όπου $Q = \{x_1, x_2, \dots, x_{|Q|}\}$ είναι το σύνολο των ακραίων σημείων του πολύεδρου P και $R = \{x_1, x_2, \dots, x_{|R|}\}$ είναι το σύνολο των ακραίων ακτίνων του πολύεδρου P, τέτοια ώστε κάθε σημείο $x \in P$ μπορεί να αναπαρασταθεί ως κυρτός συνδυασμός των ακραίων σημείων του και ως γραμμικός συνδυασμός των ακραίων ακτίνων του, δηλαδή:

$$x = \sum_{q \in Q} \lambda_q \, x_q + \sum_{r \in R} \mu_r x_r \qquad (2\text{-}4)$$

$$\sum_{q \in Q} \lambda_q = 1 \qquad (2\text{-}5)$$

$$\lambda_q \geq 0, q \in Q \qquad (2\text{-}6)$$

$$\mu_r \geq 0, r \in R \qquad (2\text{-}7)$$

<u>Σημειώσεις:</u>

- Αν το πολύεδρο P είναι φραγμένο, δεν υπάρχουν ακραίες ακτίνες.
- Αν το πολύεδρο P είναι μη φραγμένο, υπάρχουν ακραίες ακτίνες.

- Η απόδειξη του θεωρήματος αναπαράστασης Minkowski-Weyl παραλείπεται γιατί ξεφεύγει από το σκοπό της εργασίας, ωστόσο μία απόδειξή του αναφέρεται στην αναφορά [57].

2.4 Η έννοια του μειωμένου κόστους

2.4.1 Μειωμένο κόστος

Έστω το ακόλουθο πρόβλημα γραμμικού προγραμματισμού σε κανονική μορφή με:

Αντικειμενική συνάρτηση:

$$\min c^T x \tag{2-8}$$

Υποκειμενική στους περιορισμούς:

$$Ax = b \tag{2-9}$$

$$x \geq 0 \tag{2-10}$$

Όπου, A είναι ένας πίνακας διαστάσεων $m \times n$.

Αν ο συντελεστής των βασικών μεταβλητών (basic variables) συμβολιστεί με c_B^T και με π συμβολιστεί η βέλτιστη σκιώδης τιμή (optimal shadow price), τότε το μειωμένο κόστος (reduced cost) μιας μη βασικής μεταβλητής (nonbasic variables) είναι [49], [58]:

$$\bar{c} = c_B^T - \pi^T A \tag{2-11}$$

2.4.2 Ερμηνεία του μειωμένου κόστους

Στον γραμμικό προγραμματισμό, το μειωμένο κόστος είναι το ποσό κατά το οποίο θα πρέπει να βελτιωθεί ένας συντελεστής αντικειμενικής συνάρτησης προτού είναι δυνατόν μια αντίστοιχη μεταβλητή να λάβει θετική τιμή στη βέλτιστη λύση. Είναι το κόστος για την αύξηση μιας μεταβλητής κατά ένα μικρό ποσό, δηλαδή η πρώτη παράγωγος από ένα ορισμένο σημείο στο πολύεδρο που περιορίζει το πρόβλημα [49], [59].

Στην περίπτωση ενός προβλήματος ελαχιστοποίησης, «βελτιωμένο» σημαίνει «μειωμένο». Έτσι, στην περίπτωση ενός προβλήματος ελαχιστοποίησης κόστους, όπου οι

συντελεστές αντικειμενικής συνάρτησης αντιπροσωπεύουν το ανά μονάδα κόστος των δραστηριοτήτων που αντιπροσωπεύουν οι μεταβλητές, οι συντελεστές μειωμένου κόστους υποδεικνύουν πόσο θα πρέπει να μειωθεί κάθε συντελεστής κόστους πριν από την δραστηριότητα που αντιπροσωπεύεται από την αντίστοιχη μεταβλητή θα ήταν οικονομικά αποδοτική [59].

Στην περίπτωση ενός προβλήματος μεγιστοποίησης, «βελτιωμένο» σημαίνει «αυξημένο». Στην περίπτωση αυτή, όπου, για παράδειγμα, ο συντελεστής αντικειμενικής συνάρτησης μπορεί να αντιπροσωπεύει το καθαρό κέρδος ανά μονάδα της δραστηριότητας. Η μειωμένη τιμή κόστους δείχνει πόσο θα πρέπει να αυξηθεί η κερδοφορία της δραστηριότητας προκειμένου η δραστηριότητα να εμφανιστεί στη βέλτιστη λύση [59].

Κεφάλαιο 3. Η μέθοδος δημιουργίας στήλης και η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe

3.1 Εισαγωγή

Στο παρόν Κεφάλαιο, αναλύεται η κλασική μέθοδος δημιουργίας, μία αποδοτική μέθοδος για την επίλυση γραμμικών προβλημάτων μεγάλης κλίμακας. Στη συνέχεια περιγράφεται η αρχή της μεθόδου μαθηματικής αποσύνθεσης Dantzig-Wolfe αρχικά για προβλήματα γραμμικού προγραμματισμού και στη συνέχεια επεκτείνεται για προβλήματα γραμμικού ακέραιου και δυαδικού γραμμικού ακέραιου προγραμματισμού.

3.2 Η μέθοδος δημιουργίας στήλης

3.2.1 Ιστορικά στοιχεία-εφαρμογές της μεθόδου δημιουργίας στήλης

Η μέθοδος δημιουργίας στήλης (Column Generation, CG) προτάθηκε αρχικά από τους δύο Αμερικανούς μαθηματικούς Lester Randolph Ford Jr. και Delbert Ray Fulkerson, το 1958, για την επίλυση του προβλήματος μέγιστης ροής (maximum flow problem) σε δίκτυα (αλγόριθμος Ford-Fulkerson) [60]–[63]. Αργότερα, το 1960 οι George Dantzig και Philip Wolfe υιοθέτησαν την μέθοδο δημιουργίας στήλης και ανέπτυξαν την μέθοδο μαθηματικής αποσύνθεσης Dantzig-Wolfe [15]. Στο σημείο αυτό, αναφέρεται ότι η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe αποτελεί ισοδύναμη μορφή της μεθόδου δημιουργίας στήλης [29], [64].

Σε πολλές περιπτώσεις, η CG επιτρέπει την επίλυση μεγάλων γραμμικών προγραμμάτων που διαφορετικά θα ήταν δυσεπίλυτα [62]. Το κλασικό παράδειγμα ενός προβλήματος όπου χρησιμοποιείται με επιτυχία είναι το μονοδιάστατο πρόβλημα κοπής του αποθέματος (1D-CSP) που δημοσιεύτηκε το 1961 (μέρος I) [65] και το 1963 (μέρος II) [66] από τους Paul C. Gilmore και Ralph E. Gomory. Επιπλέον, η μέθοδος δημιουργίας στήλης έχει εφαρμοστεί σε πολλά προβλήματα. Το 1989 οι M. Desrochers και F. Soumis, προσέγγισαν το πρόβλημα σχεδιασμού πληρώματος αστικών συγκοινωνιών (Urban Transit Crew Scheduling Problem) με τη μέθοδο δημιουργίας στήλης [67]. Πολλά παραδείγματα αναφέρονται στο βιβλίο των G. Desaulniers, J. Desrosiers και M. Solomon [68].

3.2.2 Περιγραφή της κλασικής μεθόδου δημιουργίας στήλης

Πρόκειται για μια επαναληπτική διαδικασία που εφαρμόζεται στην επίλυση ενός προβλήματος γραμμικού προγραμματισμού με έναν πιθανόν τεράστιο αριθμό μεταβλητών, που ονομάζεται κύριο πρόβλημα (MP), έτσι ώστε οι στήλες στον πίνακα συντελεστών αυτού του προβλήματος μπορούν να δημιουργηθούν. Με την εκμετάλλευση αυτού του χαρακτηριστικού, η μέθοδος δημιουργίας στήλης ξεκινά με μια μειωμένη έκδοση του MP, που ονομάζεται περιορισμένο κύριο πρόβλημα (RMP), στο οποίο λαμβάνονται υπόψη μόνο λίγες στήλες του MP στην αρχή. Στη συνέχεια, με τις επαναλήψεις δημιουργούνται και προστίθενται νέες στήλες με αρνητικό μειωμένο κόστος στο RMP. Αυτές οι στήλες προκύπτουν από ένα ή περισσότερα υπο-προβλήματα κοστολόγησης. Γενικά η μέθοδος συγκλίνει σε μια βέλτιστη λύση δημιουργώντας ένα σχετικά μικρό υποσύνολο των στηλών. Πρέπει να σημειωθεί ότι κατά τον τερματισμό της επαναληπτικής διαδικασίας το RMP είναι ισοδύναμο με το κύριο πρόβλημα (MP), και επομένως, η βέλτιστη λύση του RMP είναι η βέλτιστη λύση του κύριου προβλήματος (MP) (**§ 3.2.4**) [62], [68]. Έστω το ακόλουθο γραμμικό πρόγραμμα με [69]:

Αντικειμενική συνάρτηση:

$$z_{MP} = \min \sum_{j \in J} c_j \lambda_j \tag{3-1}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{j \in J} a_j \lambda_j \geq b \tag{3-2}$$

$$\lambda_j \geq 0, j \in J \tag{3-3}$$

Όπου, $a_j, b \in \mathbb{R}^m, c_j, \lambda_j \in \mathbb{R}$ για $j \in J = \{1,2,\dots,n\}$ και n είναι ο αριθμός των μεταβλητών αυτού του γραμμικού προγράμματος. Αυτό το πρόβλημα αναφέρεται ως κύριο πρόβλημα (MP) με τιμή αντικειμενικής συνάρτησης z^*_{MP}. Επιπλέον, γίνεται η υπόθεση ότι ο αριθμός των μεταβλητών n είναι πολύ μεγαλύτερος από τον αριθμό των περιορισμών m **(3-2)**. Τέλος, a_j ονομάζεται συντελεστής στηλών ή απλούστερα στήλες του κύριου προβλήματος, όπου κάθε στήλη συσχετίζεται με μία από τις μεταβλητές του MP.

Σε πολλές πρακτικές καταστάσεις, το MP **(3-1)-(3-3)** μπορεί να περιγράφει ένα γραμμικό πρόγραμμα με ένα τεράστιο αριθμό μεταβλητών λ_j και απαριθμώντας αυτές τις

μεταβλητές ρητά, μπορεί να είναι υπολογιστικά δυσεπίλυτο. Σε οποιαδήποτε βασική εφικτή λύση στο πρόβλημα **(3-1)-(3-3)** οι περισσότερες από τις μεταβλητές θα είναι μη βασικές, καθώς $n = |J| \gg m$ και μπορούν να τεθούν ίσες με μηδέν. Αυτό στη θεωρία σημαίνει ότι για την επίλυση του προβλήματος **(3-1)-(3-3)** αρκεί μόνο να εξεταστεί ένα σχετικά μικρό υποσύνολο $J' \subset J$ μεταβλητών. Αντικαθιστώντας το σύνολο J από το σύνολο J' το MP **(3-1)-(3-3)** μετατρέπεται στο ακόλουθο περιορισμένο κύριο πρόβλημα (RMP).

Αντικειμενική συνάρτηση:

$$z_{RMP} = \min \sum_{j \in J'} c_j \lambda_j \qquad (3\text{-}4)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{j \in J'} a_j \lambda_j \geq b \qquad (3\text{-}5)$$

$$\lambda_j \geq 0, j \in J' \qquad (3\text{-}6)$$

Έστω ότι $\bar{z}_{RMP}$ είναι η βέλτιστη τιμή της αντικειμενικής συνάρτησης του RMP. Αν δοθεί οποιαδήποτε εφικτή λύση $\lambda_{RMP} \in \mathbb{R}^{|J'|}$ του RMP **(3-4)-(3-6)**, είναι δυνατό καθώς εξετάζεται ένα υποσύνολο μεταβλητών του MP να δημιουργηθεί μία λύση η οποία είναι εφικτή του MP. Επομένως, η βέλτιστη τιμή της αντικειμενικής συνάρτησης του RMP δίνει ένα άνω όριο στην τιμή της βέλτιστης αντικειμενικής συνάρτησης MP, δηλαδή: $z^*_{MP} \leq \bar{z}_{RMP}$. Επιπλέον, καθώς τα δύο προβλήματα περιέχουν τον ίδιο αριθμό περιορισμών, προκύπτει μία άλλη σημαντική σχέση μεταξύ του MP και του RMP: κάθε εφικτή δυϊκή λύση $\pi \in \mathbb{R}^m$ του RMP, είναι μία εφικτή δυϊκή λύση και για το MP. Επομένως, η βέλτιστη δυϊκή λύση του RMP αποδίδει πληροφορίες που μπορούν να χρησιμοποιηθούν και από το MP.

Η βασική ιδέα της μεθόδου δημιουργίας στήλης είναι η επαναληπτική δημιουργία στηλών και η πρόσθεσή τους στο RMP, μέχρι να προκύψει η βέλτιστη λύση $\bar{\lambda}_{RMP}$ που είναι και βέλτιστη λύση του MP. Στη μέθοδο simplex, κατά τη διαδικασία μετάβασης από ακραίο σημείο σε ακραίο σημείο της εφικτής περιοχής του προβλήματος, εξετάζονται επαναληπτικά βασικές εφικτές λύσεις-μη βασικές μεταβλητές με μη θετικά μειωμένα κόστη που εισέρχονται επαναληπτικά στη βάση, μέχρι να βρεθεί η βέλτιστη βάση. Στη μέθοδο δημιουργίας στήλης, αν δοθεί μία δυϊκή λύση $\pi \in \mathbb{R}^m$ του RMP, η ανάλογη διαδικασία για την εύρεση υποψήφιων μεταβλητών που δεν περιλαμβάνονται ακόμη στο υποσύνολο J', επιτυγχάνεται με επίλυση του υπο-προβλήματος:

$$\bar{c}^* = \min_{j \in J}\{c_j - \pi^T a_j\} \qquad (3\text{-}7)$$

Όπου, π είναι η αντίστοιχη βέλτιστη δυϊκή μεταβλητή που αντιστοιχεί στον περιορισμό **(3-5)**.

Αν $\bar{c}^* < 0$, προστίθεται στο RMP μία στήλη και (το RMP) επιλύεται ξανά μέχρι $\bar{c}^* \geq 0$ οπότε δεν μπορούν να προστεθούν άλλες στήλες και προκύπτει μία βέλτιστη λύση του RMP που είναι βέλτιστη και του MP. Τέλος, σημειώνεται ότι σε περίπτωση που το ελάχιστο μειωμένο κόστος δεν είναι «τόσο αρνητικό», για τη σύγκλιση του αλγορίθμου, μπορεί να εισαχθεί μία ανοχή: $c_j - \pi^T a_j \geq -\epsilon$ [35].

3.2.3 Ο αλγόριθμος της κλασικής μεθόδου δημιουργίας στήλης

Τα βήματα του κλασικού αλγορίθμου δημιουργίας στήλης (column generation algorithm) είναι τα παρακάτω [36], [62]:

Βήμα 1: Δημιουργία του περιορισμένου κύριου προβλήματος (RMP).

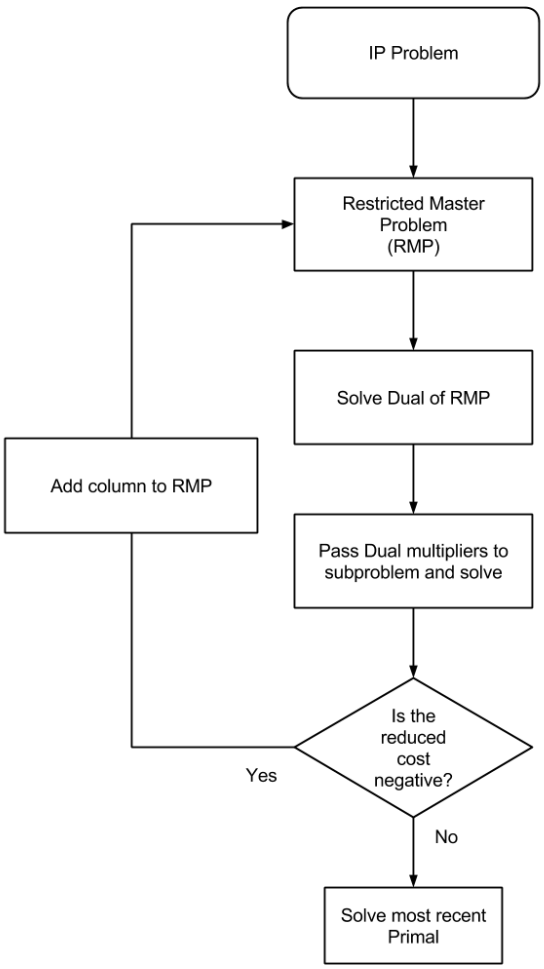
Βήμα 2: Αρχικοποίηση του περιορισμένου κύριου προβλήματος (RMP).

Βήμα 3: Επίλυση του RMP του **Βήματος 2**.

Βήμα 4: Δημιουργία του αντίστοιχου υπο-προβλήματος χρησιμοποιώντας τις δυϊκές μεταβλητές του **Βήματος 3**.

Βήμα 5: Επίλυση του υπο-προβλήματος, όπου η λύση συμβολίζεται με $\bar{c}^*$.

- Αν $\bar{c}^* \geq 0$, τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται.
- Αλλιώς αν $\bar{c}^* < 0$, τότε στο περιορισμένο κύριο πρόβλημα προστίθεται μία νέα στήλη και η νέα επανάληψη αρχίζει από το **Βήμα 2**.



Εικόνα 3-1: Διάγραμμα ροής του κλασικού αλγόριθμου δημιουργίας στήλης [70].

3.2.4 Όρια στην μέθοδο δημιουργίας στήλης

Έστω $\bar{z}_{RMP}$ η βέλτιστη τιμή της αντικειμενικής συνάρτησης για το RMP, z^*_{MP} η βέλτιστη τιμή της αντικειμενικής συνάρτησης του MP και $\bar{c}^*$ το μικρότερο μειωμένο κόστος. Όταν είναι γνωστό ένα πάνω όριο $\beta \geq \sum_{j \in J} \lambda_j$ για το άθροισμα μεταβλητών λ_j των στηλών για τη βέλτιστη λύση του MP, υπάρχει επίσης και ένα κάτω όριο σε κάθε επανάληψη. Σημειώνεται ότι το πάνω όριο του περιορισμένου κύριου προβλήματος (UB) $\bar{z}_{RMP}$ δεν μπορεί να μειωθεί περισσότερο από β φορές του μικρότερου μειωμένου κόστους $\bar{c}^*$. Δηλαδή, σύμφωνα με την αναφορά [68]:

$$\bar{z}_{RMP} + \beta \bar{c}^* \leq z^*_{MP} \leq \bar{z}_{RMP} \qquad (3\text{-}8)$$

Στη βέλτιστη λύση του MP (3-1)-(3-3) είναι $\bar{c}^* = 0$ για τις βασικές μεταβλητές και τα όρια κλείνουν. Επομένως, στην περίπτωση αυτή το πάνω και το κάτω όριο ταυτίζονται και $\bar{z}_{RMP} = \bar{z}_{MP}$[68]. Όταν η αντικειμενική συνάρτηση (3-1) είναι ήδη ένα άθροισμα μεταβλητών δηλαδή $c = 1$, χρησιμοποιείται το z^*_{MP} αντί του β και λαμβάνεται το βελτιωμένο κάτω όριο (LB):

$$\frac{\bar{z}_{RMP}}{1 - \bar{c}^*} \leq z^*_{MP} \qquad (3\text{-}9)$$

Η απόσταση μεταξύ πάνω και κάτω ορίου (gap), δίνεται από την ακόλουθη σχέση [71], [72]:

$$gap = \frac{UB - LB}{1 + |UB|} \qquad (3\text{-}10)$$

3.3 Η αρχή της μαθηματικής αποσύνθεσης Dantzig-Wolfe

Ας θεωρηθεί το ακόλουθο (αρχικό) γραμμικό πρόγραμμα:

Αντικειμενική συνάρτηση:

$$z_{LP} = c^T x \qquad (3\text{-}11)$$

Υποκειμενική στους περιορισμούς:

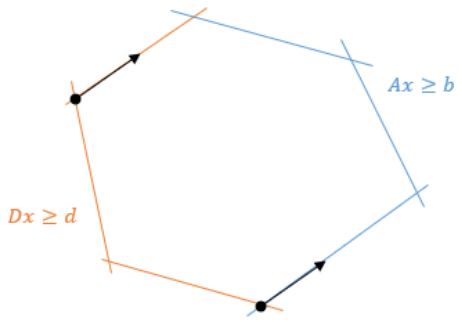
$$Ax \geq b \qquad (3\text{-}12)$$

$$Dx \geq d \qquad (3\text{-}13)$$

$$x \geq 0^n \qquad (3\text{-}14)$$

Όπου, $x, c \in \mathbb{R}^n, b \in \mathbb{R}^m, d \in \mathbb{R}^p, A \in \mathbb{R}^{m \times n}, D \in \mathbb{R}^{p \times n}$. Οι περιορισμοί μπορούν να διαχωριστούν σε δύο είδη (σύνολα): (α) σε ένα σύνολο περιορισμών P που είναι γνωστό πως να αντιμετωπιστούν (η επιλογή του $Dx \geq d$ θα πρέπει να περιγράφει μια δομή πάνω στην οποία είναι δυνατό να βελτιστοποιηθεί πιο εύκολα) και ονομάζονται «εύκολοι περιορισμοί» (easy constraints) (3-13), (3-14), όπου $P = \{x \in \mathbb{R}^n | Dx \geq d, x \geq 0^n\}$ και (β) οτιδήποτε

απομένει ορίζει ένα σύνολο περιορισμών, που ονομάζονται «περίπλοκοι περιορισμοί» (complicating constraints) **(3-12)** [36].



Εικόνα 3-2: Παράδειγμα απεικόνισης των «εύκολων» και των «δύσκολων» περιορισμών.

3.3.1 Διατύπωση του κύριου προβλήματος

Αν θεωρηθεί ότι το πολύεδρο P είναι μη φραγμένο (**Εικόνα 3-2**), τότε η ιδέα της μεθόδου αποσύνθεσης Dantzig-Wolfe, είναι η εφαρμογή του θεωρήματος αναπαράστασης Minkowski-Weyl στο σύνολο των «εύκολων» περιορισμών P. Σύμφωνα με την **§ 2.3** κάθε $x \in P$ μπορεί να αναπαρασταθεί ως γραμμικός συνδυασμός των ακραίων σημείων $Q = \{x_1, x_2, \dots, x_{|Q|}\}$ του P και των ακραίων ακτίνων $R = \{x_1, x_2, \dots, x_{|R|}\}$ του P. Δηλαδή,

$$x = \sum_{q \in Q} \lambda_q \, x_q + \sum_{r \in R} \mu_r x_r \tag{3-15}$$

$$\sum_{q \in Q} \lambda_q = 1 \tag{3-16}$$

$$\lambda_q \geq 0, q \in Q \tag{3-17}$$

$$\mu_r \geq 0, r \in R \tag{3-18}$$

Με αντικατάσταση των σχέσεων **(3-15)-(3-18)** στις σχέσεις **(3-11)** και **(3-12)**, προκύπτει η διατύπωση του κύριου προβλήματος:

Αντικειμενική συνάρτηση:

$$z_{MP} = \min c^T \left(\sum_{q \in Q} \lambda_q \, x_q + \sum_{r \in R} \mu_r x_r \right) \tag{3-19}$$

Υποκειμενική στους περιορισμούς:

$$A \left(\sum_{q \in Q} \lambda_q \, x_q + \sum_{r \in R} \mu_r x_r \right) \geq b \tag{3-20}$$

$$\sum_{q \in Q} \lambda_q = 1 \tag{3-21}$$

$$\lambda_q \geq 0, q \in Q \tag{3-22}$$

$$\mu_r \geq 0, r \in R \tag{3-23}$$

<u>Παρατηρήσεις:</u>

- Αν θεωρηθεί ότι το πολύεδρο P είναι φραγμένο (bounded), τότε δεν υπάρχουν ακραίες ακτίνες και λαμβάνονται υπόψη μόνο τα ακραία σημεία του convex hull.
- Το MP έχει ένα τεράστιο αριθμό μεταβλητών $|Q| + |R|$. Για το λόγο αυτό, μπορεί να επιλυθεί με τη μέθοδο δημιουργίας στήλης.

3.3.2 Επίλυση του κύριου προβλήματος με τη μέθοδο δημιουργίας στήλης

Το πρώτο βήμα της μεθόδου δημιουργίας στήλης είναι η διατύπωση του RMP. Όπως περιγράφεται στην **§ 3.2.2** αντί να θεωρηθούν ολόκληρα τα σύνολα των ακραίων σημείων Q και των ακραίων ακτίνων R, θεωρούνται τα υποσύνολά τους Q' και R' αντίστοιχα. Έτσι, το RMP διατυπώνεται όπως παρακάτω:

Αντικειμενική συνάρτηση:

$$z_{RMP} = \min c^T \left(\sum_{q \in Q'} \lambda_q \, x_q + \sum_{r \in R'} \mu_r x_r \right) \tag{3-24}$$

Υποκειμενική στους περιορισμούς:

$$A \left(\sum_{q \in Q'} \lambda_q \, x_q + \sum_{r \in R'} \mu_r x_r \right) \geq b \tag{3-25}$$

$$\sum_{q \in Q'} \lambda_q = 1 \tag{3-26}$$

$$\lambda_q \geq 0, q \in Q' \tag{3-27}$$

$$\mu_r \geq 0, r \in R' \tag{3-28}$$

3.3.3 Διατύπωση του υπο-προβλήματος

Η αντικειμενική συνάρτηση του υπο-προβλήματος σε κάθε επανάληψη, είναι το ελάχιστο μειωμένο κόστος ενός ακραίου σημείου $q \in Q$ ή μιας ακραίας ακτίνας $r \in R$.

3.3.3.1 Υπολογισμός του μειωμένου κόστους

Έστω ότι π είναι η δυϊκή μεταβλητή που αντιστοιχεί στον περιορισμό **(3-25)** και σ η δυϊκή μεταβλητή που αντιστοιχεί στον περιορισμό **(3-26)**, τότε το υπο-πρόβλημα διατυπώνεται ως εξής:

Αντικειμενική συνάρτηση:

$$z_{sp} = \min\{(c^T - \pi^T A)x - \sigma\} \tag{3-29}$$

Υποκειμενική στους περιορισμούς:

$$Dx \geq d \tag{3-30}$$

$$x \geq 0^n \tag{3-31}$$

3.3.4 Επίλυση του υπο-προβλήματος

Το υπο-πρόβλημα **(3-29)-(3-31)** είναι ένα γραμμικό πρόβλημα και έτσι μπορεί να επιλυθεί με κάποια μέθοδο simplex (π.χ. αναθεωρημένη μέθοδος simplex) [36], [49], [73].

Υπάρχουν τρεις περιπτώσεις για τη λύση του υπο-προβλήματος:

- Αν $z_{sp} = -\infty$, προκύπτει μία ακραία ακτίνα $r \in R$, με κόστος $\bar{c}_r < 0$ και επομένως, η μεταβλητή μ_r προστίθεται στο RMP με κόστος $c_r^T x_r$ (στην αντικειμενική συνάρτηση) και συντελεστές στήλης $\begin{bmatrix} Ax_r \\ 0 \end{bmatrix}$ (στον πίνακα περιορισμών). Δηλαδή, $\begin{bmatrix} c_r^T x_r \\ Ax_r \\ 0 \end{bmatrix}$.
- Αν $-\infty < z_{sp} < 0$, προκύπτει ένα ακραίο σημείο $q \in Q$, με κόστος $\bar{c}_q < 0$ και επομένως, η μεταβλητή λ_q προστίθεται στο RMP με κόστος $c_q^T x_q$ (στην αντικειμενική συνάρτηση) και συντελεστές στήλης $\begin{bmatrix} Ax_q \\ 1 \end{bmatrix}$ (στον πίνακα περιορισμών).Δηλαδή, $\begin{bmatrix} c_q^T x_q \\ Ax_q \\ 1 \end{bmatrix}$.
- Αν $z_{sp} \geq 0$, δεν υπάρχει $j \in Q \cup R$ με $\bar{c}_j < 0$ και έχει βρεθεί βέλτιστη λύση για το MP.

Από το γενικό κάτω όριο της τιμής της βέλτιστης αντικειμενικής συνάρτησης MP, που δίνεται από τη σχέση **(3-8)**, μπορεί να κατασκευαστεί ένα κατώτερο όριο της βέλτιστης τιμής του MP **(3-19)-(3-23)**, z_{MP}^*. Πρέπει να σημειωθεί όμως ότι λόγω των περιορισμών κυρτότητας **(3-21)**, ο συντελεστής β της σχέσης **(3-8)**, μπορεί να αντικατασταθεί με 1, αποδίδοντας τα παρακάτω όρια:

$$\bar{z}_{RMP} + z_{sp} \leq z_{MP}^* \leq \bar{z}_{RMP} \tag{3-32}$$

<u>Παρατήρηση:</u>

Η σχέση **(3-29)** μπορεί να γραφεί όπως παρακάτω:

$$z_{sp} = \min\{(c^T - \pi^T A)x\} \tag{3-33}$$

Στην περίπτωση αυτή οι περιορισμοί (3-30) και (3-31) δεν αλλάζουν. Όμως, αλλάζει ο έλεγχος για τη λύση του υπο-προβλήματος:

- Αν $z_{sp} = -\infty$, προκύπτει μία ακραία ακτίνα $r \in R$, με κόστος $\bar{c}_r < 0$ και επομένως, η μεταβλητή μ_r προστίθεται στο RMP με κόστος $c_r^T x_r$ και συντελεστές στήλης $\begin{bmatrix} Ax_r \\ 0 \end{bmatrix}$. Δηλαδή, $\begin{bmatrix} c_r^T x_r \\ Ax_r \\ 0 \end{bmatrix}$.
- Αν $-\infty < z_{sp} - \sigma < 0$, προκύπτει ένα ακραίο σημείο $q \in Q$, με κόστος $\bar{c}_q < 0$ και επομένως, η μεταβλητή λ_q προστίθεται στο RMP με κόστος $c_q^T x_q$ και συντελεστές στήλης $\begin{bmatrix} Ax_q \\ 1 \end{bmatrix}$. Δηλαδή, $\begin{bmatrix} c_q^T x_q \\ Ax_q \\ 1 \end{bmatrix}$.
- Αν $z_{sp} - \sigma \geq 0$, δεν υπάρχει $j \in Q \cup R$ με $\bar{c}_j < 0$ και έχει βρεθεί βέλτιστη λύση για το MP.

3.3.5 Ο αλγόριθμος μαθηματικής αποσύνθεσης Dantzig-Wolfe

Συνοπτικά, τα βήματα του αλγορίθμου Dantzig-Wolfe [49], [74] φαίνονται παρακάτω:

Βήμα 1: Διαχωρισμός των περιορισμών του αρχικού προβλήματος σε δύο σύνολα. Εφαρμογή του θεωρήματος αναπαράστασης Minkowski-Weyl στο convex hull του συνόλου των «εύκολων» περιορισμών $P = \{x \in \mathbb{R}^n | Dx \geq d, x \geq 0^n\}$ για τη διατύπωση του κύριου προβλήματος και επίλυση αυτού με τη μέθοδο δημιουργίας στήλης.

Βήμα 2: Δημιουργία του περιορισμένου κύριου προβλήματος (RMP) και δημιουργία ενός αρχικού συνόλου στηλών για το RMP (αρχικοποίηση), ξεκινώντας από τυχαία ακραία σημεία.

Βήμα 3: Επίλυση του RMP του **Βήματος 1**.

Βήμα 4: Χρησιμοποιώντας τις δυϊκές μεταβλητές του **Βήματος 2**, δημιουργία του αντίστοιχου υπο-προβλήματος.

Βήμα 5: Επίλυση του υπο-προβλήματος, υπολογισμός του $\bar{c}^*$.

- Αν $\bar{c}^* \geq 0$, τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται.
- Αλλιώς αν $\bar{c}^* < 0$, τότε στο περιορισμένο κύριο πρόβλημα προστίθεται η στήλη που προέκυψε από τη βέλτιστη λύση του υπο-προβλήματος και μετάβαση στο **Βήμα 2**.

Στην επόμενη ενότητα, η μαθηματική αποσύνθεση DW, διατυπώνεται στην περίπτωση που ο πίνακας περιορισμών D του αρχικού προβλήματος **(3-11)-(3-14)** έχει μία ειδική δομή.

3.3.6 Η γωνιακή δομή του πίνακα περιορισμών

Η κλασική περίπτωση της μαθηματικής αποσύνθεσης Dantzig-Wolfe, εφαρμόζεται όταν η δομή του πίνακα D είναι γωνιακή. Για το λόγο αυτό, ας θεωρηθεί η περίπτωση όπου ο πίνακας περιορισμών D του αρχικού προβλήματος **(3-11)-(3-14)** έχει γωνιακή δομή (block-diagonal structure) [73]. Η ειδική δομή του πίνακα D, τα σταθερά διανύσματα d, c και οι μεταβλητές απόφασης x σε αποσυντιθέμενη μορφή που συνδέεται με την ειδική μορφή του πίνακα D αντίστοιχα, γράφονται όπως παρακάτω:

$$D = \begin{bmatrix} D^1 & 0 & 0 & \cdots & 0 \\ 0 & D^2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & D^h \end{bmatrix}, d = \begin{bmatrix} d^1 \\ d^2 \\ \vdots \\ d^h \end{bmatrix}, c = \begin{bmatrix} c^1 \\ c^2 \\ \vdots \\ c^h \end{bmatrix}, x = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^h \end{bmatrix}$$

Επίσης, ο πίνακας γράφεται ανάλογα ως εξής: $A = [A^1 \quad A^2 \quad \cdots \quad A^h]$, όπου $A^k \in \mathbb{R}^{m \times |x^k|}, k = 1,2, \dots, h$. Από τα παραπάνω, το αρχικό πρόβλημα **(3-11)-(3-14)** έχει την ακόλουθη μορφή:

Αντικειμενική συνάρτηση:

$$z_{LP} = (c^1)^T x^1 + (c^2)^T x^2 + \cdots + (c^h)^T x^h \qquad (3\text{-}34)$$

Υποκειμενική στους περιορισμούς:

$$A^1 x^1 + A^2 x^2 + \cdots + A^h x^h \geq b \qquad (3\text{-}35)$$

$$D^1 x^1 \qquad \geq d^1 \qquad (3\text{-}36)$$

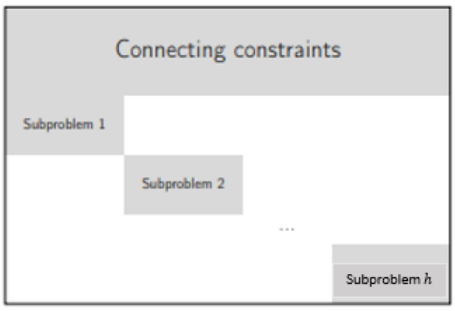
$$D^2 x^2 \qquad \geq d^2 \qquad (3\text{-}37)$$

$$\ddots \qquad \vdots \qquad (3\text{-}38)$$

$$D^h x^h \quad \geq d^h \tag{3-39}$$

$$[(x^1)^T \ \ (x^2)^T \quad \dots \quad (x^h)^T \ ]^T \geq 0^n \tag{3-40}$$

Πρέπει να σημειωθεί ότι κάθε πρόβλημα που έχει τη δομή των περιορισμών του προβλήματος **(3-34)-(3-40)** λέγεται ότι έχει γωνιακή δομή. Οι περιορισμοί **(3-35)**, λέγονται περιορισμοί σύνδεσης και καθένας από τους υπόλοιπους περιορισμούς **(3-36)-(3-40)** αποτελεί ένα υπο-πρόβλημα (μπλοκ), $k = 1,2, \dots, h$.



Εικόνα 3-3: Γωνιακή δομή όπου Connecting constraints είναι το μπλοκ των περιορισμών σύνδεσης και καθένα από τα μπλοκ Subproblem $1,2 \dots h$*, ορίζει ένα διαφορετικό υπο-πρόβλημα* [75].

Όπως μπορεί να παρατηρηθεί, μόνο οι περιορισμοί **(3-35)** περιέχουν τα στοιχεία του πίνακα A που συνδέουν τις μεταβλητές από διαφορετικά υποσύνολα $k = 1,2, \dots, h$ ενώ καθένας από τους περιορισμούς **(3-36)-(3-39)** μπορεί να αναπαρασταθεί σαν ένα ανεξάρτητο πολύτοπο $X^k = \left\{x^k \in \mathbb{R}^{|x^k|} \mid D^k x^k \geq d^k, x^k \geq 0^{|x^k|}\right\}, k = 1,2, \dots, h$.

Σύμφωνα με το θεώρημα Minkowski-Weyl **(§ 2.3)** κάθε $x^k \in X^k, k = 1,2, \dots, h$ μπορεί να αναπαρασταθεί σαν κυρτός συνδυασμός των ακραίων σημείων $Q^k = \{x_1, x_2, \dots, x_{|Q|}\}$ του X^k και σαν γραμμικός συνδυασμός των ακραίων ακτίνων $R^k = \{x_1, x_2, \dots, x_{|R|}\}$ του X^k ως εξής:

$$x^k = \sum_{q \in Q^k} \lambda_q^k \, x_q^k + \sum_{r \in R^k} \mu_r^k x_r^k \tag{3-41}$$

$$\sum_{q \in Q^k} \lambda_q^k = 1 \tag{3-42}$$

$$\lambda_q^k \geq 0^{|Q^k|}, q \in Q^k \tag{3-43}$$

$$\mu_r^k \geq 0^{|R^k|}, r \in R^k \tag{3-44}$$

Αν οι σχέσεις (3-41)-(3-44) αντικατασταθούν στο πρόβλημα (3-34)-(3-40), προκύπτει το ακόλουθο MP:

Αντικειμενική συνάρτηση:

$$z_{MP} = \min \sum_{k \in K} \sum_{q \in Q^k} (c^k)^T \lambda_q^k x_q^k + \sum_{k \in K} \sum_{r \in R^k} (c^k)^T \mu_r^k x_r^k \tag{3-45}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k \in K} \sum_{q \in Q^k} A^k \lambda_q^k x_q^k + \sum_{k \in K} \sum_{r \in R^k} A^k \mu_r^k x_r^k \geq b \tag{3-46}$$

$$\sum_{q \in Q^k} \lambda_q^k = 1, k \in \{1,2, \dots, h\} \tag{3-47}$$

$$\lambda_q^k \geq 0, q \in Q \tag{3-48}$$

$$\mu_r^k \geq 0, r \in R \tag{3-49}$$

Όμοια, το MP (3-45)-(3-49) επιλύεται με τη μέθοδο δημιουργίας στήλης και επομένως, πρέπει να διατυπωθεί το RMP. Για κάθε υποσύστημα (subsystem) $k \in K$, θεωρούνται τα υποσύνολα $\bar{Q}^k \subseteq Q^k$ και $\bar{R}^k \subseteq R^k$ και αντικαθίστανται στο MP (3-45)-(3-49). Επομένως, το RMP διατυπώνεται όπως παρακάτω:

Αντικειμενική συνάρτηση:

$$z_{RMP} = \min \sum_{k \in K} \sum_{q \in \bar{Q}^k} (c^k)^T \lambda_q^k x_q^k + \sum_{k \in K} \sum_{r \in \bar{R}^k} (c^k)^T \mu_r^k x_r^k \tag{3-50}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k\in K}\sum_{q\in\bar{Q}^k} A^k\lambda_q^k x_q^k + \sum_{k\in K}\sum_{r\in\bar{R}^k} A^k\mu_r^k x_r^k \geq b \qquad (3\text{-}51)$$

$$\sum_{q\in\bar{Q}^k} \lambda_q^k = 1, k \in \{1,2,\dots,h\} \qquad (3\text{-}52)$$

$$\lambda_q^k \geq 0, q \in \bar{Q}^k \qquad (3\text{-}53)$$

$$\mu_r^k \geq 0, r \in \bar{R}^k \qquad (3\text{-}54)$$

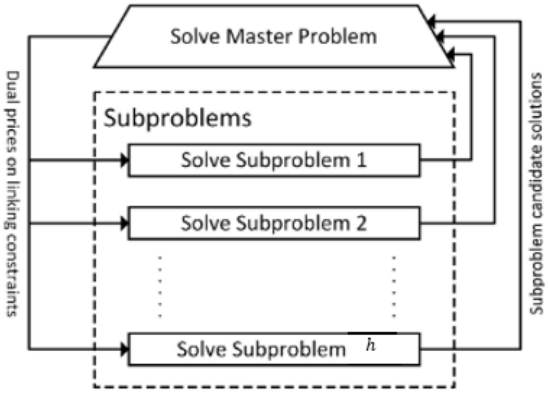
Αν $(\pi, \sigma^1, \sigma^2, \dots, \sigma^h)$ είναι μία βέλτιστη δυϊκή λύση του RMP, όπου π αντιστοιχεί στους περιορισμούς σύνδεσης **(3-51)** και $\sigma^k, k \in \{1,2,\dots,h\}$ αντιστοιχούν στους h περιορισμούς κυρτότητας **(3-52)**, τότε λύνοντας h διαφορετικά υπο-προβλήματα, δημιουργούνται στήλες που προστίθενται στο RMP. Τα h υπο-προβλήματα διατυπώνονται όπως παρακάτω:

$$\bar{c}^{k*} = \min_{x^k\in X^k}\{((c^k)^T - \pi^T A)x^k - \sigma^k\}, k \in \{1,2,\dots,h\} \qquad (3\text{-}55)$$

Η διαδικασία της μεθόδου δημιουργίας στήλης για την επίλυση των h υπο-προβλημάτων επαναλαμβάνεται μέχρι $\bar{c}^{k*} \geq 0, k \in K$, όπου και τερματίζεται.

Από τα όρια για την τιμή της βέλτιστης αντικειμενικής συνάρτησης του MP, z_{MP}^*, στην περίπτωση ενός υπο-προβλήματος, που δίνονται από τη σχέση **(3-32)**, μπορεί να εξαχθεί συμπέρασμα για την περίπτωση πολλών υπο-προβλημάτων. Αν $\bar{z}_{RMP}$ είναι η βέλτιστη τιμή της αντικειμενικής συνάρτησης του RMP, ισχύει η ακόλουθη σχέση [73]:

$$\bar{z}_{RMP} + \sum_{k\in K}\bar{c}^{k*} \leq z_{MP}^* \leq \bar{z}_{RMP} \qquad (3\text{-}56)$$



Εικόνα 3-4: Λειτουργία του αλγόριθμου μαθηματικής αποσύνθεσης Dantzig-Wolfe. Σε κάθε επανάληψη, οι δυϊκές τιμές που σχετίζονται με τους περιορισμούς σύνδεσης προκύπτουν από την επίλυση του RMP. Αυτές οι τιμές χρησιμοποιούνται από τα h υπο-προβλήματα για τον υπολογισμό μιας ενημερωμένης λύσης που βελτιώνει τη συνολική αντικειμενική συνάρτηση [76].

<u>Σημειώσεις</u>

- Στην αναφορά [77] παρουσιάζεται ένα παράδειγμα επίλυσης ενός προβλήματος γραμμικού προγραμματισμού, χρησιμοποιώντας την μέθοδο μαθηματικής αποσύνθεσης Dantzig-Wolfe.
- Στην επόμενη ενότητα εξετάζεται η χρήση της μαθηματικής αποσύνθεσης στον ακέραιο γραμμικό προγραμματισμό.

3.4 Διατύπωση της αποσύνθεσης Dantzig-Wolfe για προβλήματα ακέραιου γραμμικού προγραμματισμού

3.4.1 Εισαγωγή-ορισμός αρχικού προβλήματος-ορισμός πολυέδρων

Στην ενότητα αυτή, η αρχή της αποσύνθεσης Dantzig-Wolfe για προβλήματα γραμμικού προγραμματισμού, επεκτείνεται σε προβλήματα ακέραιου γραμμικού προγραμματισμού (Integer Linear Programming, ILP).

Ας θεωρηθεί το ακόλουθο γραμμικό ακέραιο πρόβλημα (ILP) ελαχιστοποίησης:

Αντικειμενική συνάρτηση:

$$z_{ILP} = \min\ c^T x \tag{3-57}$$

Υποκειμενική στους περιορισμούς:

$$Ax \geq b \tag{3-58}$$

$$Dx \geq d \tag{3-59}$$

$$x \in \mathbb{Z}_+^n \tag{3-60}$$

Όπου, τα διανύσματα και οι πίνακες ορίζονται όπως ακριβώς στην **§ 3.3.1**.

Ας θεωρηθεί ότι το πρόβλημα θα μπορούσε να επιλυθεί πιο εύκολα αν γίνει εκμετάλλευση της δομής των περιορισμών **(3-59)** («εύκολοι» περιορισμοί) και ορίζουν λόγω των περιορισμών **(3-60)** το σύνολο $X = \{x \in \mathbb{R}_+^n \,|Dx \geq d\,\}$. Οι περιορισμοί που απομένουν, ορίζουν το σύνολο των «πολύπλοκων περιορισμών». Επιπλέον, ας γίνει για λόγους απλότητας, η υπόθεση ότι το σύνολο X είναι φραγμένο, οπότε δεν υπάρχουν ακραίες ακτίνες [73].

Σύμφωνα με τους Vanderbeck F. & Wolsey L. [25] και τους Desaulniers G., Desrosiers J., Solomon M. [68], για την αποσύνθεση ακέραιων γραμμικών προγραμμάτων, μπορούν να χρησιμοποιηθούν δύο ισοδύναμες προσεγγίσεις. Αυτές είναι: 1) η προσέγγιση κυρτότητας (convexification approach), ***Θεώρημα 3.1*** και 2) η προσέγγιση της διακριτοποίησης (discretization approach), ***Θεώρημα 3.2***.

3.4.2 Το θεώρημα αναπαράστασης Minkowski-Weyl για προβλήματα ακέραιου γραμμικού προγραμματισμού

Έστω $P = \{x \in \mathbb{R}_+^n \,|Dx \geq d, x \geq 0^n\} \neq \emptyset$ και $X = P \cap \mathbb{Z}_+^n \neq \emptyset$. Τότε υπάρχει ένα πεπερασμένο σύνολο ακραίων σημείων $Q \subseteq X$ και ένα πεπερασμένο σύνολο ακραίων ακτίνων R του συνόλου P τέτοια ώστε [73]:

$$X = \left\{ x \in \mathbb{R}_+^n \mid x = \sum_{q \in Q} x_p \lambda_p + \sum_{r \in R} x_r \lambda_r \,, \sum_{q \in Q} \lambda_p = 1, \lambda \in \mathbb{Z}_+^{|Q|}, \mu \in \mathbb{Z}_+^{|R|} \right\} \quad (3\text{-}61)$$

Αφού έγινε η υπόθεση ότι το σύνολο X είναι φραγμένο, δεν υπάρχουν ακραίες ακτίνες, επομένως:

$$X = \left\{ x \in \mathbb{R}_+^n \mid x = \sum_{q \in Q} x_p \lambda_p \,, \sum_{q \in Q} \lambda_p = 1, \lambda \in \mathbb{Z}_+^{|Q|} \right\} \quad (3\text{-}62)$$

3.4.3 Διατύπωση του κύριου προβλήματος

Όπως σημειώθηκε, για λόγους απλότητας, το σύνολο X θεωρείται φραγμένο, ως αποτέλεσμα δεν υπάρχουν ακραίες ακτίνες [26], [27], [80]. Για τη διατύπωση του MP, **μόνο** στο σημείο αυτό, θα χρησιμοποιηθούν η προσέγγιση κυρτότητας και η προσέγγιση διακριτοποίησης.

3.4.3.1 Θεώρημα 3.1 (Διατύπωση του MP με την προσέγγιση κυρτότητας)

Αντικειμενική συνάρτηση [25]:

$$z_{MP}^{ILP} = \min \sum_{q \in Q} c^T x_q \lambda_q \quad (3\text{-}63)$$

Υποκειμενική στους περιορισμούς:

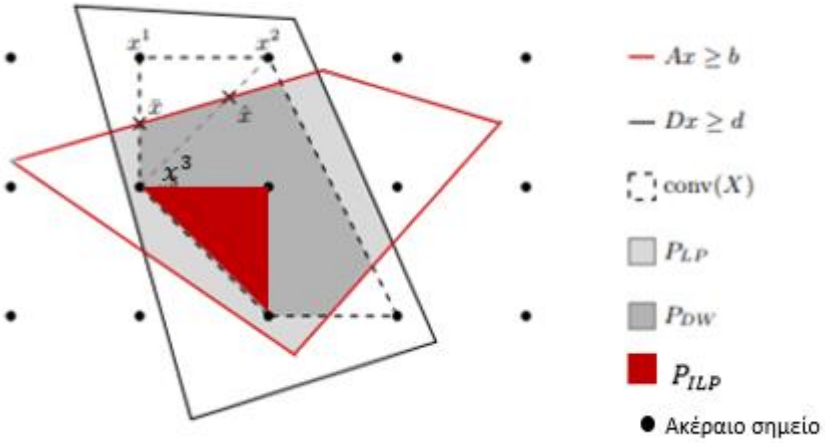
$$\sum_{q \in Q} A x_q \lambda_q \geq b \quad (3\text{-}64)$$

$$\sum_{q \in Q} \lambda_q = 1 \quad (3\text{-}65)$$

$$x = \sum_{q \in Q} x_p \lambda_p \quad (3\text{-}66)$$

$$\lambda_q \geq 0, q \in Q \quad (3\text{-}67)$$

Όπου, x_p είναι τα ακραία σημεία του $conv\ (X)$. Επίσης, η σχέση **(3-66)**, μπορεί να αφαιρεθεί .



Εικόνα 3-5: $P_{LP} = \{x: Ax \geq b, Dx \geq b\}$ είναι η εφικτή περιοχή της γραμμικής χαλάρωσης του ILP ***(3-57)-(3-60)****. $P_{DW} = \{x: Ax \geq b, x \in conv(X)\}$ είναι το πολύεδρο της μαθηματικής αποσύνθεσης Dantzig-Wolfe στο ILP. Στην μαθηματική αποσύνθεση Dantzig-Wolfe επαναδιατυπώνεται το $conv\ (X)$ όλων των ακέραιων λύσεων που είναι εφικτές για το υποσύστημα των περιορισμών $Dx \geq b$, χρησιμοποιώντας τα ακραία σημεία του $x^1, x^2, x^3, \ldots$. Η τομή του $conv\ (X)$ με τους περιορισμούς που απομένουν, δίνουν το P_{DW} που είναι πιο σφικτή προσέγγιση απ' ότι η γραμμική χαλάρωση.*

3.4.3.2 Θεώρημα 3.2 (Διατύπωση του MP με την προσέγγιση διακριτοποίησης)

Με αντικατάσταση του x στο αρχικό πρόβλημα **(3-57)-(3-61)**, το κύριο πρόβλημα διατυπώνεται όπως παρακάτω [25]:

Αντικειμενική συνάρτηση:

$$z_{MP}^{ILP} = \min \sum_{q \in Q} c^T x_q \lambda_q \qquad (3\text{-}68)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{q \in Q} A x_q \lambda_q \geq b \qquad (3\text{-}69)$$

$$\sum_{q \in Q} \lambda_q = 1 \qquad (3\text{-}70)$$

$$\lambda_q \in \{0,1\}, q \in Q \qquad (3\text{-}71)$$

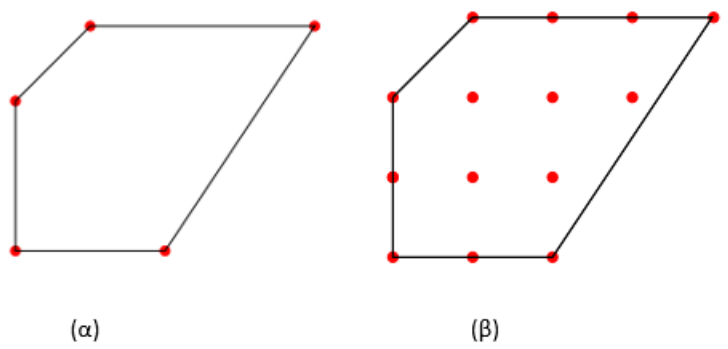
Όπου, x_q είναι όλα τα ακέραια σημεία του X.

3.4.3.3 *Διαφορές προσέγγισης κυρτότητας- προσέγγισης διακριτοποίησης*

- Από τη μία πλευρά στην προσέγγιση κυρτότητας για να εκφραστεί η ακεραιότητα, πρέπει να γίνει επιστροφή στις αρχικές μεταβλητές x . **(3-66)** και όχι στις μεταβλητές λ του MP. Από την άλλη πλευρά, τα αποτελέσματα της προσέγγισης διακριτοποίησης είναι ακέραιες τιμές.
- Από τη μία πλευρά, η προσέγγιση κυρτότητας χρησιμοποιεί μόνο τα ακέραια ακραία σημεία του $conv(X)$. Από την άλλη πλευρά, η προσέγγιση της διακριτοποίησης είναι μια εναλλακτική αναδιατύπωση Dantzig-Wolfe. Η προσέγγιση αυτή χρησιμοποιεί και τα εσωτερικά σημεία του ακέραιου πολύεδρου X. Αυτό επιτρέπει την επιβολή των περιορισμών ακεραιότητας απευθείας στις νέες μεταβλητές.

3.4.3.4 *Ομοιότητες προσέγγισης κυρτότητας- προσέγγισης διακριτοποίησης*

- Και οι δύο δίνουν την ίδια γραμμική χαλάρωση του MP.
- Για την ειδική περίπτωση όπου οι μεταβλητές είναι δυαδικές ακέραιες, δεν υπάρχει διαφορά μεταξύ της προσέγγισης κυρτότητας και της προσέγγισης διακριτοποίησης όταν $X \subset \{0,1\}^n$, όπως κάθε σημείο $x \in X$ είναι ένα ακραίο σημείο του $conv(X)$. Με άλλα λόγια: $x = \sum_{q \in Q} x_p \lambda_p \{0,1\}^n$ στην προσέγγιση κυρτότητας αν και μόνο αν $\lambda \in \{0,1\}$ στην προσέγγιση διακριτοποίησης.



Εικόνα 3-6: Παραδείγματα (α) προσέγγισης κυρτότητας σε ένα φραγμένο IP και (β) η προσέγγισης διακριτοποίησης της μαθηματικής αποσύνθεσης Dantzig-Wolfe σε ένα φραγμένο IP [79]. *Τα κόκκινου χώματος σημεία είναι ακέραια σημεία.*

Σημειώσεις:

- Τα ακραία σημεία του $conv(X)$ είναι γενικά ένα αυστηρό υποσύνολο των σημείων του X.
- Διευκρινίζεται ότι η προσέγγιση διακριτοποίησης, θα χρησιμοποιηθεί **μόνο** στη διατύπωση του MP και αυτό, για να συγκριθεί με την προσέγγιση κυρτότητας. Στο υπόλοιπο μέρος, όλες οι διατυπώσεις, γίνονται με την χρήση της προσέγγισης κυρτότητας.

3.4.4 Εφαρμογή της μεθόδου δημιουργίας στήλης

3.4.4.1 Διατύπωση του περιορισμένου κύριου προβλήματος (RMP)

Όπως σημειώθηκε, χρησιμοποιείται η προσέγγιση κυρτότητας. Για ένα υποσύνολο Q' του Q, ορίζεται το παρακάτω RMP:

Αντικειμενική συνάρτηση:

$$z_{RMP}^{ILP} = \min \sum_{q \in Q'} c^T x_q \lambda_q \qquad (3\text{-}72)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{q \in Q'} A x_q \lambda_q \geq b \qquad (3\text{-}73)$$

$$\sum_{q \in Q'} \lambda_q = 1 \qquad (3\text{-}74)$$

$$\lambda_q \geq 0, q \in Q' \qquad (3\text{-}75)$$

3.4.4.2 Διατύπωση του υπο-προβλήματος

Αν π και σ είναι οι δυϊκές μεταβλητές που αντιστοιχούν στους περιορισμούς **(3-73)** και **(3-74)** αντίστοιχα τότε το υπο-πρόβλημα διατυπώνεται όπως παρακάτω:

Αντικειμενική συνάρτηση:

$$z_{sp} = \min \{(c^T - \pi^T A)x - \sigma\} \qquad (3\text{-}76)$$

Υποκειμενική στους περιορισμούς:

$$Dx \geq d \tag{3-77}$$

$$x \geq 0^n \tag{3-78}$$

Επειδή το πολύεδρο X είναι φραγμένο (δεν υπάρχουν ακραίες ακτίνες), υπάρχουν δύο περιπτώσεις για τη λύση του υπο-προβλήματος. Αυτές είναι οι ακόλουθες:

- Αν $z_{sp} < 0$, προκύπτει ένα ακραίο σημείο $q \in Q$, με κόστος $\bar{c}_q < 0$ και επομένως, η μεταβλητή λ_q προστίθεται στο RMP με κόστος $c_q^T x_q$ και συντελεστές στήλης $\begin{bmatrix} Ax_q \\ 1 \end{bmatrix}$. Δηλαδή, $\begin{bmatrix} c_q^T x_q \\ Ax_q \\ 1 \end{bmatrix}$.
- Αν $z_{sp} \geq 0$, έχει βρεθεί βέλτιστη λύση για το MP.

Αν η δυϊκή μεταβλητή σ δεν τοποθετούνταν στην **(3-76)**, τότε ο έλεγχος για τη λύση του υπο-προβλήματος είναι:

- Αν $z_{sp} - \sigma < 0$, προκύπτει ένα ακραίο σημείο $q \in Q$, με κόστος $\bar{c}_q < 0$ και επομένως, η μεταβλητή λ_q προστίθεται στο RMP με κόστος $c_q^T x_q$ και συντελεστές στήλης $\begin{bmatrix} Ax_q \\ 1 \end{bmatrix}$. Δηλαδή, $\begin{bmatrix} c_q^T x_q \\ Ax_q \\ 1 \end{bmatrix}$.
- Αν $z_{sp} - \sigma \geq 0$, έχει βρεθεί βέλτιστη λύση για το MP.

3.4.4.3 Η γωνιακή δομή σε προβλήματα ακέραιου γραμμικού προγραμματισμού

Σε αντιστοιχία με την **§ 3.3.6**, μπορεί να οριστεί και η γωνιακή δομή προβλημάτων ακέραιου γραμμικού προγραμματισμού. Ας θεωρηθεί η περίπτωση όπου ο πίνακας περιορισμών D του αρχικού προβλήματος **(3-57)-(3-60)** έχει γωνιακή δομή. Η ειδική δομή του πίνακα D, τα σταθερά διανύσματα d, c και οι μεταβλητές απόφασης x σε αποσυντιθέμενη μορφή που συνδέεται με την ειδική μορφή του πίνακα D αντίστοιχα, γράφονται όπως παρακάτω:

$$D = \begin{bmatrix} D^1 & 0 & 0 & \cdots & 0 \\ 0 & D^2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & D^h \end{bmatrix}, d = \begin{bmatrix} d^1 \\ d^2 \\ \vdots \\ d^h \end{bmatrix}, c = \begin{bmatrix} c^1 \\ c^2 \\ \vdots \\ c^h \end{bmatrix}, x = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^h \end{bmatrix}$$

Επίσης, ο πίνακας γράφεται ανάλογα ως εξής: $A = [A^1 \quad A^2 \quad \cdots \quad A^h]$, όπου $A^k \in \mathbb{Z}^{m \times |x^k|}, k = 1,2, \dots, h$. Από τα παραπάνω, το αρχικό πρόβλημα **(3-57)-(3-60)** έχει την ακόλουθη μορφή:

Αντικειμενική συνάρτηση:

$$z_{ILP} = (c^1)^T x^1 + (c^2)^T x^2 + \cdots + (c^h)^T x^h \tag{3-79}$$

Υποκειμενική στους περιορισμούς:

$$A^1 x^1 + A^2 x^2 + \cdots + A^h x^h \geq b \tag{3-80}$$

$$D^1 x^1 \qquad\qquad \geq d^1 \tag{3-81}$$

$$D^2 x^2 \qquad\qquad \geq d^2 \tag{3-82}$$

$$\ddots \qquad\qquad \vdots \tag{3-83}$$

$$D^h x^h \quad \geq d^h \tag{3-84}$$

$$[(x^1)^T \ (x^2)^T \quad \dots \quad (x^h)^T \ ]^T \in \mathbb{Z}_+^n \tag{3-85}$$

Όπως μπορεί να παρατηρηθεί, μόνο οι περιορισμοί **(3-80)** περιέχουν τα στοιχεία του πίνακα A που συνδέουν τις μεταβλητές από διαφορετικά υποσύνολα $k = 1,2, \dots, h$ ενώ καθένας από τους περιορισμούς **(3-81)-(3-84)** μπορεί να αναπαρασταθεί σαν ένα ανεξάρτητο πολύτοπο $X^k = \left\{x^k \in \mathbb{Z}^{|x^k|} \middle| \, D^k x^k \geq d^k, x^k \in \mathbb{Z}_+^n\right\}, k = 1,2, \dots, h$. Σύμφωνα με το θεώρημα Minkowski-Weyl (**§ 2.3**) κάθε $x^k \in X^k, k = 1,2, \dots, h$ μπορεί να αναπαρασταθεί σαν γραμμικός συνδυασμός των ακραίων σημείων $Q^k = \{x_1, x_2, \dots, x_{|Q|}\}$ του X^k(θεωρήθηκε φραγμένο). Αν

επίσης $K = \{1,2, \dots, h\}$ και σύμφωνα με την **§ 3.4.3** οι διατυπώσεις του MP χρησιμοποιώντας **μόνο** την προσέγγιση κυρτότητας είναι οι παρακάτω:

3.4.4.4 Διατύπωση του MP χρησιμοποιώντας την προσέγγιση κυρτότητας:

Αντικειμενική συνάρτηση:

$$z_{MP}^{ILP} = \min \sum_{k \in K} \sum_{q \in Q^k} (c^k)^T \lambda_q^k \, x_q^k \tag{3-86}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k \in K} \sum_{q \in Q^k} A^k \lambda_q^k \, x_q^k \geq b \tag{3-87}$$

$$\sum_{q \in Q^k} \lambda_q^k = 1, k \in K \tag{3-88}$$

$$x^k = \sum_{q \in Q^k} \lambda_q^k x_q^k \tag{3-89}$$

$$\lambda_q^k \geq 0, q \in Q^k \tag{3-90}$$

Όπου, x_q^k είναι τα ακραία σημεία των $conv\ (X^k)$.

Όμοια, το MP **(3-86)-(3-90)**, επιλύεται με τη μέθοδο δημιουργίας στήλης και επομένως, πρέπει να διατυπωθεί το αντίστοιχο RMP. Για κάθε υποσύστημα (subsystem) $k \in K$, θεωρούνται τα υποσύνολα $\bar{Q}^k \subseteq Q^k$ και αντικαθίστανται στο MP **(3-86)-(3-90)**.

3.4.4.5 Διατύπωση του RMP χρησιμοποιώντας την προσέγγιση κυρτότητας:

Αντικειμενική συνάρτηση:

$$z_{RMP}^{ILP} = \min \sum_{k \in K} \sum_{q \in \bar{Q}^k} (c^k)^T \lambda_q^k \, x_q^k \tag{3-91}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k \in K} \sum_{q \in \bar{Q}^k} A^k \lambda_q^k x_q^k \geq b \tag{3-92}$$

$$\sum_{q \in \bar{Q}^k} \lambda_q^k = 1, k \in K \tag{3-93}$$

$$x^k = \sum_{q \in \bar{Q}^k} \lambda_q^k x_q^k \tag{3-94}$$

$$\lambda_q^k \geq 0, q \in \bar{Q}^k \tag{3-95}$$

Αν $(\pi, \sigma^1, \sigma^2, \dots, \sigma^h)$ είναι μία βέλτιστη δυϊκή λύση του RMP, όπου π αντιστοιχεί στους περιορισμούς **(3-92)** και $\sigma^k, k \in K$ αντιστοιχούν στους h περιορισμούς κυρτότητας **(3-93)**, τότε λύνοντας h διαφορετικά υπο-προβλήματα, δημιουργούνται στήλες που προστίθενται στο RMP. Τα h υπο-προβλήματα διατυπώνονται όπως παρακάτω:

$$\bar{c}^{k*} = \min_{x^k \in X^k} \{((c^k)^T - \pi^T A)x^k - \sigma^k\}, k \in K \tag{3-96}$$

Όπως έχει σημειωθεί επανειλημμένες φορές, η διαδικασία της μεθόδου δημιουργίας στήλης επαναλαμβάνεται μέχρι $\bar{c}^{k*} \geq 0, k \in K$, όπου και τερματίζεται.

3.4.5 Σημείωση για την ειδική περίπτωση του δυαδικού ακέραιου γραμμικού προγραμματισμού

Ας θεωρηθεί το πρόβλημα ακέραιου γραμμικού προγραμματισμού **(3-57)-(3-60)**, όπου οι περιορισμοί **(3-60)**, αντικαθίσταται από τον δυαδικό περιορισμό $x \in \{0,1\}^n$. Έτσι, προκύπτει το παρακάτω πρόβλημα δυαδικού ακέραιου γραμμικού προγραμματισμού (Binary Integer Linear Programming, BILP) [81], [82]:

Αντικειμενική συνάρτηση:

$$z = \min\ c^T x \tag{3-97}$$

Υποκειμενική στους περιορισμούς:

$$Dx \geq d \tag{3-98}$$

$$Bx \geq b \tag{3-99}$$

$$x \in \{0,1\}^n \tag{3-100}$$

Όπως και στην **§ 3.4.1** οι περιορισμοί **(3-98)** θεωρείται ότι είναι οι «πολύπλοκοι» περιορισμοί ενώ οι περιορισμοί **(3-99)**, θεωρείται ότι είναι οι «εύκολοι» και ορίζουν το σύνολο $X = \{x \in \{0,1\}^n \,|Dx \geq d\,\}$. Επιπλέον, ας γίνει η υπόθεση ότι το σύνολο X είναι φραγμένο, οπότε δεν υπάρχουν ακραίες ακτίνες. Η διατύπωση του κύριου προβλήματος του BILP **(3-97)-(3-100)**, σύμφωνα με την **§ 3.4.3**, είναι πανομοιότυπο με το MP που δόθηκε στο ***Θεώρημα 3.1*** (προσέγγιση κυρτότητας) [73], [83] και αναφέρεται ξανά όπως παρακάτω:

Αντικειμ*ενική συνάρτησ*η:

$$z_{MP}^{BILP} = \min \sum_{q \in Q} c^T x_q \lambda_q \tag{3-101}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{q \in Q} A x_q \lambda_q \geq b \tag{3-102}$$

$$\sum_{q \in Q} \lambda_q = 1 \tag{3-103}$$

$$\lambda_q \geq 0, q \in Q \tag{3-104}$$

3.4.5.1 Διατύπωση του περιορισμένου κύριου προβλήματος (RMP)

Για ένα υποσύνολο Q' του Q, ορίζεται το παρακάτω RMP:

Αντικειμενική συνάρτηση:

$$z_{RMP}^{BILP} = \min \sum_{q \in Q'} c^T x_q \lambda_q \tag{3-105}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{q \in Q'} A x_q \lambda_q \geq b \tag{3-106}$$

$$\sum_{q \in Q'} \lambda_q = 1 \tag{3-107}$$

$$\lambda_q \geq 0, q \in Q' \tag{3-108}$$

3.4.5.2 Διατύπωση του υπο-προβλήματος

Αν π και σ είναι οι δυικές μεταβλητές που αντιστοιχούν στους περιορισμούς **(3-106)** και **(3-107)** αντίστοιχα, τότε το υπο-πρόβλημα διατυπώνεται όπως παρακάτω:

Αντικειμενική συνάρτηση:

$$z_{sp} = \min \{(c^T - \pi^T A)x - \sigma\} \tag{3-109}$$

Υποκειμενική στους περιορισμούς:

$$Dx \geq d \tag{3-110}$$

$$x \in \{0,1\}^n \tag{3-111}$$

Υπάρχουν δύο περιπτώσεις για τη λύση του υπο-προβλήματος αντί των τριών της **§ 3.3.4** γιατί εδώ δεν υπάρχουν ακραίες ακτίνες. Αυτές είναι οι ακόλουθες:

- Αν $z_{sp} < 0$, προκύπτει ένα ακραίο σημείο $q \in Q$, με κόστος $\bar{c}_q < 0$ και επομένως, η μεταβλητή λ_q προστίθεται στο RMP με κόστος $c_q^T x_q$ και συντελεστές στήλης $\begin{bmatrix} Ax_q \\ 1 \end{bmatrix}$. Δηλαδή, $\begin{bmatrix} c_q^T x_q \\ Ax_q \\ 1 \end{bmatrix}$
- Αν $z_{sp} \geq 0$, έχει βρεθεί βέλτιστη λύση για το MP.

Τέλος, για την γωνιακή δομή πολλών μπλοκ, το κύριο πρόβλημα είναι όμοιο με το MP **(3-86)-(3-90)** όπως αναφέρθηκε στην **§ 3.4.4.3** και αναφέρεται ξανά:

3.4.5.3 Διατύπωση του MP χρησιμοποιώντας την προσέγγιση κυρτότητας

Αντικειμενική συνάρτηση:

$$z_{MP}^{ILP} = \min \sum_{k \in K} \sum_{q \in Q^k} (c^k)^T \lambda_q^k x_q^k \qquad (3\text{-}112)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k \in K} \sum_{q \in Q^k} A^k \lambda_q^k x_q^k \geq b \qquad (3\text{-}113)$$

$$\sum_{q \in Q^k} \lambda_q^k = 1, k \in K \qquad (3\text{-}114)$$

$$\lambda_q \geq 0, q \in Q^k \qquad (3\text{-}115)$$

Όπου, x_q^k είναι τα ακραία σημεία των $conv\ (X^k)$.

3.4.6 Γραφική και οικονομική ερμηνεία της μαθηματικής αποσύνθεσης Dantzig-Wolfe

3.4.6.1 Γραφική ερμηνεία της μαθηματικής αποσύνθεσης Dantzig-Wolfe

Η κύρια ιδέα της χρήσης της αρχής αποσύνθεσης Dantzig-Wolfe είναι να ληφθεί ένα καλύτερο κάτω όριο για ένα δέντρο branch-and-bound παρά να εφαρμοστεί γραμμική χαλάρωση στο αρχικό πρόβλημα. Όπως φαίνεται από την **Εικόνα 3-5** για τα πολύεδρα της

γραμμικής χαλάρωσης (P_{LP}), της αποσύνθεσης DW (P_{DW}) και της γραμμικής χαλάρωσης με περιορισμούς ακεραιότητας (P_{ILP}) ισχύει: $P_{ILP} \subseteq P_{DW} \subseteq P_{LP}$.

3.4.6.2 Θεώρημα 3.3

Ας θεωρηθεί το ακέραιο πρόβλημα **(3-57)-(3-60)** με τουλάχιστον μία εφικτή λύση, τότε ισχύουν πάντα:

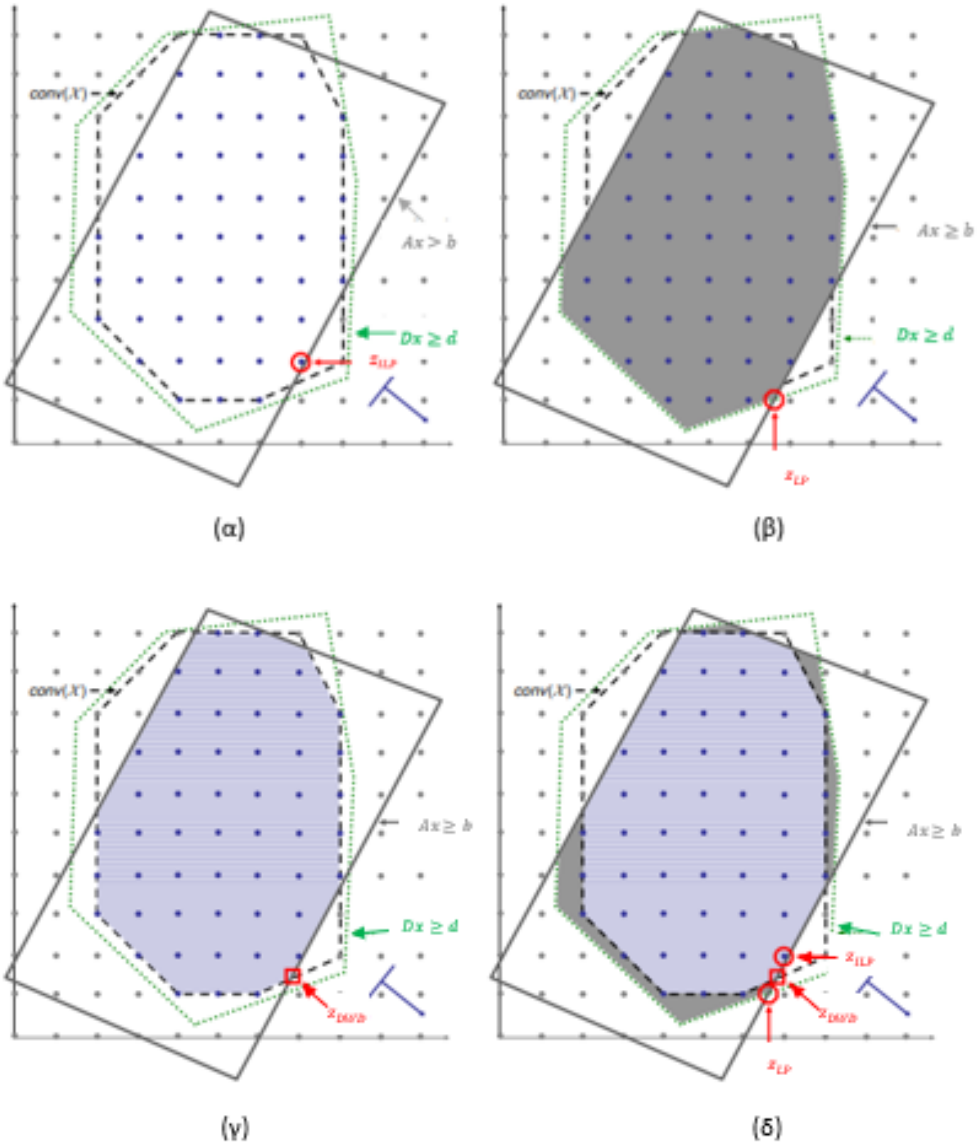
$$z_{LP} \leq z_{DWD} \leq z_{ILP} \qquad (3\text{-}116)$$

Όπου, z_{ILP} αντιπροσωπεύει τη βέλτιστη λύση του **(3-57)-(3-60)**, z_{DWD} είναι η βέλτιστη λύση του κύριου προβλήματος **(3-63)-(3-67)** και z_{LP} είναι η βέλτιστη λύση της γραμμικής χαλάρωσης του **(3-57)-(3-60)**. Η απόδειξη του θεωρήματος βρίσκεται στην αναφορά [84]. Στην **Εικόνα 3-7**, δίνεται μία γραφική ερμηνεία αυτών των αποτελεσμάτων, όπου απεικονίζονται εφικτές και βέλτιστες λύσεις των LP, ILP και DWD. Για την **Εικόνα 3-7 (α)-(δ)** αναφέρονται τα παρακάτω:

- Τα γκρι χρώματος σημεία αναπαριστούν ακέραια σημεία.
- Το περίγραμμα του εφικτού πολυτόπου περιγράφεται από περιορισμούς **(3-58)** ($Ax \geq b$), όπου αναπαρίσταται από μία συμπαγή, γκρι χρώματος γραμμή.
- Για την απεικόνιση του σχήματος του πολύτοπου που σχηματίζεται από τους περιορισμούς $Dx \geq d$, χρησιμοποιείται μια διακεκομμένη πράσινη γραμμή.
- Το $conv(X)$ αντιπροσωπεύεται από μία διακεκομμένη μαύρη γραμμή.
- Η κατεύθυνση προς την οποία η αντικειμενική συνάρτηση ελαχιστοποιείται, υποδεικνύεται από το μπλε βέλος στην κάτω δεξιά γωνία.
- Στην **Εικόνα 3-7 (α)** απεικονίζεται το εφικτό σύνολο του αρχικού προβλήματος ILP **(3-57)-(3-60)** (θεωρώντας ακέραιες λύσεις).
- Στην **Εικόνα 3-7 (β)** παρουσιάζεται η εφικτή περιοχή μετά την εφαρμογή γραμμικής χαλάρωσης (LP) στο ILP **(3-57)-(3-60)**. Είναι σαφές ότι η λύση μπορεί να επιτευχθεί λύνοντας το z_{LP} που είναι ένα κατώτερο όριο στο z_{ILP}, αφού δεν επιβάλλονται ακέραιες λύσεις. Οι εφικτές λύσεις του z_{ILP}περιλαμβάνονται στον εφικτό πολύτοπο που εξετάζεται στο z_{LP}.
- Στην **Εικόνα 3-7 (γ)** απεικονίζεται η αντίστοιχη εφικτή περιοχή μετά την εφαρμογή της DWD στο ILP **(3-57)-(3-60)**. Πρέπει να σημειωθεί ότι το $conv(X)$

διαφέρει από την εφικτή περιοχή που απεικονίζεται στην **Εικόνα 3-7 (β)**. Τονίζεται ότι, όπως και στην προηγούμενη περίπτωση, το εφικτό σύνολο του z_{ILP} περιλαμβάνεται στον πολύτοπο που λαμβάνεται χρησιμοποιώντας την προσέγγιση DWD και επομένως, για τα όρια ικανοποιούνται οι σχέσεις: $z_{ILP} \geq z_{LP}$ και $z_{ILP} \geq z_{DWD}$.

- Τέλος, στην **Εικόνα 3-7 (δ)** απεικονίζεται το κέρδος που λαμβάνεται με την εφαρμογή της DWD έναντι της γραμμικής χαλάρωσης LP στο ILP **(3-57)-(3-60)**. Η εφικτή περιοχή που απεικονίζεται στην **Εικόνα 3-7 (γ)** επικαλύπτεται έναντι της εφικτής περιοχής που απεικονίζεται στην **Εικόνα 3-7 (β)**, όπου η σκιαγραφημένη περιοχή είναι η εφικτή περιοχή που αφαιρέθηκε από την γραμμική χαλάρωση LP χρησιμοποιώντας την DWD. Αυτό δείχνει ότι το εφικτό σύνολο της DWD περιέχεται στο εφικτό πολύτοπο γραμμικής χαλάρωσης LP. Δηλαδή, ισχύει $z_{DWD} \geq z_{LP}$. Γενικά, χρησιμοποιώντας την DWD, λαμβάνεται ένα πιο αυστηρό κατώτερο όριο για το αρχικό πρόβλημα απ' ότι εφαρμόζοντας γραμμική χαλάρωση (LP).



Εικόνα 3-7: Εφικτές λύσεις για (α) το ακέραιο πρόβλημα (IP), (β) τη γραμμική χαλάρωσή του (LP), (γ) της μαθηματικής αποσύνθεσης Dantzig-Wolfe (DWD) και (δ) κέρδος της DWD έναντι της γραμμικής χαλάρωσης (LP) [85].

3.4.6.3 Η οικονομική ερμηνεία της μαθηματικής αποσύνθεσης Dantzig-Wolfe

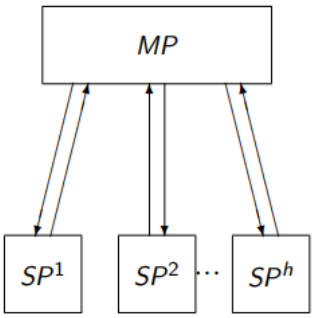
Σύμφωνα με την αναφορά [17], ο ρόλος του κύριου προβλήματος (MP) είναι:

- Να ελέγχει τη χρήση των πόρων: ενημερώνει το υπο-πρόβλημα για την τιμή της χρήσης των πόρων.
- Το κύριο πρόβλημα, «συνδυάζει» τις ανεξάρτητες λύσεις των υπο-προβλημάτων.

- Οι περιορισμοί του κύριου προβλήματος (MP), δίνουν πληροφορία για τον τρόπο με τον οποίο χρησιμοποιούνται οι πόροι από το υπο-πρόβλημα.

Ο ρόλος του/των υπο-προβλήματος/υπο-προβλημάτων:

- Ανταγωνίζονται για πόρους: κάθε υπο-πρόβλημα κάνει την «καλύτερη προσφορά» του στο MP.

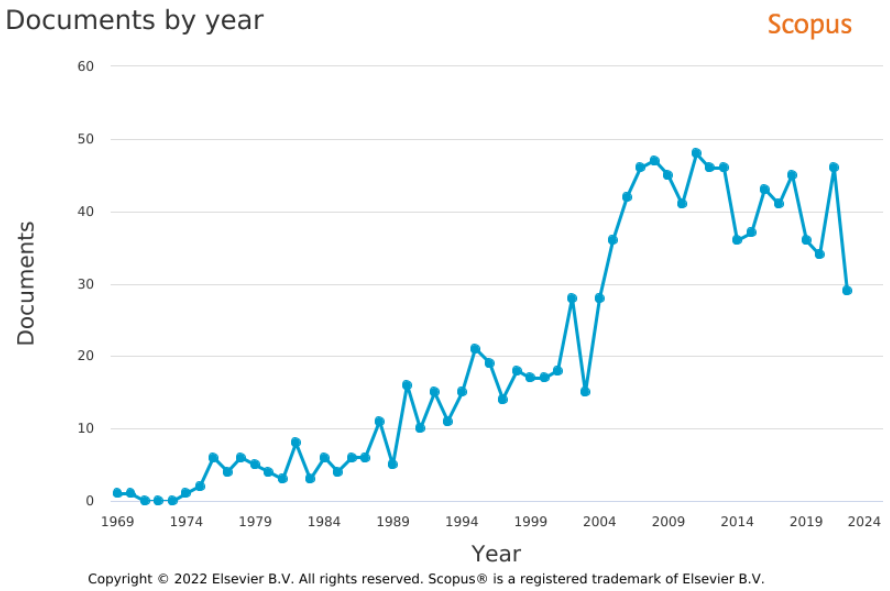


Εικόνα 3-8: Επικοινωνία μεταξύ του κύριου προβλήματος (MP) και των h υπο-προβλημάτων $SP^1, SP^2, \dots, SP^h$ [17].

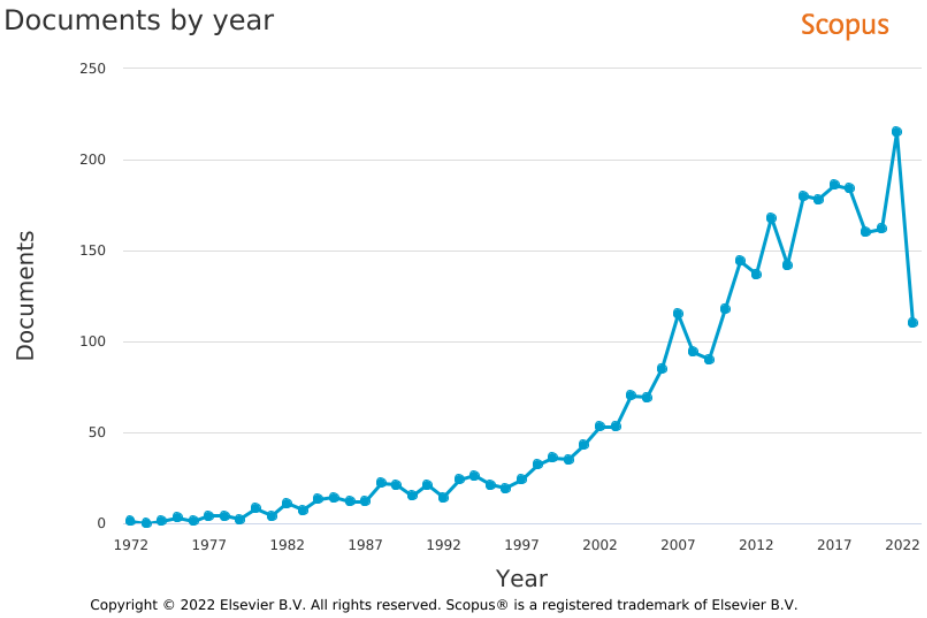
Κεφάλαιο 4. Παραδείγματα εφαρμογής της μεθόδου μαθηματικής αποσύνθεσης Dantzig-Wolfe σε προβλήματα της Επιχειρησιακής Έρευνας

4.1 Εισαγωγή

Στο Κεφάλαιο αυτό, φαίνεται η χρησιμότητα της μεθόδου μαθηματικής αποσύνθεσης Dantzig-Wolfe στην λήψη ισχυρότερων ορίων και στη διάσπαση της συμμετρίας. Αυτό θα γίνει μέσω εφαρμογής της μεθόδου σε δύο προβλήματα βελτιστοποίησης που είναι στενά συνδεδεμένα μεταξύ τους: (α) το μονοδιάστατο πρόβλημα κοπής αποθέματος και (β) το μονοδιάστατο πρόβλημα συσκευασίας κάδου. Αυτά τα δύο προβλήματα επιλέχθηκαν γιατί είναι απλά μοντέλα και φαίνεται καθαρά ο τρόπος λειτουργίας της μεθόδου δημιουργίας στήλης. Επίσης, υπάρχει πλούσια πηγή βιβλιογραφικών αναφορών. Ένα δείγμα αυτής, μπορεί αναλυθεί από μια αναζήτηση στη βάση βιβλιογραφικών δεδομένων «Scopus» για άρθρα που έχουν στον τίτλο τον όρο «cutting» ή «packing» ή και τους δύο, προκύπτουν τα δεδομένα που παρουσιάζεται στην **Εικόνα 4-1** και στην **Εικόνα 4-2**.



Εικόνα 4-1: Αριθμός δημοσιεύσεων σε συνάρτηση με το έτος έκδοσης για το πρόβλημα κοπής αποθέματος, 1970-2022 (Scopus) [46].



Εικόνα 4-2: Αριθμός δημοσιεύσεων σε συνάρτηση με το έτος έκδοσης για το πρόβλημα συσκευασίας κάδου, 1972-2022 (Scopus) [46].

4.2 Το πρόβλημα κοπής αποθέματος

Στην επιχειρησιακή έρευνα, το πρόβλημα κοπής αποθέματος (CSP) είναι το πρόβλημα της κοπής τεμαχίων τυποποιημένου μεγέθους υλικού αποθέματος (stock), όπως κύλινδροι (rolls) χαρτιού, λαμαρίνας (sheet metal) ή μεταλλικών ράβδων, σε κομμάτια καθορισμένων μεγεθών, που ονομάζονται πρότυπα (patterns) και στόχος του είναι η ελαχιστοποίησης της σπατάλης υλικού (scrap). Είναι ένα πρόβλημα βελτιστοποίησης που προκύπτει από εφαρμογές στη βιομηχανία. Όσον αφορά την υπολογιστική πολυπλοκότητα, το πρόβλημα είναι ένα πρόβλημα NP-hard που μπορεί να αναχθεί στο πρόβλημα του σακιδίου (knapsack problem) και μπορεί να διατυπωθεί ως ακέραιο πρόβλημα γραμμικού προγραμματισμού [86].

Εικόνα 4-3: (α) απόθεμα κυλίνδρων λαμαρίνας και (β) απόθεμα μεταλλικών ράβδων στη βιομηχανία [87].

4.2.1 Ταξινόμηση του προβλήματος κοπής αποθέματος και οι αντίστοιχες εφαρμογές

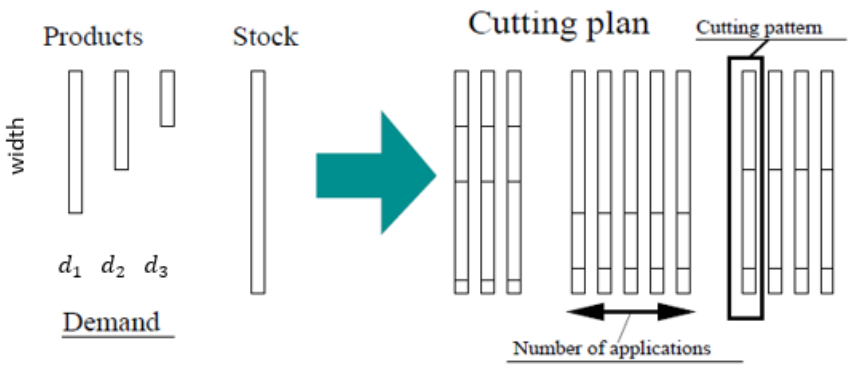
Τα προβλήματα κοπής μπορούν να ταξινομηθούν με διάφορους τρόπους. Ένας τρόπος είναι η διάσταση της κοπής. Έτσι υπάρχουν οι ακόλουθες τρεις κατηγορίες [86]:

- Το μονοδιάστατο πρόβλημα κοπής (1D cutting stock problem), που εμφανίζεται κατά την κοπή σωλήνων, καλωδίων και μεταλλικών ράβδων (π.χ. **Εικόνα 4-3** (β)).
- Το δισδιάστατο πρόβλημα κοπής (2D cutting stock problem), που εμφανίζεται στην παραγωγή επίπλων, ενδυμάτων και γυαλιού.
- Το τρισδιάστατο πρόβλημα κοπής (3D cutting stock problem). Δεν είναι γνωστές πολλές τρισδιάστατες εφαρμογές. Ωστόσο, το στενά συνδεδεμένο πρόβλημα (packing problem) της τρισδιάστατης συσκευασίας έχει πολλές βιομηχανικές εφαρμογές, όπως η συσκευασία αντικειμένων σε εμπορευματοκιβώτια μεταφοράς.

Από τις παραπάνω κατηγορίες στην παρούσα εργασία περιγράφεται το μονοδιάστατο πρόβλημα κοπής.

4.2.2 Το μονοδιάστατο πρόβλημα κοπής αποθέματος

Στο 1D πρόβλημα κοπής αποθέματος (1D CSP), δίνονται m τύποι ειδών (items), καθένας από τους οποίους έχει ένα ακέραιο βάρος (πλάτος) w_i, υπάρχουν d_i $(i = 1,2,\ldots,m)$ παραγγελίες και υπάρχει επίσης ένας μέγιστος αριθμός K πανομοιότυπων κυλίνδρων (rolls) ακέραιου πλάτους W. Ο στόχος είναι να δημιουργηθούν m αντίγραφα για κάθε είδος τύπου i χρησιμοποιώντας τον ελάχιστο αριθμό κυλίνδρων, έτσι ώστε το συνολικό βάρος (των τύπων i) που προστίθεται σε οποιονδήποτε κύλινδρο να μην υπερβαίνει το πλάτος του (W).



Εικόνα 4-4: Μονοδιάστατο πρόβλημα κοπής αποθέματος με έναν τύπο αποθέματος [88].

4.2.3 Διατύπωση του μονοδιάστατου προβλήματος κοπής αποθέματος

Ο Kantorovich (1960) εισήγαγε το ακόλουθο μοντέλο μαθηματικού προγραμματισμού για το 1D-CSP [89], [90]:

Αντικειμενική συνάρτηση:

$$z = \min \sum_{k=1}^{K} y_k \tag{4-1}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k=1}^{K} x_{ik} \geq d_i, i = 1,2, \dots, m \tag{4-2}$$

$$\sum_{i=1}^{m} w_i x_{ik} \leq W y_k, k = 1,2, \dots, K \tag{4-3}$$

$$y_k \in \{0,1\}, k = 1,2, \dots, K \tag{4-4}$$

$$x_{ik} \geq 0, i = 1,2, \dots, m \text{ και } k = 1,2, \dots, K \tag{4-5}$$

Όπου, $y_k = 1$ αν ο κύλινδρος k χρησιμοποιείται, αλλιώς $y_k = 0$. Η ακέραια μεταβλητή x_{ik} δηλώνει πόσες φορές ένα κομμάτι i πλάτους w_i κόβεται από τον κύλινδρο k. Οι περιορισμοί **(4-2)** εξασφαλίζουν ότι όλες οι απαιτήσεις ικανοποιούνται ενώ οι περιορισμοί

(4-3) εγγυόνται ότι το άθροισμα του πλάτους όλων των τεμαχίων που λαμβάνονται από ένα κύλινδρο δεν υπερβαίνει το πλάτος του W. Η ομάδα των περιορισμών (4-3), αναφέρονται ως περιορισμοί σακιδίου (knapsack constraints). Όταν $y_k = 1$, είναι ακριβώς ένας περιορισμός σακιδίου με χωρητικότητα W και όταν $y_k = 0$, είναι ένας περιορισμός σακιδίου με χωρητικότητα 0.

Ένα κατώτερο όριο (LB) για τη βέλτιστη λύση, μπορεί να ληφθεί από τη λύση της γραμμικής χαλάρωσης (LP), που προκύπτει από την αντικατάσταση των δύο τελευταίων περιορισμών (4-4) και (4-5) για $0 \leq y_k \leq 1$ και $x_{ik} \geq 0$. Ωστόσο, σύμφωνα με τους Martello και Toth [91], αυτό το όριο είναι ασθενές. Οι Martello και Toth έδειξαν ότι αυτό το κατώτερο όριο LP ισούται με $\sum_{i=0}^{m} \frac{w_i d_i}{W}$. [Στην πραγματικότητα, το απέδειξαν για το μονοδιάστατο πρόβλημα συσκευασίας κάδου (one dimensional bin packing problem) αλλά, το αποτέλεσμά τους μπορεί εύκολα να επεκταθεί για το 1D-CSP], [91].

Η ερμηνεία αυτού του κατώτερου ορίου είναι η ακόλουθη: ισούται με τον ελάχιστο χώρο που είναι απαραίτητος για να φιλοξενηθούν όλα τα είδη αν υπάρχει τρόπος να επαναχρησιμοποιηθούν τα απόβλητα κολλώντας τα μεταξύ τους. Ως εκ τούτου, αυτό το όριο είναι πολύ φτωχό σε περιπτώσεις με μεγάλα απόβλητα [89]. Για παράδειγμα όταν $w_i = \frac{W}{2} + \epsilon, \forall i$, η βέλτιστη λύση είναι $\sum_{i=0}^{m} d_i$ και το διάκενο (gap) μεταξύ της βέλτιστης λύσης και του κατώτερου ορίου προσεγγίζει το 50%.

Η ποιότητα των κατώτερων ορίων είναι ζωτικής σημασίας όταν χρησιμοποιούνται προσεγγίσεις που βασίζονται στον γραμμικό προγραμματισμό για την επίλυση ακέραιων προβλημάτων. Οι P. H. Vance [92] και Valerio de Carvalho [93] αναφέρουν ότι οι αλγόριθμοι branch-and bound που βασίζονται σε αυτό το μοντέλο απέτυχαν να λύσουν με βέλτιστο τρόπο ορισμένα παραδείγματα των προβλημάτων συσκευασίας κάδου και κοπής αποθέματος, αντίστοιχα. Ωστόσο, το ασθενές όριο, μπορεί να αντιμετωπιστεί χρησιμοποιώντας ισχυρότερες αναδιατυπώσεις, όπως η μαθηματική αποσύνθεση Dantzig-Wolfe, που περιγράφεται στην ακόλουθη παράγραφο [94].

4.2.4 Εφαρμογή της μαθηματικής αποσύνθεσης Dantzig-Wolfe στο μονοδιάστατο πρόβλημα κοπής αποθέματος

Για την διάσπαση της συμμετρίας και την λήψη καλύτερου ορίου από τη γραμμική χαλάρωση, μπορεί να εφαρμοστεί η μαθηματική αποσύνθεση Dantzig-Wolfe στο μοντέλο Kantorovich **(4-1)-(4-5)**. Σύμφωνα με τον P. H. Vance [92] αυτό γίνεται ως εξής: παρατηρώντας τη δομή των περιορισμών του προβλήματος **(4-1)-(4-5)**, οι περιορισμοί **(4-2)** είναι περιορισμοί σακιδίου, επομένως το πρόβλημα **(4-1)-(4-5)** θα μπορούσε να επιλυθεί πιο εύκολα αν γίνει εκμετάλλευση αυτής της δομής. Έτσι, οι περιορισμοί **(4-2)** διατηρούνται στο κύριο πρόβλημα (πολύπλοκοι περιορισμοί) και οι περιορισμοί σακιδίου **(4-3)** μαζί με τους περιορισμούς **(4-4)** και **(4-5)** χρησιμοποιούνται για τον ορισμό του υπο-προβλήματος (εύκολοι περιορισμοί) [92]. Έτσι, σε αντιστοιχία με την **§ 3.4**, οι περιορισμοί **(4-3)-(4-5)** ορίζουν το σύνολο X.

Το μοντέλο Kantorovich έχει πολλούς περιορισμούς σακιδίου, έναν για κάθε κύλινδρο k. Κάθε περιορισμός σακιδίου, ορίζει ένα φραγμένο σύνολο, που έχει μόνο ακραία σημεία και όχι ακραίες ακτίνες. Το σύνολο X είναι η τομή των περιορισμών σακιδίου για κάθε κύλινδρο. Το σύνολο λύσεων του περιορισμού σακιδίου του κυλίνδρου k, είναι ένα πολύτοπο που μπορεί να έχει κλασματικά ακραία σημεία. Αν προσδιοριστεί η βέλτιστη ακέραια λύση των υπο-προβλημάτων του σακιδίου, τότε οι έγκυρες κλασματικές λύσεις των περιορισμών του σακιδίου που δεν αντιστοιχούν σε έγκυρα σχέδια κοπής για τον κύλινδρο k εξαλείφονται. Λαμβάνονται υπόψη μόνο τις ακέραιες λύσεις του περιορισμού σακιδίου για τον κύλινδρο k τέτοιες ώστε $\sum_{i=1}^{m} w_i a_{ik} \leq W, \text{όπου } a_{ik} \geq 0$. Αυτές οι λύσεις περιγράφονται από το διάνυσμα $\left(a_{ik}^p, \dots, a_{ik}^p, \dots, a_{mk}^p\right), \forall\, p \in P$, όπου P είναι το σύνολο των έγκυρων μοτίβων κοπής.

Για τη διατύπωση του κύριου προβλήματος, θα χρησιμοποιηθεί η προσέγγιση κυρτότητας και επομένως, χρησιμοποιούνται τα ακραία σημεία του $conv\ (X)$.

Σύμφωνα με το θεώρημα Minkowski-Weyl:

$$x_{ik} = \sum_{p \in P} a_{ik}^p \lambda_k^p \qquad (4\text{-}6)$$

$$\sum_{p \in P} \lambda_k^p = 1 \qquad (4\text{-}7)$$

$$\lambda_k^p \geq 0 \qquad (4\text{-}8)$$

Η ακέραια μεταβλητή y_k για κάθε κύλινδρο k, ορίζεται συγκεντρωτικά ως [25],[89]:

$$y_k = \sum_{p \in P} \lambda_k^p \tag{4-9}$$

Με την αντικατάσταση των **(4-6)-(4-9)** στις σχέσεις **(4-1)**, **(4-2)**, **(4-4)** και **(4-5)** του μοντέλου Kantorovich, προκύπτει η διατύπωση του κύριου προβλήματος (με τη χρήση της προσέγγισης κυρτότητας) [89]:

Αντικειμενική συνάρτηση:

$$\min \sum_{k=1}^{K} \sum_{p \in P} \lambda_k^p \tag{4-10}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k=1}^{K} \sum_{p \in P} a_{ik}^p \lambda_k^p \geq d_i, i = 1,2, \dots, m \tag{4-11}$$

$$\sum_{p \in P} \lambda_k^p = 1, k = 1,2, \dots, K \tag{4-12}$$

$$\sum_{p \in P} a_{ik}^p \lambda_k^p \geq 0, i = 1,2, \dots, m, k = 1,2, \dots, K \tag{4-13}$$

$$\sum_{p \in P} \lambda_k^p \geq 0, k = 1,2, \dots, K \tag{4-14}$$

$$\lambda_k^p \geq 0, p \in P, k = 1,2, \dots, K \tag{4-15}$$

Στο μοντέλο αυτό, μεταβλητές απόφασης είναι τα βάρη του κυρτού συνδυασμού των έγκυρων προτύπων κοπής λ_k^p και εκφράζουν τον αριθμό των φορών που το πρότυπο p παράγεται από τον κύλινδρο k.

Η αντίστοιχη στήλη στο αναδιατυπωμένο μοντέλο έχει μηδενικούς συντελεστές στους περιορισμούς **(4-11)** και συντελεστή 1 στον περιορισμό κυρτότητας για τον κύλινδρο k, και επομένως έχει τη δομή μιας χαλαρής μεταβλητής. Όταν παίρνει την τιμή 1, σημαίνει ότι ο

κύλινδρος k δεν κόβεται. Οι περιορισμοί **(4-13)** υπονοούνται από τους περιορισμούς **(4-14)** και από το γεγονός ότι όλα τα a_{ik}^{p} είναι ακέραια.

Μπορεί να φανεί ότι η χαλάρωση LP αυτού του αναδιατυπωμένου μοντέλου έχει μόνο εκείνες τις εφικτές λύσεις που είναι μη αρνητικός γραμμικός συνδυασμός των ακέραιων λύσεων του προβλήματος σακιδίου. Οι μη αρνητικοί γραμμικοί συνδυασμοί των κλασματικών ακραίων σημείων του πολυτόπου του σακιδίου, που είναι εφικτά στη γραμμική χαλάρωση του μοντέλου Kantorovich εξαλείφονται. Λόγω αυτού του γεγονότος, το αναδιατυπωμένο μοντέλο **(4-10)-(4-15)** δίνει καλύτερο όριο της χαλάρωσης LP.

Η χαλάρωση LP του αναδιατυπωμένου μοντέλου δίνει ένα ισχυρότερο κάτω φράγμα για το 1D-CSP, γιατί οι μόνες εφικτές λύσεις είναι αυτές που είναι μη αρνητικοί γραμμικοί συνδυασμοί των ακέραιων λύσεων του προβλήματος σακιδίου. Με τον τρόπο αυτό, εξαλείφονται οι μη αρνητικοί γραμμικοί συνδυασμοί των κλασματικών ακραίων σημείων του πολυτόπου του σακιδίου, τα οποία είναι εφικτά στη χαλάρωση LP του μοντέλου Kantorovich.

Το υπο-πρόβλημα που ορίζεται από τους περιορισμούς **(4-3)-(4-5)**, αποσυντίθεται σε $|K|$ υπο-προβλήματα, που είναι προβλήματα σακιδίου. Στην ιδανική περίπτωση στην οποία όλοι οι κύλινδροι έχουν το ίδιο πλάτος, όλα τα υπο-προβλήματα είναι πανομοιότυπα και το μοντέλο **(4-10)-(4-15)**, είναι ισοδύναμο με το κλασικό μοντέλο των Gilmore-Gomory [89].

Οι Gilmore-Gomory [65], [66] πρότειναν ένα μοντέλο στο οποίο τα πιθανά πρότυπα κοπής (cutting patterns) περιγράφονται από το ακέραιο διάνυσμα $A_p = \left(a_{1p}, \cdots, a_{ip}, \cdots a_{mp}\right)^T$, όπου το στοιχείο a_{ip} αντιπροσωπεύει τον αριθμό των κυλίνδρων πλάτους w_i που λαμβάνονται από το πρότυπο κοπής p. Αν x_p είναι η μεταβλητή απόφασης που υποδεικνύει τον αριθμό των κυλίνδρων που θα κοπούν σύμφωνα με το πρότυπο κοπής p, λ_p είναι η μεταβλητή απόφασης που προσδιορίζει τον αριθμό των κυλίνδρων που πρέπει να κοπούν σύμφωνα με πρότυπο κοπής p. Το μοντέλο των Gilmore-Gomory διατυπώνεται ως εξής [89], [90]:

Αντικειμενική συνάρτηση:

$$\min \sum_{p \in P} \lambda_p \qquad (4\text{-}16)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{p \in P} a_{ip} \lambda_p \geq d_i, i = 1,2, \dots, m \qquad (4\text{-}17)$$

$$\lambda_p \geq 0, \forall\, p \in P \qquad (4\text{-}18)$$

Όπου, ο στόχος είναι η ελαχιστοποίηση του αριθμού του χρησιμοποιούμενου αποθέματος.

Δηλαδή, το μοντέλο των Gilmore-Gomory αποκτάται με την εφαρμογή της μαθηματικής αποσύνθεσης Dantzig-Wolfe στο μοντέλο Kantorovich (αρκεί οι κύλινδροι να έχουν το ίδιο πλάτος) και έχει τις παρακάτω δύο παραμέτρους και μία μεταβλητή απόφασης:

<u>Παράμετροι:</u>

a_{ip}: αριθμός των κυλίνδρων πλάτους w_i που λαμβάνονται από το πρότυπο κοπής p.

d_i: ζήτηση παραγγελίας με πλάτος $w_i, i = 1,2, \dots, m$

<u>Μεταβλητή απόφασης:</u>

λ_p : αριθμός κυλίνδρων που κόβονται χρησιμοποιώντας το πρότυπο p.

Ο αριθμός των στηλών στη διατύπωση των Gilmore-Gomory **(4-16)-(4-18)** είναι εκθετικός ακόμα και για προβλήματα μέτριου μεγέθους, μπορεί να είναι πολύ μεγάλος. Για το λόγο αυτό, οι Gilmore-Gomory πρότειναν τη μέθοδο δημιουργίας στήλης για την επίλυση της χαλάρωσης LP, με το μοντέλο **(4-16)-(4-18)** να αποτελεί το κύριο πρόβλημα. Η αρχικοποίηση πραγματοποιείται με ένα σύνολο προτύπων κοπής (για παράδειγμα, το καθένα με πολλαπλά αντίγραφα του ίδιου είδους σε ποσότητες $\left\lfloor \frac{W}{w_i}, \forall i \right\rfloor$). Στη συνέχεια, σύμφωνα με την **§ 3.2.3** διατυπώνεται το περιορισμένο κύριο πρόβλημα (RMP) σε ένα $P' \subseteq P$ δηλαδή το RMP ορίζεται σε ένα υποσύνολο των έγκυρων προτύπων κοπής (cutting patterns). Η διατύπωση του RMP είναι η εξής:

Αντικειμενική συνάρτηση:

$$\min \sum_{p \in P'} \lambda_p \qquad (4\text{-}19)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{p \in P'} a_{ip} \lambda_p \geq d_i, i = 1,2, \dots, m \qquad (4\text{-}20)$$

$$\lambda_p \geq 0, \forall\, p \in P' \qquad (4\text{-}21)$$

Αν π_i είναι οι δυϊκές μεταβλητές (μία για κάθε ζήτηση) που αντιστοιχούν στους περιορισμούς **(4-20)** το υπο-πρόβλημα είναι το ακόλουθο πρόβλημα σακιδίου (knapsack problem):

Αντικειμενική συνάρτηση:

$$\bar{c}_p = \max \sum_{i=1}^{m} \pi_i a_i \qquad (4\text{-}22)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{i=1}^{m} a_i w_i \leq W, i = 1,2, \dots, m \qquad (4\text{-}23)$$

$$a_i \in \mathbb{Z}_+ \qquad (4\text{-}24)$$

Όπου, a_i είναι πλέον η μεταβλητή απόφασης του υπο-προβλήματος, που καθορίζει τον αριθμό των κομματιών μήκους i που κόβονται στο νέο πρότυπο (pattern).

Με την επίλυση του RMP **(4-19)-(4-21)** προκύπτει η αρχική βέλτιστη λύση λ και οι δυϊκές βέλτιστες λύσεις $\pi^T = (\pi_1, \pi_2, \dots, \pi_m)$.

Το πρόβλημα σακιδίου **(4-22)-(4-24)** αντιστοιχεί στην εύρεση της στήλης με το ελάχιστο μειωμένο κόστος $\bar{c}_p = 1 - \pi_i A_p$. Αν $1 - \bar{c}_p < 0$, προστίθεται μία στήλη (που είναι ένα καινούριο πρότυπο, new pattern) στο περιορισμένο κύριο πρόβλημα, το οποίο βελτιστοποιείται εκ νέου, διαφορετικά (αν $1 - \bar{c}_p \geq 0$), η λύση είναι βέλτιστη (η μεταβλητή του κύριου προβλήματος αντιστοιχεί σε εφικτό πρότυπο, pattern).

Η επίλυση του 1D-CSP με τη μέθοδο δημιουργίας στήλης περιλαμβάνει τα ακόλουθα βήματα:

- **Βήμα 1:** Επιλογή ενός αρχικού συνόλου προτύπων (patterns).
- **Βήμα 2:** Επίλυση του RMP **(4-19)-(4-21)**.
- **Βήμα 3:** Με τη χρήση των δυϊκών τιμών π_i που αντιστοιχούν στους περιορισμούς **(4-20)** του RMP **(4-19)-(4-21)**, πρέπει να επιλυθεί το υπο-πρόβλημα **(4-22)-(4-24)** που είναι πρόβλημα σακιδίου. Αν $1 - \bar{c}_p < 0$, προστίθεται μία στήλη (που είναι ένα καινούριο

πρότυπο, new pattern) στο περιορισμένο κύριο πρόβλημα, το οποίο βελτιστοποιείται εκ νέου (μετάβαση στο **Βήμα 2**), διαφορετικά αν $1 - \bar{c}_p \geq 0$, η λύση είναι βέλτιστη.

$$
\begin{array}{llllll}
min & \sum_{p \in P'} 1\,\lambda_p + & \boxed{1}\,\lambda_{p^*} + & \boxed{1}\,\lambda_{p^{**}} + & \cdots & \\
s.t. & \sum_{p \in P'} a_{1p}\lambda_p + & a_{1p*}\,\lambda_{p^*} + & a_{1p**}\,\lambda_{p^{**}} + & \cdots & \geq d_1 \\
 & \sum_{p \in P'} a_{2p}\lambda_p + & a_{2p*}\,\lambda_{p^*} + & a_{2p**}\,\lambda_{p^{**}} + & \cdots & \geq d_2 \\
 & \vdots & \vdots & \vdots & \vdots & \vdots \\
 & \sum_{p \in P'} a_{mp}\lambda_p + & a_{mp*}\,\lambda_{p^*} + & a_{mp**}\,\lambda_{p^{**}} + & \cdots & \geq d_m \\
 & \lambda_p, & \lambda_{p^*}, & \lambda_{p^{**}}, & \cdots & \geq 0 \quad \forall\, p \in P'
\end{array}
$$

Εικόνα 4-5: Προσθήκη στηλών στο περιορισμένο κύριο πρόβλημα του μονοδιάστατου προβλήματος κοπής αποθέματος. Όπου, λ_{p^*} είναι η βέλτιστη τιμή της μεταβλητής λ_p στην πρώτη επανάληψη, $\lambda_{p^{**}}$ είναι η βέλτιστη τιμή της μεταβλητής λ_p στην δεύτερη επανάληψη. Οι συντελεστές a_{ip} λαμβάνονται από την επίλυση του προβλήματος σακιδίου.

4.3 Το πρόβλημα συσκευασίας κάδου

Το πρόβλημα συσκευασίας κάδου (Bin Packing Problem, BPP) είναι ένα NP-hard πρόβλημα βελτιστοποίησης, στο οποίο αντικείμενα διαφορετικών μεγεθών πρέπει να συσκευάζονται σε έναν πεπερασμένο αριθμό κάδων ή δοχείων, καθένα από μια σταθερή δεδομένη χωρητικότητα, με τρόπο που ελαχιστοποιεί τον αριθμό των κάδων που χρησιμοποιούνται [95].

4.3.1 Είδη του προβλήματος συσκευασίας κάδου και αντίστοιχες εφαρμογές

4.3.1.1 Είδη του προβλήματος συσκευασίας κάδου

Υπάρχουν πολλές παραλλαγές αυτού του προβλήματος, όπως η συσκευασία 2D, η συσκευασία 3D, η γραμμική συσκευασία, η συσκευασία κατά βάρος, η συσκευασία κατά κόστος κ.λπ. Το πρόβλημα συσκευασίας κάδου μπορεί επίσης να θεωρηθεί ως μια ειδική περίπτωση του προβλήματος του αποθέματος κοπής. Όταν ο αριθμός των δοχείων

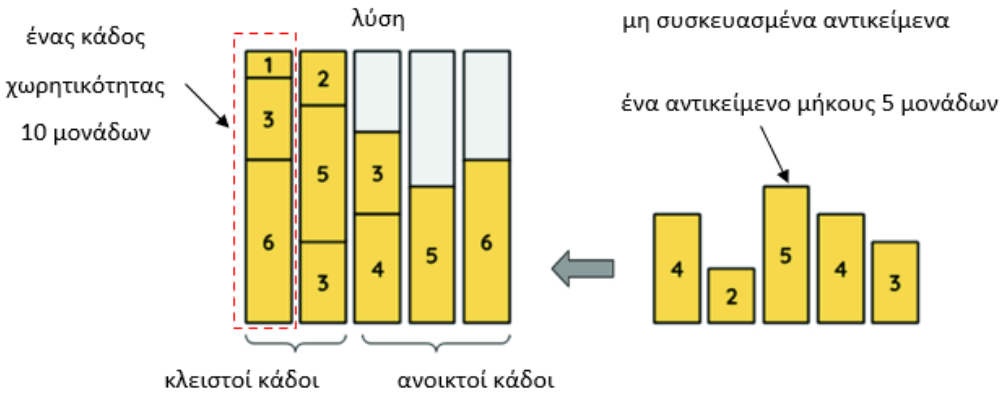
περιορίζεται σε 1 και κάθε στοιχείο χαρακτηρίζεται τόσο από όγκο όσο και από τιμή, το πρόβλημα της μεγιστοποίησης της αξίας των αντικειμένων που μπορούν να χωρέσουν στον κάδο είναι γνωστό ως πρόβλημα σακιδίου [95].

4.3.1.2 Εφαρμογές

Το πρόβλημα έχει πολλές εφαρμογές, όπως γέμισμα εμπορευματοκιβωτίων, φόρτωση φορτηγών με περιορισμούς χωρητικότητας βάρους (3D bin packing problem), δημιουργία αντιγράφων ασφαλείας αρχείων σε μέσα και χαρτογράφηση τεχνολογίας στο σχεδιασμό τσιπ (chip) ημιαγωγών FPGA (Field-Programmable Gate Array) [95].

4.3.2 Το μονοδιάστατο πρόβλημα συσκευασίας κάδου

Το μονοδιάστατο πρόβλημα συσκευασίας κάδου, μπορεί να οριστεί ανεπίσημα με πολύ απλό τρόπο. Δίνονται: n αντικείμενα που το καθένα έχει ένα ακέραιο βάρος $w_i, i = 1,2, \dots, m$ και ένας απεριόριστος αριθμός πανομοιότυπων δοχείων ακέραιας χωρητικότητας c. Ο στόχος του προβλήματος είναι να συσκευαστούν όλα τα αντικείμενα χρησιμοποιώντας τον ελάχιστο δυνατό αριθμό κάδων έτσι ώστε το συνολικό βάρος που συσκευάζεται σε οποιονδήποτε κάδο να μην υπερβαίνει τη χωρητικότητά του [96], [97].



Εικόνα 4-6: Παράδειγμα της διαδικασίας επίλυσης ενός 1D-BPP 15 στοιχείων με μήκη μεταξύ 1 και 6 μονάδων και κάδους χωρητικότητας 10 μονάδων. Αυτή τη στιγμή, δέκα αντικείμενα έχουν συσκευαστεί σε πέντε κάδους (δύο κλειστοί και τρεις ανοιχτοί) και πέντε αντικείμενα αναμένουν να συσκευαστούν [98].

Έστω K είναι ο μέγιστος αριθμός κάδων. Επίσης, εισάγονται οι ακόλουθες δυαδικές μεταβλητές:

$$y_k = \begin{cases} 1, \alpha\nu\ o\ \kappa\acute{\alpha}\delta o\varsigma\ k\ \chi\rho\eta\sigma\iota\mu o\pi o\iota\acute{\epsilon}\iota\tau\alpha\iota\ \sigma\tau\eta\ \lambda\acute{\upsilon}\sigma\eta\ (k = 1,2,\dots,K) \\ 0, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

$$x_{ik} = \begin{cases} 1, \alpha\nu\ \tau o\ \alpha\nu\tau\iota\kappa\epsilon\acute{\iota}\mu\epsilon\nu o\ i\ \sigma\upsilon\sigma\kappa\epsilon\upsilon\acute{\alpha}\zeta\epsilon\tau\alpha\iota\ \sigma\tau o\nu\ \kappa\acute{\alpha}\delta o\ k\ (i = 1,2,\dots,m; k = 1,2,\dots,K) \\ 0, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Επομένως, το BPP μπορεί να μοντελοποιηθεί με το ακόλουθο πρόγραμμα ακέραιου γραμμικού προγραμματισμού, σύμφωνα με τους Martello και Toth [91]:

Αντικειμενική συνάρτηση:

$$z = \min \sum_{k=1}^{K} y_k \qquad (4\text{-}25)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k=1}^{K} x_{ik} = 1, i = 1,2,\dots,m \qquad (4\text{-}26)$$

$$\sum_{i=1}^{n} w_i x_{ik} \leq c y_k, k = 1,2,\dots,K \qquad (4\text{-}27)$$

$$y_k \in \{0,1\}, k = 1,2,\dots,K \qquad (4\text{-}28)$$

$$x_{ik} \in \{0,1\}, i = 1,2,\dots,m\ \kappa\alpha\iota\ k = 1,2,\dots,K \qquad (4\text{-}29)$$

Οι περιορισμοί **(4-26)**, εξασφαλίζουν ότι όλα τα αντικείμενα είναι συσκευασμένα, ενώ οι περιορισμοί **(4-27)** διασφαλίζουν ότι η χωρητικότητα οποιουδήποτε χρησιμοποιημένου κάδου δεν υπερβαίνεται.

Πρόκειται δηλαδή για μία ειδική περίπτωση του μονοδιάστατου προβλήματος κοπής αποθέματος της **§ 4.2.** Το μοντέλο **(4-25)-(4-29)** είναι περίπου όμοιο με τη διατύπωση Kantorovich για το CSP **(4-1)-(4-5)**. Η διαφορά είναι ότι στο BPP δεν υπάρχουν οι απαιτήσεις $d_i, i = 1,2,\dots,m$. Στη βιβλιογραφία του CSP οι κάδοι ονομάζονται συχνά κύλινδροι (rolls) και χρησιμοποιείται η λέξη «κοπή» («cutting») αντί της «συσκευασίας» («packing»).

Εγκαταλείποντας την υπόθεση της ακεραιότητας, λαμβάνεται το ακόλουθο μοντέλο γραμμικής χαλάρωσης (LP):

Αντικειμενική συνάρτηση:

$$z = \min \sum_{k=1}^{K} y_k \tag{4-30}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k=1}^{K} x_{ik} = 1, i = 1,2, \dots, m \tag{4-31}$$

$$\sum_{i=1}^{n} w_i x_{ik} \leq c y_k, k = 1,2, \dots, K \tag{4-32}$$

$$y_k \in [0,1], k = 1,2, \dots, K \tag{4-33}$$

$$x_{ik} \in \{0,1\}, i = 1,2, \dots, m \; \kappa\alpha\iota \; k = 1,2, \dots, K \tag{4-34}$$

Το μοντέλο έχει ένα τετριμμένο LP όριο:

$$LP = \frac{\sum_{i=1}^{n} w_i}{c} \tag{4-35}$$

$$y_k = \frac{LP}{K}, k = 1,2, \dots, K \tag{4-36}$$

$$x_{ik} = \frac{1}{K}, i = 1,2, \dots, m; k = 1,2, \dots, K \tag{4-37}$$

Όμως, αυτό το LP όριο είναι ασθενές και επίσης η διατύπωση του 1D-BPP χαρακτηρίζεται από συμμετρία.

Για την αντιμετώπιση των επιπτώσεων του αδύναμου LP που λαμβάνεται από τη συμπαγή σύνθεση του, μπορεί να εφαρμοστεί η μαθηματική αποσύνθεση Dantzig-Wolfe στο μοντέλο **(4-25)-(4-29)**.

4.3.3 Εφαρμογή της μαθηματικής αποσύνθεσης Dantzig-Wolfe στο μονοδιάστατο πρόβλημα συσκευασίας κάδου

Για την εφαρμογή της μαθηματικής αποσύνθεσης Dantzig-Wolfe, όπως και στο 1D-CSP στο μοντέλο **(4-25)-(4-29)**, γίνεται εκμετάλλευση της δομής των περιορισμών **(4-27)** που είναι περιορισμοί σακιδίου. Έτσι, οι περιορισμοί **(4-26)** διατηρούνται στο κύριο πρόβλημα και οι περιορισμοί **(4-27)-(4-29)**, χρησιμοποιούνται για τον ορισμό του υπο-προβλήματος [92]. Έτσι, σύμφωνα με την **§ 3.4**, οι περιορισμοί **(4-27)-(4-29)** ορίζουν το σύνολο X, που είναι ένα πολύεδρο σακιδίου [91]. Αυτός ο διαχωρισμός, διασπά τη συμμετρική δομή της συμπαγούς σύνθεσης του μοντέλου και δίνει τη δυνατότητα της επίλυσής του με τη μέθοδο δημιουργίας στήλης.

Το μοντέλο **(4-25)-(4-29)** έχει πολλούς περιορισμούς σακιδίου, έναν για κάθε κάδο k. Κάθε περιορισμός σακιδίου, ορίζει ένα φραγμένο σύνολο, που έχει μόνο ακραία σημεία και όχι ακραίες ακτίνες. Το σύνολο X είναι η τομή των σταθερών του σακιδίου για κάθε κάδο. Το σύνολο λύσεων του περιορισμού σακιδίου του κάδου k, μπορεί να περιοριστεί με την εξάλειψη έγκυρων κλασματικών λύσεων των περιορισμών σακιδίου που δεν αντιστοιχούν σε έγκυρα σχέδια κοπής για τον κάδο k. Επίσης, λαμβάνονται υπόψη μόνο οι ακέραιες λύσεις του περιορισμού σακιδίου για τον κάδο k. Χρησιμοποιώντας την προσέγγιση κυρτότητας, θα χρησιμοποιηθούν τα ακραία σημεία του $conv\ (X)$. Σύμφωνα με το θεώρημα Minkowski-Weyl:

$$x_{ik} = \sum_{p \in P} a_{ik}^{p} \lambda_{k}^{p} \qquad (4\text{-}38)$$

$$\sum_{p \in P} \lambda_{k}^{p} = 1 \qquad (4\text{-}39)$$

$$\lambda_{k}^{p} \geq 0 \qquad (4\text{-}40)$$

Η ακέραια μεταβλητή y_k για κάθε κύλινδρο k, ορίζεται συγκεντρωτικά ως [25],[89]:

$$y_k = \sum_{p \in P} \lambda_{k}^{p} \qquad (4\text{-}41)$$

Με την αντικατάσταση των **(4-38)-(4-41)** στις σχέσεις **(4-25)**, **(4-26)**, **(4-28)** και **(4-29)**, προκύπτει η διατύπωση του κύριου προβλήματος (με τη χρήση της προσέγγισης κυρτότητας) [89]:

Αντικειμενική συνάρτηση:

$$\min \sum_{k=1}^{K} \sum_{p \in P} \lambda_k^p \tag{4-42}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{k=1}^{K} \sum_{p \in P} a_{ik}^p \lambda_k^p = 1, i = 1,2, \dots, m \tag{4-43}$$

$$\sum_{p \in P} \lambda_k^p = 1, k = 1,2, \dots, K \tag{4-44}$$

$$\sum_{p \in P} a_{ik}^p \lambda_k^p \in \{0,1\}, i = 1,2, \dots, m, k = 1,2, \dots, K \tag{4-45}$$

$$\sum_{p \in P} \lambda_k^p \geq 0, k = 1,2, \dots, K \tag{4-46}$$

$$\lambda_k^p \geq 0, p \in P, k = 1,2, \dots, K \tag{4-47}$$

Εδώ, το πρώτο σύνολο περιορισμών **(4-43)** διασφαλίζει, την αντιστοίχιση όλων των αντικειμένων σε πρότυπα και οι περιορισμοί κυρτότητας εγγυώνται ότι τα πρότυπα βρίσκονται στην κυρτή θήκη (convex hull) των δυαδικών λύσεων του προβλήματος σακιδίου. Επιπλέον, ο περιορισμός ακεραιότητας απαιτεί ότι ο κυρτός συνδυασμός των ακραίων σημείων στην κυρτή θήκη, θα πρέπει να οδηγεί σε ένα δυαδικό πρότυπο.

Όπως και στο 1D-CSP, έτσι και εδώ, όταν όλοι οι κάδοι έχουν το ίδιο πλάτος, όλα τα υπο-προβλήματα είναι πανομοιότυπα και το μοντέλο **(4-42)-(4-47)** είναι ισοδύναμο με το ακόλουθο μοντέλο (σχεδόν παρόμοιο με το κλασικό μοντέλο των Gilmore-Gomory για το 1D-CSP) [96], [97]:

Αντικειμενική συνάρτηση:

$$\min \sum_{p \in P} \lambda_p \tag{4-48}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{p \in P} a_{ip} \lambda_p \geq 1, i = 1,2, \dots, m \tag{4-49}$$

$$\lambda_p \in \{0,1\}, \forall\, p \in P \tag{4-50}$$

Όπου, το πρότυπο p ορίζεται ως ένα δυαδικό διάνυσμα $A_p = \left(a_{1p}, \dots, a_{ip}, \dots, a_{mp}\right)^T$: $a_{ip} = 1$ αν το αντικείμενο i περιέχεται σε πρότυπο p, αλλιώς 0 και η μεταβλητή απόφασης λ_p είναι δυαδική $\lambda_p = 1$ αν το μοτίβο p χρησιμοποιείται στη λύση αλλιώς 0 [96], [97].

Ο αριθμός των εφικτών προτύπων είναι εκθετικός στον αριθμό των αντικειμένων, επομένως η απαρίθμηση όλων είναι απαγορευτική ακόμη και για περιπτώσεις μέτριου μεγέθους είναι απαγορευτική. Επομένως, το κύριο πρόβλημα επιλύεται με τη μέθοδο δημιουργίας στήλης: αρχικά διατυπώνεται το περιορισμένο πρόβλημα (RMP) που είναι το ακόλουθο [96], [97]:

Αντικειμενική συνάρτηση:

$$\min \sum_{p \in P'} \lambda_p \tag{4-51}$$

Υποκειμενική στους περιορισμούς:

$$\sum_{p \in P'} a_{ip} \lambda_p \geq 1, i = 1,2, \dots, m \tag{4-52}$$

$$\lambda_p \in \{0,1\}, \forall\, p \in P' \tag{4-53}$$

Μία εναλλακτική διατύπωση είναι η αντικατάσταση του «≥» με «=» [96], [97]. Αν π_i είναι οι δυϊκές μεταβλητές που αντιστοιχούν στους περιορισμούς **(4-52)**, το υπο-πρόβλημα είναι το ακόλουθο πρόβλημα σακιδίου [96], [97]:

Αντικειμενική συνάρτηση:

$$\bar{c}_p = \max \sum_{i=1}^{m} \pi_i a_i \qquad (4\text{-}54)$$

Υποκειμενική στους περιορισμούς:

$$\sum_{i=1}^{n} a_i w_i \leq c, i = 1,2, \dots, m \qquad (4\text{-}55)$$

$$a_i \in \{0,1\} \qquad (4\text{-}56)$$

Το πρόβλημα σακιδίου **(4-54)-(4-56)**, αντιστοιχεί στην εύρεση της στήλης με το ελάχιστο μειωμένο κόστος $\bar{c}_p = 1 - \pi_i A_p$.

Η επίλυση με τη μέθοδο δημιουργίας στήλης περιλαμβάνει τα ακόλουθα βήματα:

Βήμα 1: Επιλογή ενός αρχικού συνόλου προτύπων (patterns).

Βήμα 2: Επίλυση του RMP **(4-51)-(4-53)**.

Βήμα 3: Με τη χρήση των δυϊκών τιμών π_i που αντιστοιχούν στους περιορισμούς **(4-49)** του RMP, πρέπει να επιλυθεί το υπο-πρόβλημα **(4-54)-(4-56)** που είναι πρόβλημα σακιδίου. Η επίλυση του προβλήματος σακιδίου μπορεί να γίνει με διάφορους τρόπους. Αν $1 - \bar{c}_p < 0$, προστίθεται μία στήλη (που είναι ένα καινούριο πρότυπο, new pattern) στο περιορισμένο κύριο πρόβλημα, το οποίο βελτιστοποιείται εκ νέου (μετάβαση στο **Βήμα 2**), διαφορετικά (αν $1 - \bar{c}_p \geq 0$ η λύση είναι βέλτιστη.

$$
\begin{array}{llllll}
min & \sum_{p \in P'} 1\,\lambda_p + & \boxed{1}\,\lambda_{p^*} + & \boxed{1}\,\lambda_{p^{**}} + & \cdots & \\
s.t. & \sum_{p \in P'} a_{1p}\lambda_p + & a_{1p*}\,\lambda_{p^*} + & a_{1p**}\,\lambda_{p^{**}} + & \cdots & = 1 \\
 & \sum_{p \in P'} a_{2p}\lambda_p + & a_{2p*}\,\lambda_{p^*} + & a_{2p**}\,\lambda_{p^{**}} + & \cdots & = 1 \\
 & \vdots & \vdots & \vdots & \vdots & \vdots \\
 & \sum_{p \in P'} a_{mp}\lambda_p + & a_{mp*}\,\lambda_{p^*} + & a_{mp**}\,\lambda_{p^{**}} + & \cdots & = 1 \\
 & \lambda_p, & \lambda_{p^*}, & \lambda_{p^{**}}, & \cdots & \in \{0,1\} \quad \forall\, p \in P'
\end{array}
$$

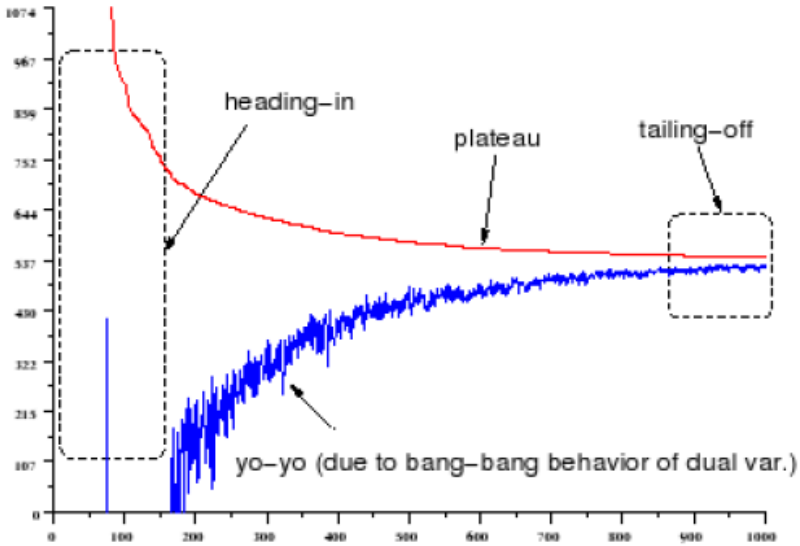
Εικόνα 4-7: Προσθήκη στηλών στο περιορισμένο κύριο πρόβλημα του μονοδιάστατου προβλήματος κοπής αποθέματος. Όπου, λ_{p^*} είναι η βέλτιστη τιμή της μεταβλητής λ_p στην πρώτη επανάληψη, $\lambda_{p^{**}}$ είναι η βέλτιστη τιμή της μεταβλητής λ_p στην δεύτερη επανάληψη. Οι συντελεστές a_{ip} λαμβάνονται από την επίλυση του προβλήματος σακιδίου.

Κεφάλαιο 5. Βελτίωση της απόδοσης της μεθόδου μαθηματικής αποσύνθεσης Dantzig-Wolfe

5.1 Εισαγωγή

Όπως είναι γνωστό, η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe χρησιμοποιεί την μέθοδο δημιουργίας στήλης για την επίλυση του περιορισμένου κύριου προβλήματος (RMP). Τις τελευταίες δεκαετίες, η μέθοδος δημιουργίας στήλης έχει αποκτήσει σημαντική δημοτικότητα για την επίλυση διαφόρων πρακτικών προβλημάτων απόφασης [68]. Αν και χρησιμοποιήθηκε με επιτυχία, «υποφέρει» κυρίως από αργή σύγκλιση που περιορίζει την αποτελεσματικότητά της. Για το λόγο αυτό, στη βιβλιογραφία η μέθοδος δημιουργίας στήλης αναφέρεται και ως μέθοδος **αργής δημιουργίας στήλης** (delayed column generation) [49], [62]. Για το λόγο αυτό, η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe μπορεί να επιταχυνθεί, επεμβαίνοντας στη μέθοδο δημιουργίας στήλης.

Ωστόσο, η αργή σύγκλιση δεν είναι το μοναδικό μειονέκτημα. Η διαδικασία δημιουργίας στήλης «πάσχει» από διάφορα μειονεκτήματα που απεικονίζονται στην ***Εικόνα 5-1***. Το πρώτο είναι το «heading-in effect»: στην αρχή της διαδικασίας, «φτωχές» δυϊκές πληροφορίες οδηγούν σε «όχι και τόσο καλά» δυϊκά όρια, οπότε το χάσμα μεταξύ αρχικού και δυϊκού ορίου είναι σημαντικό. Επιπλέον, η αργή σύγκλιση, που αναφέρθηκε προηγουμένως, συνήθως ονομάζεται «tailing-off effect»: πλησιάζοντας τη βέλτιστη λύση απαιτούνται πολλές επαναλήψεις για να αποδειχθεί η βελτιστότητα. Υπάρχουν επίσης προβλήματα εκφυλισμού (degeneracy) στο αρχικό πρόβλημα: οι νέες στήλες που προστίθενται στο περιορισμένο κύριο πρόβλημα, δεν βελτιώνουν την τιμή του, που παραμένει σταθερή κατά τη διάρκεια πολλών επαναλήψεων (plateau effect) [99]. Η αλματώδης συμπεριφορά (jumpy behavior) αντιστοιχεί σε ταλαντώσεις των ενδιάμεσων δυϊκών ορίων (όρια Lagrange), που δεν συγκλίνουν μονοτονικά στη βέλτιστη τιμή. Αυτό το φαινόμενο οφείλεται στην αστάθεια των τιμών των δυϊκών μεταβλητών από μία επανάληψη στην επόμενη: οι δυϊκές μεταβλητές, οι οποίες οδηγούν τη διαδικασία παραγωγής στηλών (υπο-πρόβλημα), ενδέχεται να ταλαντωθούν σε μεγάλο βαθμό προτού τελικά συγκλίνουν σε ορισμένες βέλτιστες δυϊκές τιμές [99].



Εικόνα 5-1: Μειονεκτήματα της μεθόδου δημιουργίας στήλης [99]*. Ο κατακόρυφος άξονας αντιπροσωπεύει τις τιμές του περιορισμένου κύριου προβλήματος, ο οριζόντιος άξονας αντιπροσωπεύει τον αριθμό των επαναλήψεων, η κόκκινή καμπύλη αντιπροσωπεύει τη μεταβολή της τιμής του περιορισμένου κύριου προβλήματος και τέλος, η μπλε καμπύλη αντιπροσωπεύει μεταβολή της τιμής του δυϊκού ορίου.*

Για την μελέτη και την αντιμετώπιση των μειονεκτημάτων, έχει καταβληθεί μεγάλη προσπάθεια. Στη βιβλιογραφία προτείνονται διάφορες προσεγγίσεις. Κάποιες από αυτές, είναι γνωστές ως μέθοδοι σταθεροποίησης, γιατί προσπαθούν να μειώσουν την ασταθή συμπεριφορά των δυϊκών μεταβλητών. Άλλες βασίζονται στην ιδέα της εντατικοποίησης (δημιουργία πολλαπλών στηλών), προσθέτοντας ένα σύνολο στηλών σε κάθε επανάληψη.

Στο παρόν Κεφάλαιο, περιγράφονται κάποιες από τις μεθόδους σταθεροποίησης, η ιδέα της εντατικοποίησης και στη συνέχεια, αναπτύσσεται η μέθοδος δημιουργίας πολλαπλών στηλών με εξομάλυνση δυϊκών τιμών, που προτείνεται ως μία νέα μέθοδος επιτάχυνσης του αλγορίθμου δημιουργίας στήλης (4^{ος} στόχος της εργασίας).

5.2 Τεχνικές σταθεροποίησης της μεθόδου δημιουργίας στήλης

Στην βιβλιογραφία, υπάρχουν διάφορες τεχνικές σταθεροποίησης της κλασικής μεθόδου δημιουργίας στήλης, όπως μέθοδοι δεσμού (bundle stabilization methods) [35], πολυεδρικοί μέθοδοι σταθεροποίησης (polyhedral stabilization methods) [34], [100], μέθοδοι

σταθεροποίησης κέντρου (center stabilization methods), μέθοδοι εσωτερικού σημείου [85] και μέθοδοι εξομάλυνσης [40], [41], που λειτουργούν στον δυϊκό χώρο. Στόχος τους είναι η σταθεροποίηση της ασταθούς συμπεριφοράς των δυϊκών μεταβλητών [36], [40], [41]. Άλλες τεχνικές είναι υβριδικές μέθοδοι της μεθόδου δημιουργίας στήλης και άλλων μεθόδων βελτιστοποίησης, όπως η μέθοδος υπο-βαθμίδας (subgradient method) [42], [101]. Τέλος, μία από τις πιο ελπιδοφόρες προσεγγίσεις σταθεροποιημένης μεθόδου δημιουργίας στήλης έχει προταθεί από τον V. Carvalho [43]. Η μέθοδος προσθέτει έναν πολυωνυμικό αριθμό δυϊκών τομών (dual cuts) στο περιορισμένο κύριο πρόβλημα. Οι δυϊκές τομές μπορούν να θεωρηθούν ως περιορισμοί που αποφεύγουν τις ταλαντώσεις των διπλών μεταβλητών.

Από όλα αυτά τα είδη των τεχνικών σταθεροποίησης, στις ακόλουθες παραγράφους, περιγράφονται κάποιες μέθοδοι ποινής και οι μέθοδοι εξομάλυνσης.

5.2.1 Μέθοδος Boxstep

Η μέθοδος Boxstep (Boxstep Method) χρησιμοποιείται για την φραγή των δυϊκών μεταβλητών γύρω από ένα κέντρο σταθερότητας $\hat{\pi}$ και εισήχθη από τους R. E. Marsten, W.W. Hogan και J. W. Blankenship [102]. Με την επιβολή κατώτερων και άνω ορίων, οι δυϊκές μεταβλητές περιορίζονται σε ένα πλαίσιο (Box) γύρω από την προηγούμενη δυϊκή λύση. Το περιορισμένο κύριο πρόβλημα (RMP) βελτιστοποιείται εκ νέου. Αν η νέα βέλτιστη δυϊκή λύση επιτευχθεί στο όριο του πλαισίου, προκύπτει μια κατεύθυνση προς την οποία θα πρέπει να μεταφερθεί το πλαίσιο. Διαφορετικά, η βέλτιστη δυϊκή λύση επιτυγχάνεται στο εσωτερικό αυτού του πλαισίου, παράγοντας την επιδιωκόμενη βέλτιστη λύση. Αυτή είναι η αρχή της μεθόδου Boxtep και η βασική ιδέα της χρήσης ενός **κέντρου σταθερότητας**, δηλαδή, η τρέχουσα καλύτερη εικασία για μία βέλτιστη δυϊκή λύση που είναι κατά μία έννοια πιο «αξιόπιστη» από τις υπόλοιπες δυϊκές λύσεις. Αυτό είναι γνωστό, π.χ. στις μεθόδους αξιοπιστίας-περιοχής και βασικός μηχανισμός όλων όσων ακολουθούν.

Έστω το ακόλουθο αρχικό γραμμικό πρόβλημα:

Αντικειμενική συνάρτηση:

$$\min cx \tag{5-1}$$

Υποκειμενική στους περιορισμούς:

$$Ax = b \tag{5-2}$$

$$x \geq 0 \tag{5-3}$$

Αν είναι π η δυϊκή μεταβλητή που αντιστοιχεί στον περιορισμό **(5-2)**, τότε το αντίστοιχο δυϊκό πρόβλημα είναι το ακόλουθο:

Αντικειμενική συνάρτηση:

$$\max \pi b \tag{5-4}$$

Υποκειμενική στους περιορισμούς:

$$\pi A \leq c \tag{5-5}$$

Το τροποποιημένο κύριο πρόβλημα της μεθόδου δημιουργίας στήλης, σύμφωνα με τη μέθοδο Boxstep, είναι το ακόλουθο, όπου εισάγονται χαλαρές και πλεονασματικές μεταβλητές:

Αντικειμενική συνάρτηση:

$$\min cx - \delta_{-}u_{-} + \delta_{+}u_{+} \tag{5-6}$$

Υποκειμενική στους περιορισμούς:

$$Ax - u_{-} + u_{+} = b \tag{5-7}$$

$$x, u_{-}, u_{+} \geq 0 \tag{5-8}$$

Αν π είναι η δυϊκή μεταβλητή που αντιστοιχεί στον περιορισμό **(5-7)**, τότε το αντίστοιχο δυϊκό πρόβλημα είναι το ακόλουθο:

Αντικειμενική συνάρτηση:

$$\max \pi b \tag{5-9}$$

Υποκειμενική στους περιορισμούς:

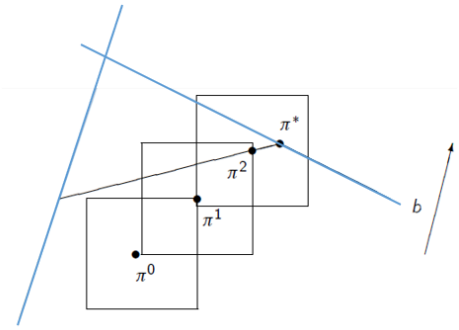
$$\pi A \leq c \tag{5-10}$$

$$-\pi \leq -\delta_{-} \tag{5-11}$$

$$\pi \leq \delta_{+} \tag{5-12}$$

Οι δυϊκοί περιορισμοί **(5-11)** και **(5-12)** αν συγχωνευτούν, ορίζουν ένα πλαίσιο (Box):

$$\delta_{-} \leq \pi \leq \delta_{+} \tag{5-13}$$



Εικόνα 5-2: Οπτικοποίηση της μεθόδου Boxstep. Όπου π^1, π^2 είναι οι νέες βέλτιστες δυϊκες λύσεις που επιτυγχάνονται στα όρια του πλαισίου και ορίζουν την κατεύθυνση στην οποία θα μεταφερθεί το πλαίσιο και π^* είναι η βέλτιστη δυϊκή λύση που επιτυγχάνεται στο εσωτερικό του πλαισίου [17].

5.2.2 Πολυεδρικοί όροι ποινής

Το 1999, οι Du Merle et al. [34], πρότειναν ως μία τεχνική βελτίωσης της σύγκλισης της κλασικής μεθόδου δημιουργίας στήλης, τη δυϊκή σταθεροποίηση. Η τεχνική αυτή, σταθεροποιεί τις δυϊκές μεταβλητές του περιορισμένου κύριου προβλήματος (Polyhedral Penalty Terms). Στην κλασική μέθοδο δημιουργίας στήλης, οι βέλτιστες λύσεις δημιουργούνται στις τελευταίες επαναλήψεις, όπου οι τιμές των δυϊκών μεταβλητών έχουν σχεδόν συγκλίνει. Επιπλέον, αυτές οι δυϊκές μεταβλητές τείνουν να ταλαντώνονται πολύ στις αρχικές επαναλήψεις. Η ιδέα είναι η δυνατότητα αρχικοποίησης και σταθεροποίησης των τιμών, των δυϊκών μεταβλητών κοντά στις βέλτιστες τιμές τους. Το γεγονός αυτό, θα μπορούσε να μειώσει τις επαναλήψεις και ενδεχομένως την αποφυγή δημιουργίας πολλών στηλών που δεν θα

χρησιμοποιηθούν ποτέ για την εύρεση της τελικής βέλτιστης λύσης [36]. Παρακάτω, αναπτύσσεται μια απλοποιημένη έκδοση της δυϊκής σταθεροποίησης της κλασικής μεθόδου δημιουργίας στήλης.

Ας θεωρηθεί το πρόβλημα **(5-1)-(5-3)** και το δυϊκό του **(5-4)-(5-5)**. Το σταθεροποιημένο μοντέλο της μεθόδου δημιουργίας στήλης προκύπτει με πρόσθεση των χαλαρών και των πλεονασματικών μεταβλητών:

Αντικειμενική συνάρτηση:

$$\min cx - \delta_- u_- + \delta_+ u_+ \tag{5-14}$$

Υποκειμενική στους περιορισμούς:

$$Ax - u_- + u_+ = b \tag{5-15}$$

$$u_- \leq \varepsilon_- \tag{5-16}$$

$$u_+ \leq \varepsilon_+ \tag{5-17}$$

$$x, y_-, y_+ \geq 0 \tag{5-18}$$

Το αντίστοιχο δυϊκό πρόβλημα είναι το ακόλουθο:

Αντικειμενική συνάρτηση:

$$\max \pi b - \varepsilon_- w_- - \varepsilon_+ w_+ \tag{5-19}$$

Υποκειμενική στους περιορισμούς:

$$\pi A \leq c \tag{5-20}$$

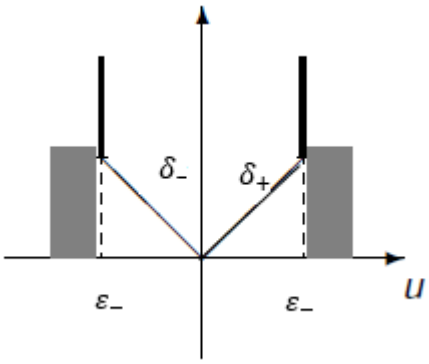
$$-\pi - w_- \leq -\delta_- \tag{5-21}$$

$$\pi - w_+ \leq \delta_+ \tag{5-22}$$

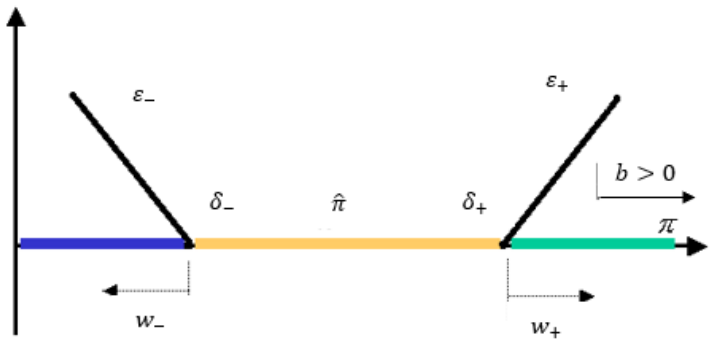
$$w_-, w_+ \geq 0 \qquad (5\text{-}23)$$

Όπου, u_+ και u_- είναι οι μεταβλητές πλεονασμού και οι χαλαρές μεταβλητές αντίστοιχα, διαταράσσουν το b με ένα $\varepsilon \in [-\varepsilon_-, \varepsilon_+]$, που συμβάλλει στη μείωση του εκφυλισμού. Η ερμηνεία του δυϊκού προβλήματος **(5-19)-(5-23)**, είναι πιο ενδιαφέρουσα. Η δυϊκή μεταβλητή π είναι περιορισμένη στο διάστημα $[\delta_- - w_-, \delta_+ + w_+]$, δηλαδή απόκλιση της δυϊκής μεταβλητή π από το διάστημα $[\delta_-, \delta_+]$ επιτρέπεται αλλά τιμωρείται κατά ένα ποσό $\varepsilon_-, \varepsilon_+$ανά μονάδα, αντίστοιχα. Από το αρχικό πρόβλημα **(5-14)-(5-18)** λαμβάνεται μία βέλτιστη λύση του αδιατάρακτου προβλήματος $\min\{cx | Ax = b, x \geq 0\}$ όταν $\varepsilon_- = \varepsilon_+ = 0$ ή $\delta_- < \hat{\pi} < \delta_+$, όπου $\hat{\pi}$ είναι η βέλτιστη δυϊκή λύση του δυϊκού προβλήματος **(5-19)-(5-23)**. Το κριτήριο τερματισμού του αλγορίθμου δημιουργίας στήλης γίνεται: η τιμή του υπο-προβλήματος είναι ίση με 0 και $u_- = u_+ = 0$.

Αυτή η προσέγγιση μπορεί να χρειάζεται κάποια ρύθμιση παραμέτρων, αλλά για ορισμένα προβλήματα επιταχύνει σημαντικά τον αλγόριθμο δημιουργίας στήλης. Η αλλαγή στο RMP απαιτεί την προσθήκη μόνο τεχνικών μεταβλητών με άνω όριο, που δεν αυξάνει το μέγεθος της βάσης. Μπορεί εύκολα να γενικευτεί σε τμηματικά γραμμικές συναρτήσεις ποινής με περισσότερα τμήματα, όπου αυτά φαίνεται να δίνουν έναν καλό συμβιβασμό με ισχυρότερη ποινή πιο μακριά από το κέντρο σταθερότητας [**Εικόνα 5-5 (β)**, **(γ)**, **(δ)**] [36]. Τέλος, σημειώνεται ότι το αρχικό πρόβλημα **(5-14)-(5-18)** είναι χαλάρωση του μη διαταραγμένου RMP, και μπορεί να υπολογιστεί ταχύτερα [36].



Εικόνα 5-3: Ποινή σύμφωνα με Du Merle et al [17].



Εικόνα 5-4: Περιοχή εμπιστοσύνης σύμφωνα με Du Merle et al [103].

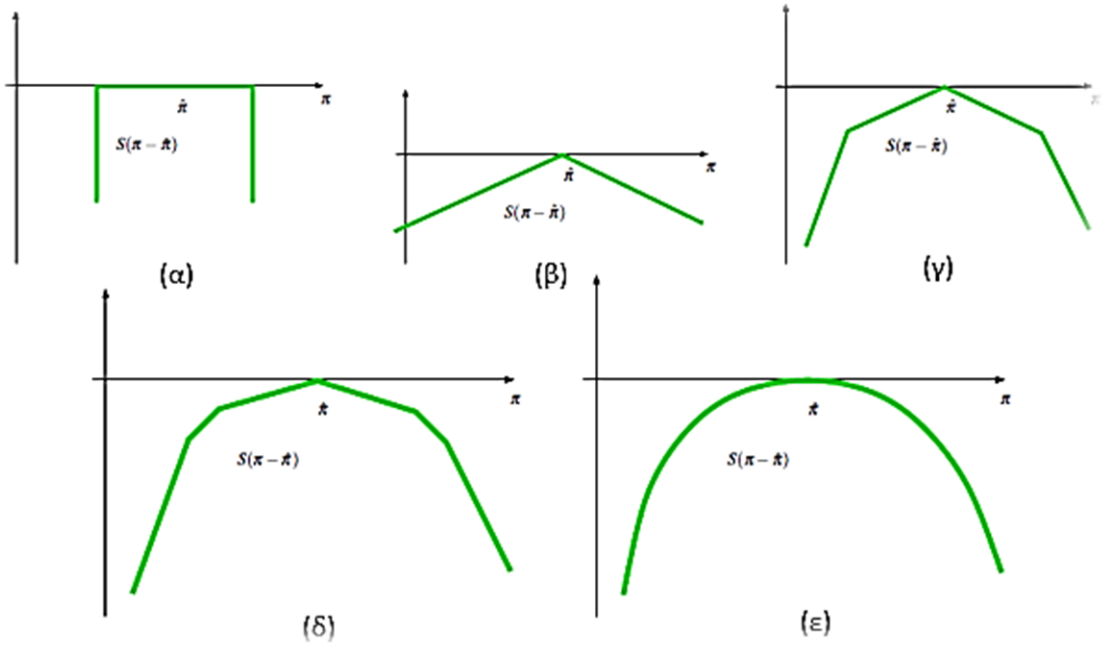
5.2.3 Μέθοδοι δέσμης (Τετραγωνικοί όροι ποινής)

Οι μέθοδοι δέσμης (Bundle Methods) αποτελούν μία ειδική κατηγορία μεθόδων ποινής που ονομάζονται μέθοδοι τετραγωνικής ποινής (Quadratic Penalty). Για την εφαρμογή αυτών των μεθόδων, στην αντικειμενική συνάρτηση του δυϊκού περιορισμένου κύριου προβλήματος, προστίθεται μια συνάρτηση ποινής για να οδηγήσει τη βελτιστοποίηση σε δυϊκές λύσεις που βρίσκονται κοντά σε ένα κέντρο σταθερότητας, που συνήθως ορίζεται ως η υπάρχουσα δυϊκή λύση $\hat{\pi}$.

Η συνάρτηση ποινής είναι ένας τετραγωνικός όρος (μια ευκλείδεια νόρμα) που «τιμωρεί» την απόκλιση των δυϊκών μεταβλητών από το κέντρο σταθερότητας $\hat{\pi}$. Η συνάρτηση ποινής αποκτά την ακόλουθη μορφή [35]:

$$\hat{S}(\pi) = \frac{1}{2t}\|\pi - \hat{\pi}\|^2 \qquad (5\text{-}24)$$

Όπου t είναι μία θετική παράμετρος που προσαρμόζεται δυναμικά για να βοηθήσει στη σύγκλιση. Η συνάρτηση ποινής $\hat{S}$, είναι τυπικά κυρτή, η τιμή της είναι 0 στο κέντρο σταθερότητας $\hat{\pi}$ και αυξάνεται καθώς η $||\pi - \hat{\pi}||$ αυξάνεται. Ως συνάρτηση ποινής μπορεί να θεωρηθεί και μία τμηματική γραμμική συνάρτηση ποινής S, προκειμένου να διασφαλιστεί ότι το κύριο πρόβλημα εξακολουθεί να είναι ένα γραμμικό πρόγραμμα.



Εικόνα 5-5: Παραδείγματα συναρτήσεων ποινής: (α) Boxstep, (β), (γ), (δ) τρεις γραμμικές τμηματικές συναρτήσεις ποινής και (ε) η τετραγωνική ποινή της μεθόδου δέσμης (Bundle Method). Όπου S είναι μία συνάρτηση ποινής που αυξάνεται καθώς η δυϊκή μεταβλητή π απομακρύνεται από το κέντρο σταθερότητας $\hat{\pi}$.

5.2.4 Κυρτοί συνδυασμοί με προηγούμενες δυϊκές λύσεις

Μια διαφορετική προσέγγιση για την σταθεροποίηση των δυϊκών τιμών, όπου δεν απαιτείται καμία τροποποίηση του RMP, αλλά η ρύθμιση μίας μόνο παραμέτρου, είναι οι μέθοδοι εξομάλυνσης (smoothing methods) των δυϊκών μεταβλητών με κυρτούς συνδυασμούς της τρέχουσας δυϊκής λύσης π_k με μια οποιαδήποτε προηγούμενη $\hat{\pi}$ (convex combination with previous dual solutions) [36]. Όταν βρίσκεται μια στήλη, προστίθεται στο RMP μόνο όταν έχει αρνητικό μειωμένο κόστος σε σχέση με την τρέχουσα δυϊκή λύση π_k. Έτσι, μετά από κάθε λύση του περιορισμένου κύριου προβλήματος (RMP), οι δυϊκές τιμές, πριν σταλούν στο υπο-πρόβλημα, έχουν εξομαλυνθεί. Έτσι, το υπο-πρόβλημα επιλύεται με την σταθεροποιημένη τιμή $\bar{\pi}_k$, που υπολογίζεται από την ακόλουθη σχέση (μέθοδος Neame) [41]:

$$\bar{\pi}_k = \alpha\hat{\pi} + (1-\alpha)\pi_k \qquad (5\text{-}25)$$

Όπου, $\alpha \in [0,1]$ είναι μία σταθερά που παραμετροποιεί το επίπεδο εξομάλυνσης και είναι αυτή που ρυθμίζεται. Από αυτή τη στατική επιλογή της παραμέτρου α, το κέντρο σταθερότητας $\hat{\pi}$ κινείται με λιγότερη ευελιξία σε σχέση με τις προηγούμενες προτάσεις.

Θα πρέπει να σημειωθούν τα ακόλουθα:

- Στην ειδική περίπτωση, όπου $\alpha = 1$, η μέθοδος ταυτίζεται με την κλασική μέθοδο δημιουργίας στήλης.
- Αυτή η μέθοδος δεν μπορεί να εφαρμοστεί στην πρώτη επανάληψη γιατί υπάρχει μόνο μία δυϊκή λύση.
- Η επίλυση του τροποποιημένου προβλήματος τιμολόγησης (υπο-πρόβλημα) ενδέχεται να μην αποφέρει αρνητική στήλη μειωμένου κόστους, ακόμη και όταν υπάρχει για π_k. Αυτή η κατάσταση είναι αποτέλεσμα λανθασμένης τιμολόγησης (mis-pricing). Στην περίπτωση αυτή, εφαρμόζοντας την σχέση (5-25) με την ίδια τιμή π_k, η λύση οδηγεί σε μία νέα δυϊκή τιμή που είναι πιο κοντά στο κέντρο σταθερότητας $\hat{\pi}$.

5.3 Η μέθοδος δημιουργίας πολλαπλών στηλών

Εκτός από τις μεθόδους σταθεροποίησης, ένας άλλος τρόπος που συνήθως οδηγεί στη μείωση του αριθμού των επαναλήψεων της κλασικής μεθόδου δημιουργίας στήλης, για την επίτευξη της βέλτιστης λύσης, είναι η προσθήκη ενός συνόλου στηλών που αντιστοιχούν σε λύσεις με αρνητικό μειωμένο κόστος (περίπτωση προβλήματος ελαχιστοποίησης), συμπεριλαμβανομένων επίσης μη βέλτιστων λύσεων στο RMP [44]. Αυτή η μέθοδος είναι γνωστή ως μέθοδος δημιουργίας πολλαπλών στηλών ή εντατικοποιημένη μέθοδος δημιουργίας στήλης (Intensified Column Generation, ICG). Σε κάθε επανάληψη, στο RMP αντί για μία, προστίθενται αρκετές διαφορετικές στήλες που βρέθηκαν μέσω του υπο-προβλήματος.

Για την παραγωγή k στηλών από το υπο-πρόβλημα, το οποίο για τα 1D-CSP & 1D-BPP, είναι ένα πρόβλημα σακιδίου, μπορεί να χρησιμοποιηθεί ο αλγόριθμος που περιγράφεται στις αναφορές [104], [105], ο οποίος βρίσκει τις k-καλύτερες λύσεις (k-best solutions): 1η, 2η,..., k του προβλήματος σακιδίου. Στη συνέχεια αυτές προστίθενται στο περιορισμένο κύριο πρόβλημα.

Στην διπλωματική εργασία, δίνεται μία άλλη προσέγγιση της μεθόδου δημιουργίας πολλαπλών στηλών: εξετάζεται η μέθοδος δημιουργίας πολλαπλών στηλών όπου προστίθενται $n = k + 1$ στήλες στο RMP σε κάθε επανάληψη, όπου «1» είναι η μία στήλη που

προέρχεται από το υπο-πρόβλημα της κλασικής μεθόδου δημιουργίας στήλης και k είναι οι υπόλοιπες στήλες που προέρχονται από τροποποιημένο/-α υπο-πρόβλημα/-τα, των οποίων οι δυϊκές μεταβλητές της αντικειμενικής συνάρτησης σταθεροποιούνται μέσω της σχέσης Neame **(5-25)** (κυρτοί συνδυασμοί της τρέχουσας δυϊκής λύσης με μια οποιαδήποτε προηγούμενη). Στο εξής αυτή η μέθοδος, θα αναφέρεται ως προτεινόμενη μέθοδος.

5.3.1.1 Προσθήκη $n = 2$ στηλών

Η ιδέα είναι η ακόλουθη: σε κάθε επανάληψη, μπορούν να προστεθούν δύο στήλες: α) η στήλη που παράγεται με την τρέχουσα δυϊκή λύση (δηλαδή από την κλασική μέθοδο δημιουργίας στήλης, η οποία εξασφαλίζει και τη σύγκλιση) και β) μία επιπλέον στήλη που προκύπτει από την εξομάλυνση των δυϊκών τιμών μέσω της σχέσης Neame **(5-25)**, από ένα δεύτερο τροποποιημένο υπο-πρόβλημα.

<u>Σημείωση:</u> παρακάτω, θα χρησιμοποιηθούν δύο όροι: ο όρος «κλασικό υπο-πρόβλημα», αναφέρεται στο υπο-πρόβλημα του οποίου οι δυϊκές τιμές είναι εκείνες που αντιστοιχούν στην κλασική μέθοδο δημιουργίας στήλης. Ο όρος «τροποποιημένο υπο-πρόβλημα», αναφέρεται στο υπο-πρόβλημα του οποίου οι δυϊκές τιμές έχουν εξομαλυνθεί σύμφωνα με τη σχέση **(5-25)** (κυρτοί συνδυασμοί της τρέχουσας δυϊκής λύσης με μια οποιαδήποτε προηγούμενη).

Τα βήματα του αλγορίθμου είναι τα παρακάτω:

Βήμα 0: Επιλογή μίας τιμής για την παράμετρο εξομάλυνσης $a \in (0,1)$.

Βήμα 1α: Επίλυση του RMP μόνο στην πρώτη επανάληψη με την κλασική μέθοδο δημιουργίας στήλης. Αν $\bar{c}^* \geq 0$, τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται. Αλλιώς αν $\bar{c}^* < 0$, από τη δεύτερη επανάληψη:

Βήμα 1β: Επίλυση του νέου RMP.

Βήμα 2: Επίλυση του κλασικού και του τροποποιημένου υπο-προβλήματος.

Βήμα 3: Αν $\bar{c}^* \geq 0$ (του κλασικού υπο-προβλήματος), τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται. Αλλιώς αν $\bar{c}^* < 0$, στο RMP προστίθενται δύο στήλες (όπου η μία προέρχεται από το κλασικό υπο-πρόβλημα και η επιπλέον $k = 1$ προέρχεται από το τροποποιημένο υπο-πρόβλημα). Ο αλγόριθμος επαναλαμβάνεται από το **Βήμα 1β**.

Παρατηρήσεις:

- Ο παραπάνω αλγόριθμος αποτελεί γενίκευση της κλασικής μεθόδου δημιουργίας στήλης, για $a = 1$. Ωστόσο, χρησιμοποιήθηκε $a \in (0,1)$ για να αποφευχθεί αυτή η ειδική περίπτωση.
- Ο συμβολισμός $\overline{c^*}$ δηλώνει το μειωμένο κόστος του κλασικού υπο-προβλήματος.

5.3.1.2 Αλγόριθμος δημιουργίας πολλαπλών στηλών

Ο παραπάνω αλγόριθμος, μπορεί να επεκταθεί για την προσθήκη περισσότερων από $n = 2$ στηλών στο RMP. Αυτό γίνεται με την δημιουργία επιπλέον $k \geq 2$ στηλών από το τροποποιημένο υπο-πρόβλημα, όπου η παράμετρος k εκφράζει τον μέγιστο αριθμό επιπλέον δημιουργούμενων στηλών από το τροποποιημένο υπο-πρόβλημα. Τα βήματα του αλγορίθμου δημιουργίας πολλαπλών στηλών είναι τα παρακάτω:

Βήμα 0: α) Επιλογή μίας τιμής για την παράμετρο εξομάλυνσης $a \in (0,1)$.
β) Επιλογή μίας τιμής για την παράμετρο $k \geq 2$.

Βήμα 1α: Επίλυση του RMP μόνο στην πρώτη επανάληψη με την κλασική μέθοδο δημιουργίας στήλης. Αν $\overline{c^*} \geq 0$, τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται. Αλλιώς αν $\overline{c^*} < 0$, από τη δεύτερη επανάληψη:

Βήμα 1β: Επίλυση του νέου RMP.

Βήμα 2: Επίλυση του κλασικού και του τροποποιημένου υπο-προβλήματος.

Βήμα 3: Αν $\overline{c^*} \geq 0$ (του κλασικού υπο-προβλήματος), τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται. Αλλιώς αν $\overline{c^*} < 0$, στο RMP προστίθενται $n = k + 1$ στήλες (όπου η μία προέρχεται από το κλασικό υπο-πρόβλημα και η επιπλέον $k \geq 2$, προέρχονται από το τροποποιημένο υπο-πρόβλημα). Ο αλγόριθμος επαναλαμβάνεται από το **Βήμα 1β**.

5.3.1.3 Γενικευμένος αλγόριθμος δημιουργίας στήλης

Με βάση τα παραπάνω, ρυθμίζοντας το εύρος των διαστημάτων απ' όπου παίρνουν τιμές οι παράμετροι a και k, μπορεί να διατυπωθεί και ένας γενικευμένος αλγόριθμος δημιουργίας στήλης, ο οποίος συμπυκνώνει όλες τις περιπτώσεις, π.χ. για $a = 1$ και $k = 0$, προκύπτει η κλασική μέθοδος δημιουργίας στήλης. Τα βήματα είναι τα ακόλουθα:

Βήμα 0: α) Επιλογή μίας τιμής για την παράμετρο εξομάλυνσης $a \in [0,1]$.
β) Επιλογή μίας τιμής για την παράμετρο $k \geq 0$.

Βήμα 1α: Στην περίπτωση όπου $a = 1$ και $k = 0$, ακολουθούνται τα βήματα του κλασικού αλγόριθμου δημιουργίας στήλης, αγνοώντας το τροποποιημένο υπο-πρόβλημα. Σε διαφορετικές τιμές: Επίλυση του RMP μόνο στην πρώτη επανάληψη με την κλασική μέθοδο δημιουργίας στήλης. Αν $\bar{c}^* \geq 0$, τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται. Αλλιώς αν $\bar{c}^* < 0$, από τη δεύτερη επανάληψη:

Βήμα 1β: Επίλυση του νέου RMP.

Βήμα 2: Επίλυση του κλασικού και του τροποποιημένου υπο-προβλήματος.

Βήμα 3: Αν $\bar{c}^* \geq 0$ (του κλασικού υπο-προβλήματος), τότε η λύση είναι βέλτιστη και ο αλγόριθμος τερματίζεται. Αλλιώς αν $\bar{c}^* < 0$, στο RMP προστίθενται $n = k + 1$ στήλες (όπου η μία προέρχεται από το κλασικό υπο-πρόβλημα και η επιπλέον $k \geq 1$, προέρχονται από το τροποποιημένο υπο-πρόβλημα). Ο αλγόριθμος επαναλαμβάνεται από το **Βήμα 1β**.

Π.χ. για $k = 3$, σε κάθε επανάληψη θα προστίθενται στο RMP $n = 3 + 1 = 4$ στήλες, όπου η μία στήλη προέρχεται από το κλασικό και οι 3 επιπλέον στήλες προέρχονται από το τροποποιημένο υπο-πρόβλημα.

Κεφάλαιο 6. Υπολογιστικά πειράματα

6.1 Εισαγωγή

Στο Κεφάλαιο αυτό, περιγράφεται η υπολογιστική μέθοδος που ακολουθήθηκε για την υλοποίηση αριθμητικών πειραμάτων που αφορούν την σύγκριση της κλασικής μεθόδου δημιουργίας στήλης με την μέθοδο δημιουργίας πολλαπλών στηλών που προτείνεται. Πιο συγκεκριμένα, περιγράφονται τα ακόλουθα:

- Σχολιασμοί για την εκτέλεση των υπολογιστικών πειραμάτων.
- Η βιβλιοθήκη BPPLIB για το μονοδιάστατο πρόβλημα κοπής αποθέματος και για το μονοδιάστατο πρόβλημα συσκευασίας κάδου.
- Τα διαθέσιμα προβλήματα δοκιμής (test instances) που παρέχονται από τη βιβλιοθήκη BPPLIB και η αντίστοιχη βιβλιογραφία όπου αυτά εμφανίζονται.
- Το λογισμικό που χρησιμοποιήθηκε για την εκτέλεση υπολογιστικών πειραμάτων.

6.2 Σημειώσεις που αφορούν την εκτέλεση των υπολογιστικών πειραμάτων

Για την εκτέλεση των υπολογιστικών πειραμάτων, σημειώνονται τα ακόλουθα:

- Για την επίλυση του 1D-CSP, θεωρήθηκε ότι όλοι οι κύλινδροι έχουν το ίδιο πλάτος, όλα επομένως τα υπο-προβλήματα είναι πανομοιότυπα και μπορεί να χρησιμοποιηθεί το απλοποιημένο μοντέλο των Gilmore-Gomory **(4-19)-(4-21)**.
- Για την επίλυση του 1D-BPP, θεωρήθηκε ότι όλοι οι κάδοι έχουν το ίδιο πλάτος και επομένως, όλα επομένως τα υπο-προβλήματα είναι πανομοιότυπα και μπορεί να χρησιμοποιηθεί το απλοποιημένο μοντέλο **(4-51)-(4-53)**.
- Συγκρίνεται η κλασική μέθοδος δημιουργίας στήλης και η μέθοδος δημιουργίας πολλαπλών στηλών ως τον μέσο χρόνο επίλυσης [μέσω του χρόνου επεξεργασίας (process time ή Central Processing Unit, CPU)].
- Το κατώτερο όριο της τιμής της αντικειμενικής συνάρτησης, υπολογίστηκε για $\beta = 1$ σύμφωνα με τη σχέση **(3-8)**.
- Για κάθε πρόβλημα δοκιμής, ορίστηκε μία ανοχή ϵ, ώστε ο αλγόριθμος να τερματίζεται.

Η μέθοδος δημιουργίας πολλαπλών στηλών που προτείνεται, εξαρτάται από τη ρύθμιση δύο παραμέτρων της παραμέτρου a και της παραμέτρου $n = k + 1$. Σκοπός των υπολογιστικών πειραμάτων είναι η μελέτη του τρόπου με τον οποίο αυτές οι δύο παράμετροι επηρεάζουν την απόδοση της προτεινόμενης μεθόδου.

- Για την παράμετρο επίπεδου εξομάλυνσης a, εξετάζονται τρεις τιμές $a = 0{,}3$, $a = 0{,}6$ και $a = 0{,}8$.
- Για την παράμετρο του μέγιστου αριθμού στηλών που προστίθενται στο RMP σε κάθε επανάληψη $n = k + 1$, εξετάζεται η προσθήκη 2, 6 και 10 στηλών.

Από τα αποτελέσματα, για κάθε περίπτωση προσθήκης 2, 6 και 10 στηλών, εντοπίζεται η τιμή της παραμέτρου a, για την οποία επιτυγχάνονται καλύτερα αποτελέσματα (ελάχιστος μέσος χρόνος επίλυσης) για κάθε σύνολο παραδειγμάτων και συγκρίνεται με την κλασική μέθοδο δημιουργίας στήλης.

6.3 Περιγραφή της βιβλιοθήκης BPPLIB

Η BPPLIB είναι μια συλλογή υπολογιστικών προγραμμάτων, προβλημάτων δοκιμής-σημείων αναφοράς (benchmarks) και συνδέσμων για το μονοδιάστατο πρόβλημα συσκευασίας κάδου και κοπής αποθέματος. Η βιβλιοθήκη είναι διαθέσιμη στη διεύθυνση της αναφοράς [20].

Η ενότητα υπολογιστικών προγραμμάτων περιλαμβάνει δώδεκα προγράμματα: τα επτά μπορούν να κατέβουν απευθείας από τη σελίδα της βιβλιοθήκης, ενώ για τα υπόλοιπα πέντε παρέχονται οι αντίστοιχες διευθύνσεις. Η βιβλιοθήκη παρέχει δύο εκδόσεις: μία που χρησιμοποιεί τον εμπορικό επιλύτη CPLEX και μία που χρησιμοποιεί τον επιλύτη SCIP [106].

Η ενότητα αναφορών (benchmarks) παρέχει πάνω από έξι χιλιάδες (πιο συγκεκριμένα 6.195) παραδείγματα είτε από τη βιβλιογραφία είτε δημιουργήθηκαν τυχαία, μαζί με τις αντίστοιχες λύσεις σε ένα αρχείο με όνομα «Solutions.xlsx». Σε κάθε υπολογιστικό φύλλο αυτού του αρχείου βρίσκονται οι λύσεις κάθε κατηγορίας προβλημάτων και μπορούν να αντληθούν τα ακόλουθα: το όνομα του παραδείγματος, το καλύτερο χαμηλότερο όριο (LB) που προέκυψε από τον ακριβή αλγόριθμο, το καλύτερο ανώτερο όριο (UB) που προέκυψε από τον ακριβή αλγόριθμο, την κατάσταση (Status, «Solved» αν LB=UB, αλλιώς «Open»), σχόλια για τη λύση [20].

6.4 Τα διαθέσιμα παραδείγματα δοκιμής της βιβλιοθήκης BPPLIB

6.4.1 Η μορφή των παραδειγμάτων δοκιμής

Όλα τα παραδείγματα δοκιμής δίνονται στις δύο ακόλουθες μορφές [20]:

Για τη διατύπωση του 1D-CSP:

- Αριθμός του τύπου των αντικειμένων (m)
- Χωρητικότητα του κυλίνδρου (roll) (W)
- Για κάθε αντικείμενο $i(i = 1,2, \dots, m)$ δίνεται το αντίστοιχο βάρος w_i και η αντίστοιχη ζήτηση d_i.

Για τη διατύπωση του 1D-BPP:

- Αριθμός των αντικειμένων (m)
- Χωρητικότητα του κάδου (bin) (c)
- Για κάθε τύπο αντικείμενου i $(i = 1,2, \dots, m)$ δίνεται το αντίστοιχο βάρος w_i.

6.4.2 Παραδείγματα δοκιμής και η αντίστοιχη βιβλιογραφία

Αυτή η ενότητα περιέχει τις 1.615 περιπτώσεις που προτείνονται από την ακόλουθη βιβλιογραφία [20], [107]:

- E. Falkenauer [108]: Χωρίζονται σε δύο κατηγορίες των 80 παραδειγμάτων η καθεμία: η πρώτη κατηγορία έχει ομοιόμορφα κατανεμημένα μεγέθη αντικειμένων («Falkenauer U») με m μεταξύ 120 και 1000, και W ή c= 150. Οι περιπτώσεις της δεύτερης κατηγορίας («Falkenauer T») περιλαμβάνουν τριπλέτες, δηλαδή ομάδες τριών στοιχείων (ένα μεγάλο, δύο μικρά) που πρέπει να αντιστοιχιστούν στον ίδιο κάδο σε οποιαδήποτε βέλτιστη συσκευασία, με m μεταξύ 60 και 501 και W ή c = 1000.
- Scholl, R. Klein, and C. Jürgens [109]: Χωρίζονται σε τρία σύνολα των 720, 480 και 10, αντίστοιχα, ομοιόμορφα κατανεμημένα παραδείγματα με m μεταξύ 50 και 500. Η χωρητικότητα W ή c είναι μεταξύ 100 και 150 (σύνολο «Scholl 1»), ίση με 1.000 (σύνολο «Scholl 2»), και ίσο με 100.000 (σύνολο «Scholl 3»), αντίστοιχα.

- G. Wäscher & T. Gau [110]: Πρόκειται για 17 δύσκολα προβλήματα, με m μεταξύ 27 και 239 και W ή c = 10.000.
- P. Schwerin & G. Wäscher [111]: Χωρίζονται σε δύο σύνολα («Schwerin 1» και «Schwerin 2») των 100 παραδειγμάτων με m = 100 και m = 120, αντίστοιχα, και W = 1.000.
- J.E. Schoenfield [112]: Πρόκειται για 28 δύσκολα παραδείγματα με m μεταξύ 160 και 200 και W ή c = 1000.
- M. Delorme, M. Iori, and S. Martello (τυχαία παραδείγματα) [107]: Πρόκειται για 3.840. Έχουν διαφορετικές τιμές m (50, 100, 200, 300, 400, 500, 750, 1.000), W ή c (50, 75, 100, 120, 125, 150, 200, 300, 400, 75, 50.000), και του ελάχιστου (0.1 W ή c, 0.2 W ή c) και του μέγιστου (0.7 W ή c, 0.8 W ή c) βάρους αντικειμένου. Για κάθε τετράδα (m, W ή c, ελάχιστο βάρος, μέγιστο βάρος), δημιουργήθηκαν 10 παραδείγματα.
- M. Delorme, M. Iori, and S. Martello (Επαυξημένα Non-IRUP και επαυξημένα IRUP παραδείγματα) [107]: Πρόκειται για δύσκολα παραδείγματα.
- Παραδείγματα GI: Πρόκειται για 240 παραδείγματα που προτάθηκαν από τους T. Gschwind και S. Irnich [99], για τη σύγκριση της κλασικής μεθόδου δημιουργίας στήλης με μεθόδους σταθεροποίησής της.

Από τα προηγούμενα παραδείγματα δοκιμής, για το 1D-CSP, χρησιμοποιήθηκαν 10 παραδείγματα των Scholl, R. Klein, and C. Jürgens [109] και 160 παραδείγματα των T. Gschwind και S. Irnich [99]. Για το 1D-BPP, χρησιμοποιήθηκαν 60 παραδείγματα του συνόλου «T» του E. Falkenauer [108] και τα 100 παραδείγματα του πρώτου συνόλου «Schwerin 1» των P. Schwerin & G. Wäscher [111].

6.5 Περιγραφή του λογισμικού

6.5.1 Συνοπτική περιγραφή του λογισμικού

Για την σύγκριση της απόδοσης κλασικής μεθόδου δημιουργίας στήλης και μεθόδου δημιουργίας πολλαπλών στηλών, έχουν αναπτυχθεί υπολογιστικά προγράμματα. Η ανάπτυξη αυτών, έγινε στο Εργαστήριο Οργάνωσης Παραγωγής, Τμήμα Μηχανολόγων Μηχανικών του Π.Θ.. Τα χαρακτηριστικά του ηλεκτρονικού υπολογιστή που χρησιμοποιήθηκε είναι τα εξής:

- Επεξεργαστής (Processor): Intel(R) Core (TM) i3-4030U CPU @ 1,90GHz 1,90 GHz
- Εγκατεστημένη μνήμη RAM (Installed RAM): 4,00 GB
- Λογισμικό: Windows 10 Home 64-bit

Επιπλέον, οι βιβλιοθήκες βελτιστοποίησης που χρησιμοποιήθηκαν ήταν του IBM ILOG CPLEX Optimization Studio® v. 20.1.0 σε Microsoft Visual Studio® 2022 v. 17.1.0 C++ (Concert Technology in C++).

Ο συνδυασμός της γλώσσας προγραμματισμού C++ και του πακέτου βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® μπορεί να χρησιμοποιηθεί για τους παρακάτω λόγους:

<u>Γλώσσα προγραμματισμού C++:</u>

- Είναι μία γρήγορη γλώσσα προγραμματισμού [113].
- Αρκετοί επιλύτες (solvers) ανοικτού κώδικα έχουν αναπτυχθεί σε C++. Για παράδειγμα COIN-OR [114] και COR@L [115].

<u>IBM ILOG CPLEX Optimization Studio®:</u>

- Είναι ένα δημοφιλές πακέτο βελτιστοποίησης μεταξύ των ερευνητών του τομέα της επιχειρησιακής έρευνας. Έχει χρησιμοποιηθεί σε πολλές από τις αναφορές της εργασίας π.χ. [24], [33], [116].
- Είναι δωρεάν για ακαδημαϊκή χρήση [117].

6.5.2 Περιγραφή των υπολογιστικών προγραμμάτων

Τόσο για το 1D-CSP όσο και για το 1D-BPP, οι δύο κώδικες υλοποιούν τον γενικευμένο αλγόριθμο της **§ 5.3.1.2**. Οι κώδικες, ρυθμίζονται ώστε να διαβάζουν γραμμή προς γραμμή ένα αρχείο κειμένου (.txt) με την ονομασία «DataSet.txt» τόσες φορές όσες είναι το πλήθος των παραδειγμάτων δοκιμής του συνόλου. Κάθε γραμμή αποτελείται από 4 στήλες: στην πρώτη

στήλη βρίσκεται το όνομα του παραδείγματος .txt, στην δεύτερη και στην τρίτη στήλη βρίσκονται οι ακριβείς τιμές των UB και LB αντίστοιχα και τέλος στην τέταρτη στήλη βρίσκεται η ανοχή σύγκλισης. Αυτό γίνεται για να επιλύονται όλα τα παραδείγματα δοκιμής του συνόλου σειριακά. Τέλος, από την εκτέλεσή τους δημιουργούνται φάκελοι με την ίδια ονομασία του παραδείγματος δοκιμής και υπο-φάκελοι με την τιμή της παραμέτρου a, μέσα στους οποίους αποθηκεύονται τα παρακάτω έντεκα αρχεία κειμένου:

- Το αρχείο με την ονομασία «1.Initial patterns.txt», που περιέχει τα αρχικά πρότυπα (προϊόν αρχικοποίησης).
- Το αρχείο με την ονομασία «2.RMP_values.txt», που περιέχει την τιμή του ανώτερου ορίου της αντικειμενικής συνάρτησης και μπορεί να χρησιμοποιηθεί απευθείας από κάποιο λογισμικό για την εξαγωγή διαγράμματος ανώτερου ορίου-αριθμός επαναλήψεων.
- Το αρχείο με την ονομασία «3.Solution Summary.txt», που περιέχει συγκεντρωτικά τα αποτελέσματα: την ονομασία του προβλήματος (π.χ. 1D CSP), τον αλγόριθμο που χρησιμοποιήθηκε (π.χ. Classical Column Generation), τις τιμές των UB, LB, του υπο-προβλήματος, το σφάλμα των UB LB σε σχέση με την ακριβή τιμή τους και την κατάσταση της λύσης (π.χ. βέλτιστη) σε κάθε επανάληψη, τον αριθμό των παραγγελιών του εξεταζόμενου αρχείου (m), το πλάτος των προτύπων W ή c, τον αριθμό των επαναλήψεων, τον αριθμό των στηλών που έχουν προστεθεί στο RMP, την ανοχή σύγκλισης και τέλος, τον χρόνο επίλυσης (σε δευτερόλεπτα).
- Το αρχείο με την ονομασία «4.Final patterns.txt», περιέχει όλα τα πρότυπα που έχουν παραχθεί (στήλες), συμπεριλαμβανομένου και των αρχικών.
- Το αρχείο με την ονομασία «5.dual_values.txt», περιέχει τις τιμές των δυϊκών μεταβλητών (ίσες σε αριθμό με τον αριθμό των παραγγελιών m) σε κάθε επανάληψη.
- Το αρχείο με την ονομασία «6.primal_values.txt», περιέχει τις τιμές της μεταβλητής απόφασης λ του RMP.
- Το αρχείο με την ονομασία «7.stab_dual_values.txt», περιέχει τις τιμές των σταθεροποιημένων δυϊκών μεταβλητών.

- Το αρχείο με την ονομασία «8.report.txt», περιέχει τον αριθμό των επαναλήψεων, τον χρόνο επίλυσης και τον αριθμό στηλών που έχουν δημιουργηθεί.
- Το αρχείο με την ονομασία «CurrentMaster.txt», δίνει στον χρήστη τη δυνατότητα παρακολούθησης της τρέχουσας μορφής του RMP (οπτικοποίηση της τρέχουσας αντικειμενικής συνάρτησης και των περιορισμών).
- Το αρχείο με την ονομασία «CurrentSlave.txt», δίνει στον χρήστη τη δυνατότητα παρακολούθησης της τρέχουσας μορφής του υπο-προβλήματος (οπτικοποίηση της τρέχουσας αντικειμενικής συνάρτησης και των περιορισμών).
- Το αρχείο με την ονομασία «CurrentStabSlave.txt», δίνει στον χρήστη τη δυνατότητα παρακολούθησης της τρέχουσας μορφής του τροποποιημένου υπο-προβλήματος (οπτικοποίηση της τρέχουσας αντικειμενικής συνάρτησης και των περιορισμών).

6.5.3 Πληροφορίες για το IBM ILOG CPLEX Optimization Studio®

6.5.3.1 Ιστορικά στοιχεία

Το IBM ILOG CPLEX Optimization Studio® (ανεπίσημα αναφέρεται απλώς ως CPLEX) είναι ένα πακέτο λογισμικού βελτιστοποίησης. Αναπτύχθηκε αρχικά από τον Αμερικανό μαθηματικό Robert E. Bixby το 1987 [118], [119]. Έγινε εμπορικά διαθέσιμο από το 1988 με την επωνυμία CPLEX Optimizer από την εταιρεία CPLEX Optimization, που ιδρύθηκε από τον ίδιο και από την Αμερικανίδα χημικό μηχανικό Janet Lowe [118]–[120]. Το όνομα «CPLEX» προέρχεται από τον συνδυασμό του γράμματος «C», για τη γλώσσα προγραμματισμού και της λέξης «PLEX», για τη μέθοδο simplex που χρησιμοποιείται στο γραμμικό προγραμματισμό (επειδή η μέθοδος simplex υλοποιούνταν στην νέα για την εποχή γλώσσα προγραμματισμού C) [118].

Θεωρείται ένα από τα πιο διαδεδομένα υπολογιστικά πακέτα που χρησιμοποιούνται για την επίλυση προβλημάτων της επιχειρησιακής έρευνας είτε στον ακαδημαϊκό είτε στον επαγγελματικό χώρο [19]. Η Janet Lowe ερεύνησε την αγορά και ανέπτυξε μια στρατηγική για την εμπορία του CPLEX. Το σχέδιό της είχε τρεις διαστάσεις [19]:

Η πρώτη ήταν να δημιουργηθεί ένα ακαδημαϊκό πρόγραμμα πωλήσεων και μάρκετινγκ που παρείχε λογισμικό εμπορικής ποιότητας σε μειωμένες τιμές για ακαδημαϊκούς και ερευνητικούς σκοπούς. Μάλιστα, ένα μέτρο της επιτυχίας του IBM ILOG CPLEX Optimization Studio® στον ακαδημαϊκό χώρο είναι ότι το 95 % των εργασιών σε συνέδρια ανέφεραν ότι χρησιμοποίησαν το CPLEX [19].

Η δεύτερη βασική πτυχή του επιχειρηματικού σχεδίου ήταν η δημιουργία ενός ανεξάρτητου καναλιού πωλητή λογισμικού που επέτρεπε στους συνεργάτες της CPLEX να ενσωματώνουν το CPLEX σε εφαρμογές και να πληρώνουν δικαιώματα στην εταιρεία κάθε φορά που πωλούν την εφαρμογή τους. Ως αποτέλεσμα αυτού του μοντέλου, το CPLEX έχει γίνει ο τυπικός επιλύτης (solver), που χρησιμοποιείται στις εφαρμογές της εφοδιαστικής αλυσίδας.

Η τρίτη πτυχή του επιχειρηματικού σχεδίου ήταν η δημιουργία ενός μοντέλου τηλεπώλησης για τις πωλήσεις και την υποστήριξη του προϊόντος.

Το 1997 η CPLEX Optimization, εξαγοράστηκε από την ILOG και στη συνέχεια τον Ιανουάριο του 2009, η ILOG εξαγοράστηκε από την IBM, όπου το λογισμικό αναπτύσσεται ενεργά [19], [119].

Για τη δημιουργία και την ευρεία διάδοση του CPLEX, το Ινστιτούτο Επιχειρησιακής Έρευνας και Επιστημών Διοίκησης (Institute for Operations Research and the Management Sciences, INFORMS) απένειμε στους Robert Bixby και Janet Lowe το βραβείο INFORMS Impact Prize για το 2004 [19], [119].

6.5.3.2 Χαρακτηριστικά του IBM ILOG CPLEX Optimization Studio®

Το IBM ILOG CPLEX Optimization Studio® επιλύει προβλήματα ακέραιου προγραμματισμού, πολύ μεγάλα προβλήματα γραμμικού προγραμματισμού χρησιμοποιώντας διαφορετικές μορφές της μεθόδου simplex, μεθόδους εσωτερικού σημείου (IPM), κυρτά (convex) και μη κυρτά (non-convex) τετραγωνικά προβλήματα προγραμματισμού (Quadratic Programming problems, QP) και κυρτά τετραγωνικά περιορισμένα προβλήματα (convex quadratically constrained problems) [119].

Επίσης, το IBM ILOG CPLEX Optimization Studio® δίνει δυνατότητες προγραμματισμού σε διάφορες γλώσσες όπως C++, C#, Java, Python και γίνεται προσβάσιμο από άλλα,

ανεξάρτητα λογισμικά βελτιστοποίησης όπως AIMMS®, AMPL®, GAMS®, OptimJ®, TOMLAB®, MATLAB®, MS EXCEL® και SPSS® [119].

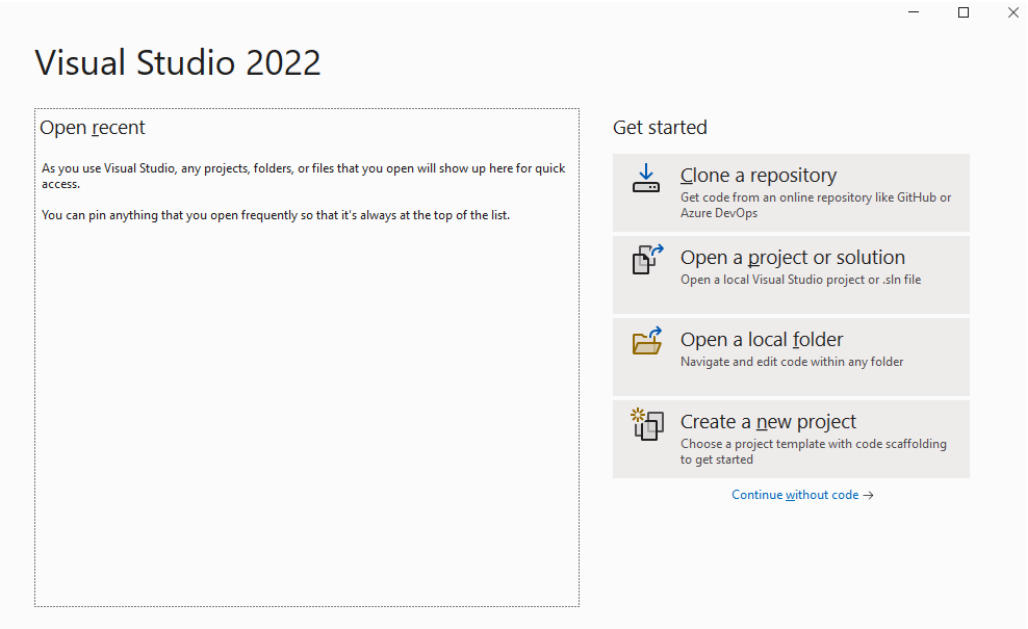
Τέλος, το πλήρες IBM ILOG CPLEX Optimization Studio® αποτελείται από: το CPLEX Optimizer Interactive για μαθηματικό προγραμματισμό, το CP Optimizer για προγραμματισμό περιορισμών, τη γλώσσα προγραμματισμού βελτιστοποίησης (Optimization Programming Language, OPL) και ένα στενά ενσωματωμένο ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE). Η τρέχουσα έκδοσή του είναι η IBM ILOG CPLEX Optimization Studio® v. 20.1.0 [121].

6.6 Εισαγωγή στο περιβάλλον του MS Visual Studio - σύνδεση με τις βιβλιοθήκες βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio®

6.6.1 Δημιουργία αρχείου .cpp στο MS Visual Studio

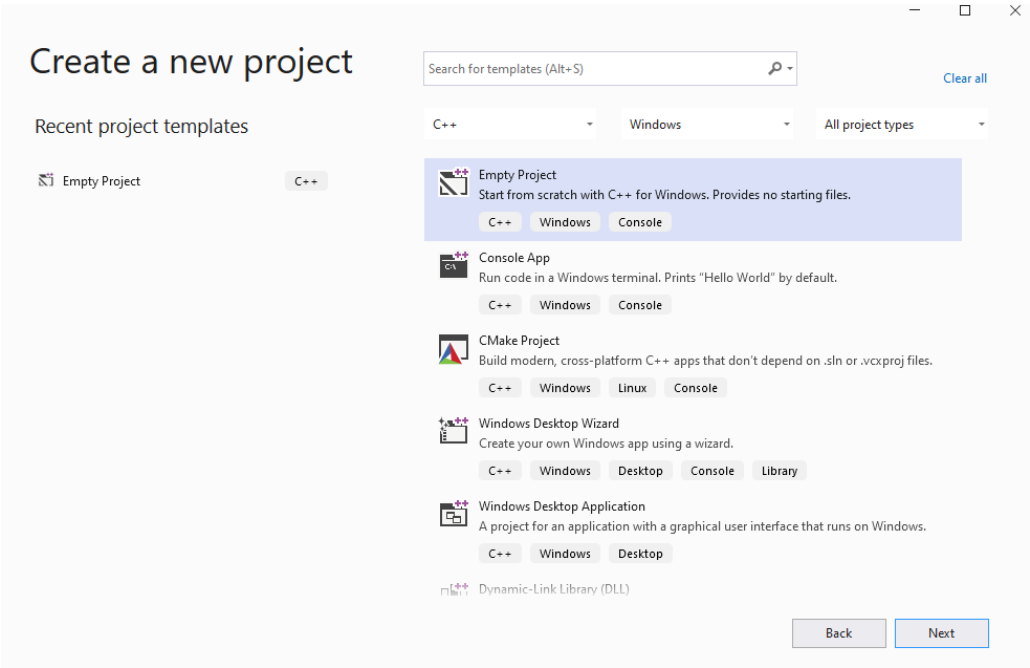
Για τη δημιουργία ενός αρχείου C++ (.cpp), ακολουθούνται τα παρακάτω πέντε βήματα:

Βήμα 1: Αρχικά, μετά το άνοιγμα του Visual Studio, επιλέγεται «Create a new project».



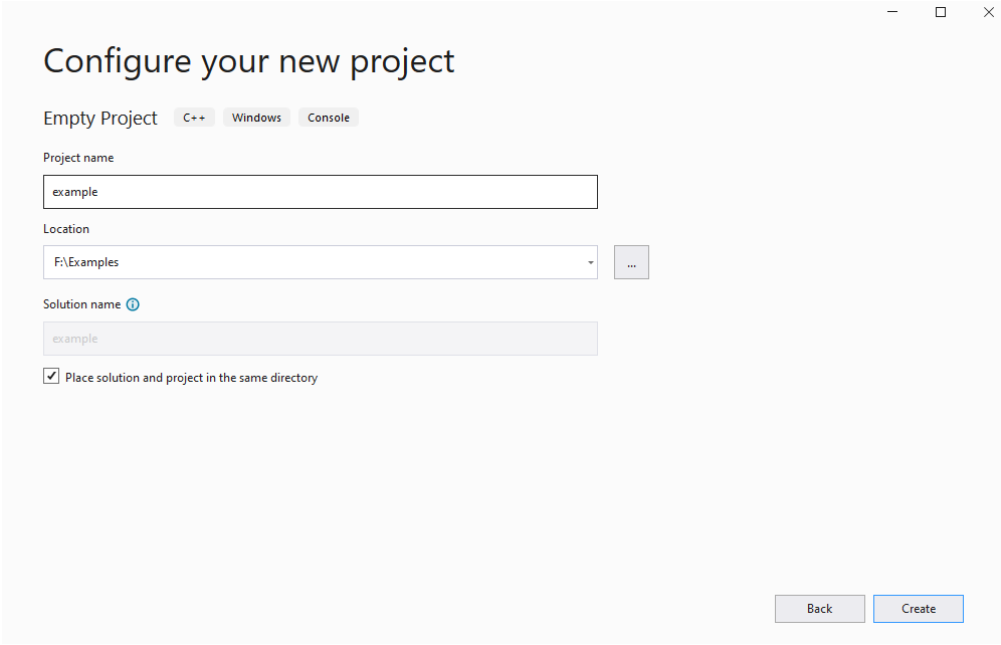
Εικόνα 6-1: Βήμα 1 δημιουργίας αρχείου.cpp

Βήμα 2: Επιλογή «C++», «Windows», «All project types», «Empty Project» και τέλος «Next».



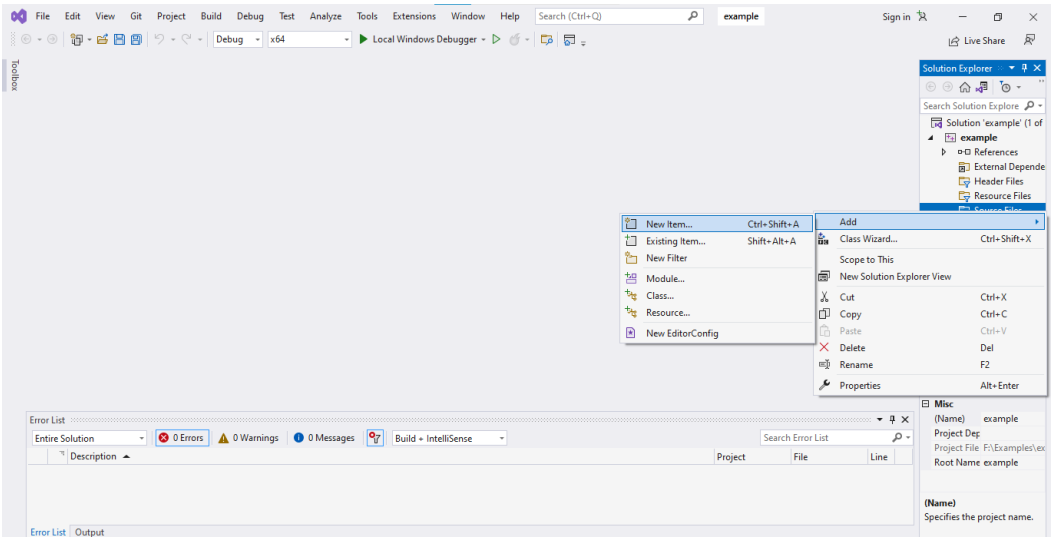
Εικόνα 6-2: Βήμα 2 δημιουργίας αρχείου.cpp

Βήμα 3: Ονομασία του Project (Project name), π.χ. «example», ορισμός της τοποθεσίας (Location) του Project, π.χ. «F:\Examples» και επιλογή «Place solution and project in the same directory» (προαιρετικά). Τέλος, επιλέγεται «Create».



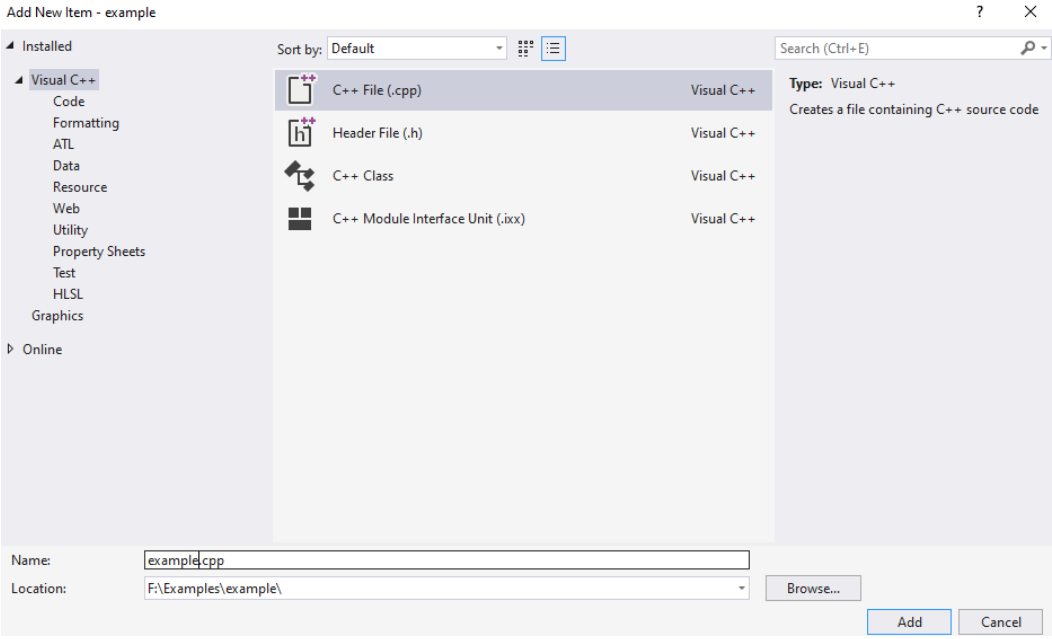
Εικόνα 6-3: Βήμα 3 δημιουργίας αρχείου.cpp

Βήμα 4: Δεξί κλικ στο «Source Files», επιλογή «Add», επιλογή «New Item».



Εικόνα 6-4: Βήμα 4 δημιουργίας αρχείου.cpp

Βήμα 5: Επιλογή «C++ File (.cpp)», ονομασία αρχείου στο πεδίο «Name», π.χ. «example» και επιλογή «Add».

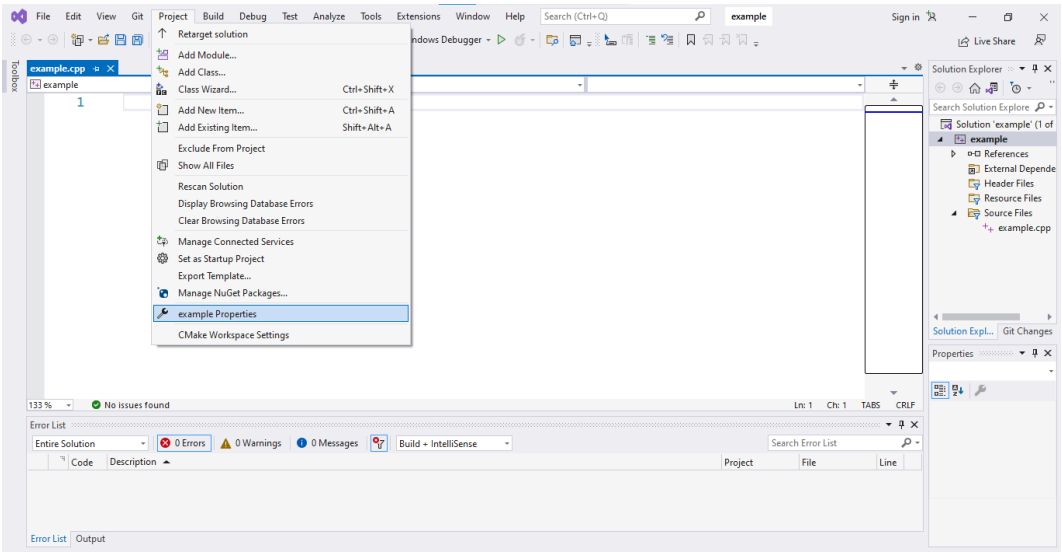


Εικόνα 6-5: Βήμα 5 δημιουργίας αρχείου.cpp.

6.6.2 Σύνδεση του Visual Studio με τις βιβλιοθήκες βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio®

Η σύνδεση του MS Visual Studio με τις βιβλιοθήκες βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio® πραγματοποιείται με την εκτέλεση των ακόλουθων έξι βημάτων:

Βήμα 1: Επιλογή «Project», <όνομα αρχείου, π.χ. example> Properties.



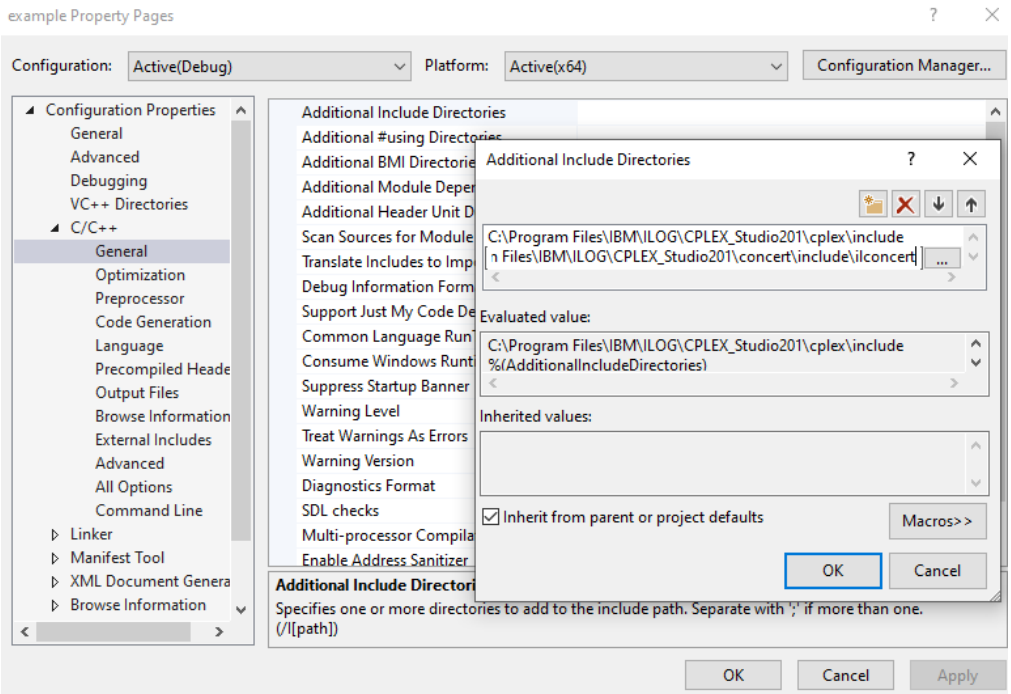
Εικόνα 6-6: Βήμα 1 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® .

Βήμα 2: Επιλογή «C/C++», «General», «Additional Include Directories», «Edit», προσθήκη των παρακάτω δύο διαδρομών των βιβλιοθηκών βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio®:

- C:\Program Files\IBM\ILOG\CPLEX_Studio201\cplex\include;
- C:\Program Files\IBM\ILOG\CPLEX_Studio201\concert\include

<u>Σημείωση:</u> το 201 στο «CPLEX_Studio201» σημαίνει η έκδοση 20.1 του IBM ILOG CPLEX Optimization Studio®.

Τέλος, επιλέγεται αρχικά «OK» κλείνει το παράθυρο και στη συνέχεια «Apply».



Εικόνα 6-7: Βήμα 2 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® .

Βήμα 3: Παραμονή στο «C/C++», επιλογή «Preprocessor», «Preprocessor Definitions», «Edit», εγγραφή:

WIN32;

_CONSOLE;

IL_STD

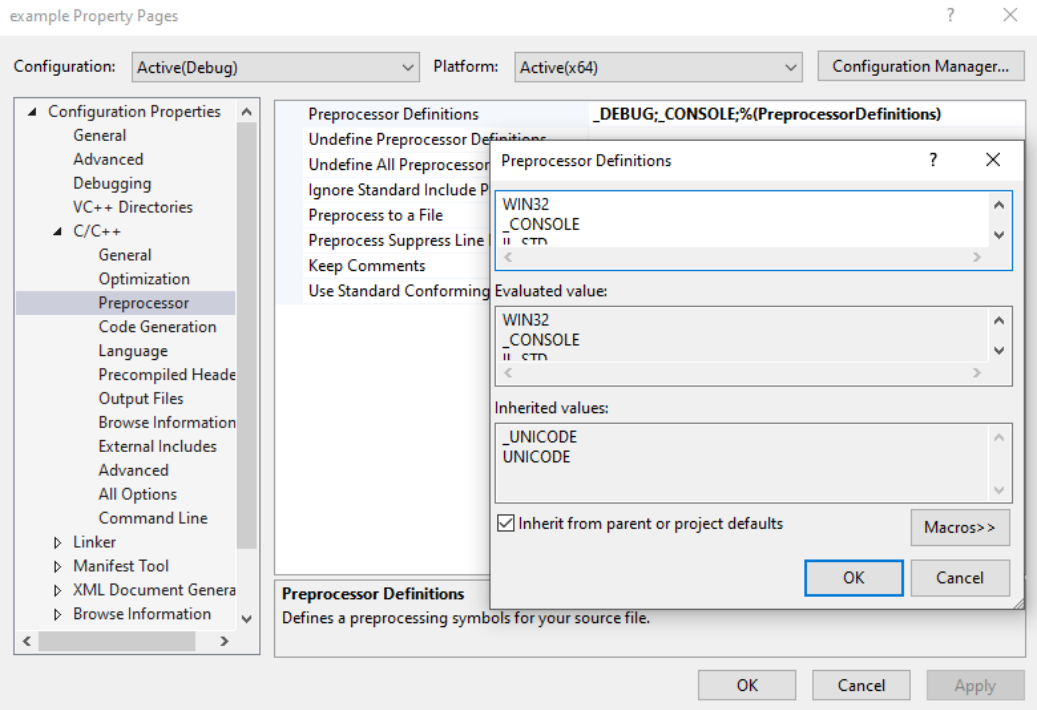
<u>Εναλλακτική εγγραφή:</u>

NDEBUG

_CONSOLE

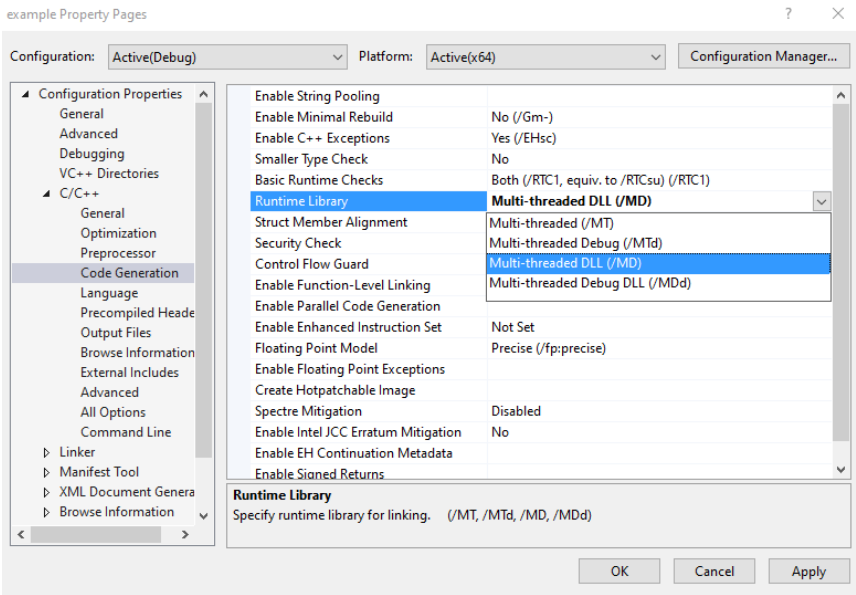
IL_STD

Τέλος, επιλέγεται αρχικά «OK» κλείνει το παράθυρο και στη συνέχεια «Apply».



Εικόνα 6-8: Βήμα 3 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® .

Βήμα 4: Παραμονή στο «C/C++», επιλογή «Code Generation», «Runtime Library», «Multi-threaded DLL (/MD)». Τέλος, επιλέγεται αρχικά «OK» κλείνει το παράθυρο και στη συνέχεια «Apply».

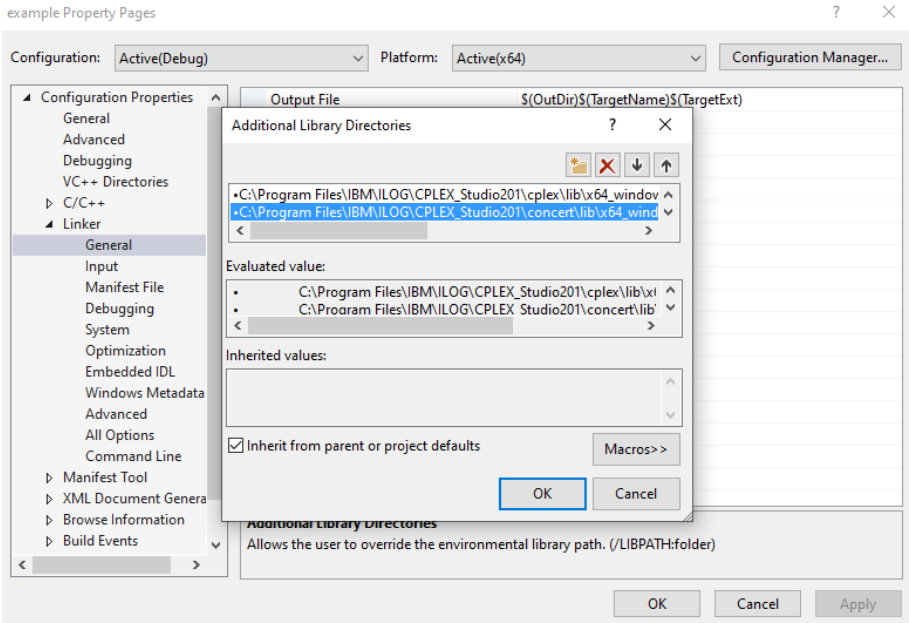


Εικόνα 6-9: Βήμα 4 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® .

Βήμα 5: Επιλογή «Linker», «General», «Additional Library Directories», «Edit» και προσθήκη των δύο παρακάτω διαδρομών:

- C:\Program Files\IBM\ILOG\CPLEX_Studio201\cplex\lib\x64_windows_msvc14\stat_mda;
- C:\Program Files\IBM\ILOG\CPLEX_Studio201\concert\lib\x64_windows_msvc14\stat_mda

Τέλος, επιλέγεται αρχικά «OK» κλείνει το παράθυρο και στη συνέχεια «Apply».

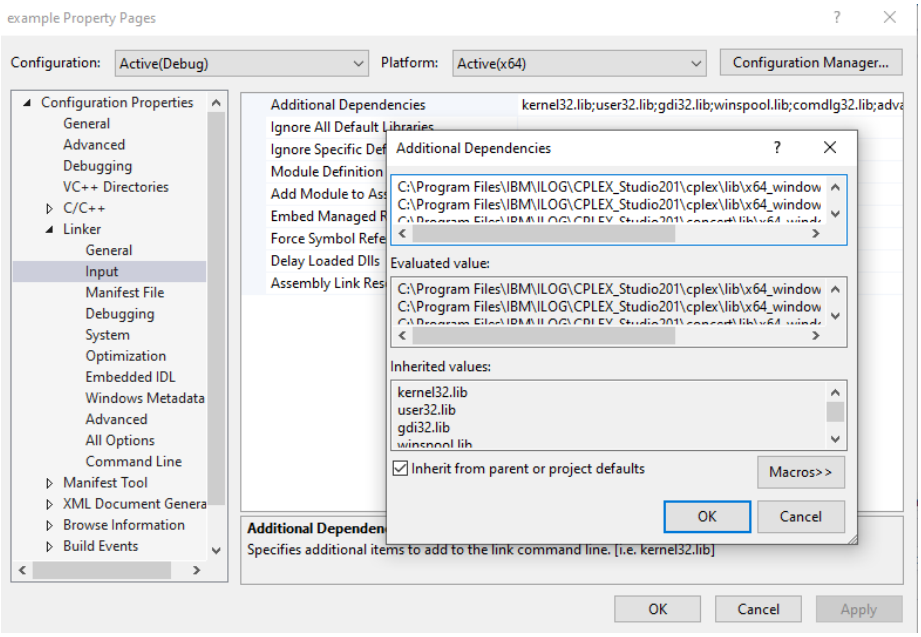


Εικόνα 6-10: Βήμα 5 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® .

Βήμα 6: Επιλογή «Linker», «Input», «Additional Dependencies», «Edit» και προσθήκη των τριών παρακάτω διαδρομών:

- C:\Program Files\IBM\ILOG\CPLEX_Studio201\cplex\lib\x64_windows_msvc14\stat_mda\cplex2010.lib;
- C:\Program Files\IBM\ILOG\CPLEX_Studio201\cplex\lib\x64_windows_msvc14\stat_mda\ilocplex.lib;
- C:\Program Files\IBM\ILOG\CPLEX_Studio201\concert\lib\x64_windows_msvc14\stat_mda\concert.lib

Τελικό στάδιο: επιλέγεται αρχικά «OK» κλείνει το παράθυρο, στη συνέχεια «Apply» και τέλος «OK».



Εικόνα 6-11: Βήμα 6 δημιουργίας σύνδεσης Visual Studio-βιβλιοθηκών βελτιστοποίησης IBM ILOG CPLEX Optimization Studio® .

6.6.3 Βασικές συναρτήσεις του IBM ILOG CPLEX Optimization Studio®

Όλοι οι ορισμοί που απαιτούνται για μια εφαρμογή IBM ILOG CPLEX Concert Technology εισάγονται συμπεριλαμβάνοντας το αρχείο <ilcplex/ilocplex.h>.

Η ILOSTLBEGIN είναι μια μακροεντολή που επιτρέπει να γίνεται εναλλαγή μεταξύ των δύο τύπων κώδικα χωρίς να χρειάζεται να αλλαχθεί ο πηγαίος κώδικας. Επίσης, η συνάρτηση usage καλείται σε περίπτωση που το πρόγραμμα εκτελείται με λανθασμένα ορίσματα γραμμής εντολών. Οι πιο βασικές συναρτήσεις είναι:

- IloEnv: δημιουργεί το περιβάλλον μοντελοποίησης
- IloModel: δημιουργεί το αντικείμενο μοντέλου (model object)
- IloNumVarArray: ορισμός μίας μονοδιάστατης μεταβλητής απόφασης
- IloRangeArray: για τον υπολογισμό των δυϊκών τιμών
- IloExpr: ορισμός μιας μεταβλητής που αποθηκεύει μία συλλογή όρων
- IloMinimize: προσθέτει μία αντικειμενική συνάρτηση ελαχιστοποίησης
- IloMaximize: προσθέτει μία αντικειμενική συνάρτηση μεγιστοποίησης
- IloCplex: δημιουργεί ένα αντικείμενο Cplex και επιλύει το πρόβλημα

- Model.add: προσθέτει την αντικειμενική συνάρτηση και τους περιορισμούς του προβλήματος
- cplex.solve (): επιλύει το μοντέλο
- cplex.cplex.getObjValue(): απόκτηση της τιμής της αντικειμενικής συνάρτησης
- cplex.getMIPRelativeGap(): απόκτηση της ποσοστιαίας απόκλισης από την ακριβή λύση
- cplex.getDual(IIoRange): απόκτηση της δυϊκής τιμής ενός περιορισμού
- cplex.getValue(): απόκτηση οποιαδήποτε τιμής.
- cplex.getReducedCost():απόκτηση του μειωμένου κόστους
- cplex.getSlack(): απόκτηση της χαλαρής μεταβλητής

Στο Παράρτημα Γ, παρουσιάζεται ο πηγαίος κώδικας που επιλύει το ακόλουθο πρόβλημα γραμμικού προγραμματισμού από το εγχειρίδιο χρήσης του CPLEX [122]:

Αντικειμενική συνάρτηση:

$$\max x_1 + 2x_2 + 3x_3 \tag{6-1}$$

Υποκειμενική τους περιορισμούς:

$$-x_1 + x_2 + x_3 \leq 20 \tag{6-2}$$

$$x_1 - 3x_2 + x_3 \leq 30 \tag{6-3}$$

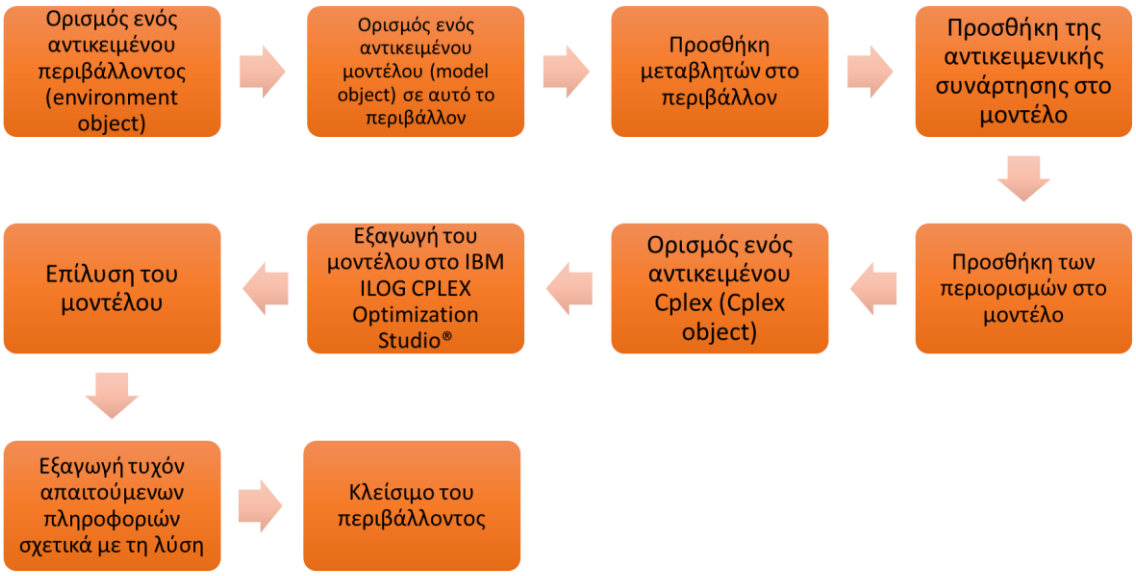
$$0 \leq x_1 \leq 40 \tag{6-4}$$

$$x_2, x_3 \geq 0 \tag{6-5}$$

Εκτός από τη βέλτιστη λύση και την βέλτιστη τιμή της αντικειμενικής συνάρτησης, ο πηγαίος κώδικας στο **Παράρτημα Γ**, υπολογίζει τις τιμές των δυϊκών μεταβλητών που αντιστοιχούν στους περιορισμούς, τα μειωμένα κόστη και τις τιμές των χαλαρών μεταβλητών.

6.6.4 Βασική ροή εργασίας για την επίλυση ενός προβλήματος μέσω C++ σε συνδυασμό με τις βιβλιοθήκες βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio®

Η βασική ροή εργασίας για την επίλυση ενός προβλήματος μέσω C++ και των βιβλιοθηκών βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio® παρουσιάζεται στην **Εικόνα 6-12**.



Εικόνα 6-12: Η βασική ροή εργασίας για την επίλυση ενός προβλήματος μέσω C++ και των βιβλιοθηκών βελτιστοποίησης του IBM ILOG CPLEX Optimization Studio®.

Κεφάλαιο 7. Αριθμητικά Αποτελέσματα

7.1 Περιγραφή των συμβολισμών

Πίνακας 7-1: Περιγραφή των συμβολισμών που χρησιμοποιούνται στους πίνακες των αριθμητικών αποτελεσμάτων.

συμβολισμός	**περιγραφή**
$\boldsymbol{n}$	Μέγιστος αριθμός στηλών σε κάθε επανάληψη
$\boldsymbol{k}$	Μέγιστος αριθμός παραγόμενων επιπλέον στηλών από το τροποποιημένο υπο-πρόβλημα σε κάθε επανάληψη
$\boldsymbol{m}$	Μέγιστος αριθμός περιορισμών των παραδειγμάτων δοκιμής
$\bar{\boldsymbol{m}}$	Μέσος αριθμός περιορισμών των παραδειγμάτων δοκιμής
$\boldsymbol{a}$	Παράμετρος επίπεδου εξομάλυνσης
$\hat{\boldsymbol{a}}$	Τιμή της παραμέτρου επίπεδου εξομάλυνσης για την οποία παρουσιάζονται καλύτερα αποτελέσματα.
iter.	Συνολικός αριθμός επαναλήψεων
time	Χρόνος επίλυσης (CPU time), σε δευτερόλεπτα
Av. iter	Μέσος συνολικός αριθμός επαναλήψεων
AST	Μέσος χρόνος επίλυσης (CPU time), σε δευτερόλεπτα (από Average Solution Time)
$\boldsymbol{N}$	Αριθμός έγκυρων παραδειγμάτων δοκιμής
*	Το πρόβλημα τερματίστηκε και δεν λαμβάνεται υπόψη στον μέσο όρο

7.2 Αριθμητικά αποτελέσματα της πρώτης φάσης υπολογιστικών πειραμάτων

7.2.1 Αριθμητικά αποτελέσματα της πρώτης φάσης υπολογιστικών πειραμάτων για το 1D-CSP

Πίνακας 7-2: Αριθμητικά αποτελέσματα από τη επίλυση των 10 παραδειγμάτων του συνόλου «HARD10» των Scholl, R. Klein, and C. Jürgens [109] *με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).*

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
α		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	*m*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
HARD0	199	112	60	67	33	60	23	62	23	52	47	55	30	57	44	48	78	56	53	58	50
HARD1	200	75	72	59	16	64	20	65	20	54	58	59	40	56	34	47	89	57	101	57	77
HARD2	199	92	40	63	33	60	25	62	56	52	68	52	58	52	44	46	86	51	65	51	71
HARD3	197	584	839	91	39	99	56	111	49	84	129	89	99	90	64	79	134	86	228	93	125
HARD4	198	232	158	84	37	101	21	110	59	67	50	74	41	77	80	62	88	80	132	77	113
HARD5	199	262	559	101	71	101	59	102	50	80	105	84	90	87	74	68	157	82	115	85	134
HARD6	199	260	173	95	66	93	53	84	32	75	93	81	59	82	78	45	50	46	34	56	54
HARD7	200	260	194	103	38	105	46	111	78	88	97	87	83	100	117	76	140	90	174	88	168
HARD8	199	86	38	70	29	68	27	77	51	58	55	58	42	58	66	55	90	59	73	61	75
HARD9	200	232	131	125	71	135	79	140	66	96	103	105	123	116	177	79	109	96	140	115	201
Μέσος όρος:	199	219,5	226,4	85,8	43,3	88,6	40,9	92,4	48,4	70,6	80,5	74,4	66,5	77,5	77,8	60,5	219,5	70,3	111,5	74,1	106,8

Πίνακας 7-3: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAA125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
a		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	m	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csAA125_1	125	204	67	103	24	105	24	100	21	68	42	71	40	72	52	61	61	58	61	75	83
csAA125_2*	125	12	9	10	3	10	3	10	3	9	8	10	6	8	4	9	6	8	3	8	3
csAA125_3	125	157	34	87	26	100	31	106	47	54	36	54	32	66	44	50	55	53	86	59	88
csAA125_4	125	177	26	172	92	164	63	175	95	66	46	72	61	76	55	65	90	68	94	74	124
csAA125_5	125	156	27	107	30	116	27	127	46	67	44	59	41	72	62	59	81	56	70	65	97
csAA125_6	125	288	57	158	43	161	73	181	84	89	86	93	111	99	118	73	60	78	85	92	154
csAA125_7	125	172	28	107	67	95	35	101	35	55	41	61	34	71	43	41	89	58	71	69	111
csAA125_8	125	180	29	99	31	122	47	121	55	62	47	70	46	70	62	60	77	66	93	68	86
csAA125_9	125	221	52	137	38	134	31	159	39	77	75	95	86	94	120	65	96	70	78	81	110
csAA125_10	125	285	88	160	54	171	78	179	60	98	105	98	95	122	115	77	105	73	91	109	219
csAA125_11*	125	4	10	9	2	4	7	4	9	3	4	4	2	4	3	3	10	4	7	4	3
csAA125_12	125	192	77	89	32	90	26	103	39	60	56	64	36	79	48	56	103	49	75	60	102
csAA125_13	125	307	100	108	28	126	41	124	23	66	42	70	43	77	54	56	53	67	60	62	68
csAA125_14	125	169	46	107	40	110	30	122	65	67	50	69	47	82	51	58	55	69	96	70	92
csAA125_15	125	301	94	83	35	83	49	82	44	54	38	54	51	62	47	47	82	48	71	51	68
csAA125_16	125	296	85	108	47	133	34	137	51	65	39	73	48	81	52	62	62	60	45	66	61
csAA125_17	125	229	52	158	43	144	38	148	39	84	65	90	60	95	94	68	69	72	122	84	113
csAA125_18	125	197	31	77	22	71	24	77	22	48	48	48	43	58	47	49	52	46	45	55	64
csAA125_19	125	232	31	106	37	107	28	118	31	71	54	72	60	84	67	62	76	68	80	77	87
csAA125_20	125	153	37	107	34	102	30	104	35	60	61	62	42	73	59	49	71	56	80	62	70
Μέσος όρος:	125	217,6	53,4	109,6	38,2	112,8	37,5	120,2	43,9	67,3	54,2	70,8	54,2	79,6	66,1	58,8	74,3	61,9	77,9	71,1	99,8

Πίνακας 7-4: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAB125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	*m*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csAB125_1	125	128	29	81	18	90	35	98	28	50	32	57	33	63	34	42	42	52	44	49	60
csAB125_2*	125	3	1	10	3	10	3	10	4	11	5	10	4	10	4	3	1	3	1	3	1
csAB125_3	125	139	32	86	39	91	26	98	28	54	36	53	35	65	39	43	41	50	39	58	53
csAB125_4	125	175	41	100	40	115	44	122	67	54	36	58	57	67	57	48	50	49	53	56	102
csAB125_5	125	119	16	64	29	67	28	73	27	40	44	43	32	46	43	39	53	40	47	48	56
csAB125_6	125	104	25	67	20	67	28	76	15	37	44	37	25	46	28	31	39	37	41	42	48
csAB125_7	125	91	43	62	14	67	14	74	27	42	27	46	44	51	52	37	48	42	31	43	33
csAB125_8	125	125	27	82	23	82	19	101	28	51	41	50	41	62	50	46	65	48	48	51	52
csAB125_9	125	140	36	89	24	83	32	104	44	50	33	55	32	63	32	43	51	50	48	60	66
csAB125_10	125	160	37	94	30	103	34	107	37	48	36	53	34	60	43	45	67	46	47	52	64
csAB125_11	125	107	21	76	28	78	39	74	59	44	28	41	27	55	42	38	46	41	43	41	59
csAB125_12	125	126	26	81	32	77	41	83	24	41	44	51	38	56	65	40	48	47	37	49	48
csAB125_13	125	133	36	80	22	82	29	102	28	49	38	53	32	55	32	40	39	45	54	51	57
csAB125_14	125	117	33	62	21	81	55	82	24	42	30	44	30	49	35	36	50	41	42	45	50
csAB125_15	125	99	31	67	17	65	16	66	16	44	29	47	36	53	41	39	43	38	42	46	52
csAB125_16	125	124	45	76	17	78	21	87	37	46	27	50	30	62	32	38	42	46	52	53	69
csAB125_17	125	108	18	66	19	78	23	74	36	36	28	45	35	51	40	38	35	41	32	45	35
csAB125_18	125	118	23	76	192	78	37	78	30	41	27	43	33	55	59	35	38	37	50	42	42
csAB125_19	125	19	8	19	10	18	5	18	3	15	47	15	8	19	10	15	14	17	30	17	17
csAB125_20	125	88	19	52	11	53	16	59	11	31	25	33	26	38	29	27	29	33	36	38	38
Μέσος όρος:	125	116,8	28,7	72,63	31,9	76,47	28,5	82,95	29,9	42,9	34,3	46	33,1	53,5	40,2	37,9	44,2	42,1	42,9	46,6	52,7

Πίνακας 7-5: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBA125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	*m*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csBA125_1	125	240	36	106	28	99	24	103	25	63	48	65	42	70	38	54	57	60	62	57	71
csBA125_2	125	187	27	117	40	119	39	122	37	64	47	74	56	81	67	61	61	63	58	71	84
csBA125_3	125	265	46	89	25	89	21	101	25	53	62	58	68	62	46	51	47	57	57	53	64
csBA125_4*	125	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1
csBA125_5	125	25	15	13	5	23	11	15	4	13	9	13	10	13	9	15	12	16	16	12	21
csBA125_6	125	204	111	112	36	107	26	18	27	67	52	71	57	82	69	59	58	72	153	72	77
csBA125_7	125	170	75	98	47	99	35	103	40	56	39	69	53	75	63	53	69	62	90	63	76
csBA125_8	125	222	53	81	31	90	43	95	60	53	31	55	44	69	37	47	65	52	56	58	49
csBA125_9	125	198	38	101	26	111	30	103	32	62	47	63	61	69	57	62	66	51	68	59	67
csBA125_10	125	147	31	76	20	84	27	99	26	57	57	54	55	66	43	58	51	52	59	65	70
csBA125_11	125	243	59	77	46	83	25	81	25	56	75	54	48	60	43	50	59	49	70	53	53
csBA125_12	125	145	32	79	23	84	24	85	24	51	49	57	43	59	63	47	45	49	114	48	49
csBA125_13	125	160	41	97	34	103	27	97	28	49	42	59	47	65	65	46	52	55	99	57	62
csBA125_14	125	207	41	114	28	107	25	110	24	70	69	67	51	72	75	59	71	62	93	66	70
csBA125_15	125	173	27	92	21	94	33	102	28	48	32	52	37	64	38	50	62	57	59	61	71
csBA125_16	125	103	16	71	17	56	12	55	13	53	39	49	29	50	30	46	48	49	39	45	44
csBA125_17	125	184	33	51	16	66	17	71	22	38	29	45	31	44	33	37	43	43	46	48	47
csBA125_18	125	182	47	104	33	103	30	103	28	63	46	65	42	73	51	53	57	60	60	69	64
csBA125_19	125	130	27	50	14	52	19	59	23	32	21	41	31	45	29	27	29	43	38	40	42
csBA125_20	125	239	71	93	34	95	35	95	31	54	43	55	43	59	45	51	65	55	79	58	80
Μέσος όρος:	125	180,2	43,5	85,32	27,6	87,58	26,5	85,11	27,5	52,7	44,1	56,1	44,6	62	47,4	48,7	53,5	53	69,3	55,5	61,1

Πίνακας 7-6: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBB125» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	m	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csBB125_1	125	129	51	82	45	82	30	85	26	44	29	49	54	56	33	40	48	47	47	47	36
csBB125_2	125	141	63	77	25	83	32	87	90	47	58	51	28	57	29	39	58	42	30	50	33
csBB125_3	125	124	32	94	56	84	25	93	25	43	25	59	35	66	55	42	30	52	60	57	44
csBB125_4	125	133	23	78	19	80	18	89	31	40	25	43	38	55	38	34	36	40	35	50	39
csBB125_5	125	130	42	81	52	87	49	88	46	55	46	60	36	60	35	41	34	43	38	51	41
csBB125_6	125	124	31	76	28	72	24	92	67	40	41	45	27	50	31	38	30	38	50	41	63
csBB125_7	125	129	63	72	47	73	46	81	36	41	41	46	33	47	45	35	39	38	38	39	36
csBB125_8	125	135	91	86	38	88	91	90	34	56	52	56	38	63	41	42	46	44	76	55	63
csBB125_9	125	125	39	83	36	81	34	88	229	47	36	51	44	55	35	36	38	46	59	45	53
csBB125_10	125	88	24	64	15	59	14	66	18	41	29	43	36	47	44	33	29	39	65	43	48
csBB125_11	125	142	42	75	19	78	34	81	21	52	40	49	47	54	42	49	56	43	55	42	66
csBB125_12	125	119	30	74	24	80	19	89	38	46	45	54	52	54	53	40	40	43	47	43	45
csBB125_13	125	81	15	53	11	48	12	59	27	33	23	37	37	42	45	31	30	31	56	38	34
csBB125_14	125	123	45	82	30	89	31	91	44	55	42	49	47	58	73	43	51	45	56	49	55
csBB125_15	125	68	21	43	32	44	11	50	15	30	41	32	32	36	31	29	32	32	34	29	32
csBB125_16*	125	5	2	5	2	5	2	5	2	5	2	5	2	5	2	5	2	5	2	5	2
csBB125_17	125	199	65	110	26	118	24	125	24	52	68	56	74	69	53	49	50	54	72	64	68
csBB125_18	125	126	38	80	40	88	32	84	21	47	29	54	65	60	100	44	47	44	84	48	43
csBB125_19*	125	3	3	3	1	3	1	3	1	3	3	3	3	3	3	3	3	3	3	3	3
csBB125_20	125	98	22	61	19	67	19	75	27	47	28	42	45	42	31	36	43	39	36	38	26
Μέσος όρος:	125	123	40,9	76,17	31,2	77,83	30,3	84,06	45,5	45,3	38,8	48,7	42,7	53,9	45,2	38,9	40,9	42,2	52,1	46,1	45,8

Πίνακας 7-7: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAA250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
α		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	m	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csAA250_1	250	34	15	26	13	25	10	25	12	16	18	21	21	18	22	19	32	20	43	21	38
csAA250_2	250	754	502	241	122	236	118	248	255	126	151	101	169	127	201	103	149	107	171	138	241
csAA250_3	250	332	218	160	67	149	61	178	89	106	133	105	189	107	191	96	194	96	179	112	361
csAA250_4	250	30	20	22	16	21	13	21	16	19	26	17	16	20	19	15	21	19	27	19	27
csAA250_5	250	653	242	129	45	140	55	166	75	78	108	87	93	86	83	72	147	74	140	83	168
csAA250_6	250	758	420	167	100	164	89	207	413	93	111	95	106	106	106	82	138	86	169	95	206
csAA250_7	250	452	117	174	65	201	88	210	104	112	128	113	148	114	158	94	251	104	237	102	183
csAA250_8	250	268	70	164	64	190	94	143	59	78	85	91	82	113	128	78	164	100	151	97	198
csAA250_9	250	343	169	183	77	187	75	202	91	107	140	109	122	134	168	92	142	92	122	123	268
csAA250_10	250	320	130	192	85	189	74	199	88	105	112	113	128	114	147	91	225	105	179	98	239
csAA250_11	250	365	149	227	90	206	88	224	126	112	120	111	124	129	125	103	215	101	202	131	257
csAA250_12	250	301	82	183	93	205	117	217	118	109	192	106	162	111	195	93	209	104	363	107	254
csAA250_13	250	57	20	36	19	37	18	36	19	25	34	23	18	27	30	25	41	22	43	22	38
csAA250_14	250	347	201	181	155	185	85	175	89	99	115	109	111	112	115	96	199	92	155	112	205
csAA250_15	250	199	65	111	45	91	29	125	62	73	92	74	69	78	73	64	86	62	104	72	127
csAA250_16	250	35	16	18	7	29	11	22	14	16	20	15	15	18	18	16	18	18	27	20	37
csAA250_17	250	346	161	199	78	239	143	235	160	114	132	126	152	130	156	90	132	104	247	131	321
csAA250_18	250	343	125	128	54	124	65	139	93	66	116	81	120	96	137	77	158	79	216	81	303
csAA250_19	250	231	120	142	60	138	52	147	54	85	91	90	104	100	109	82	148	79	160	94	147
csAA250_20*	250	40	16	35	15	22	9	18	7	77	81	80	105	70	87	62	124	84	266	72	120
Μέσος όρος:	250	324,6	149,6	135,9	63,5	138,9	64,7	146,9	97,2	80,8	100,3	83,35	102,7	90,5	113,4	72,5	139,7	77,4	160,1	86,5	186,9

Πίνακας 7-8: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csAB250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n = 2)$						**MCG** $(n = 6)$						**MCG** $(n = 10)$					
a		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	m	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csAB250_1	250	242	64	142	53	156	80	159	67	81	72	89	67	94	80	70	102	77	113	91	122
csAB250_2	250	217	52	135	72	137	85	157	65	73	68	80	61	95	83	60	80	69	93	84	114
csAB250_3	250	54	15	42	16	42	15	39	12	29	29	30	22	32	29	27	31	28	32	35	52
csAB250_4	250	243	68	152	46	159	63	154	56	77	70	88	73	97	95	64	106	73	158	81	143
csAB250_5	250	252	103	146	67	151	64	154	75	84	99	91	138	98	144	70	112	68	147	82	154
csAB250_6	250	241	83	144	71	155	68	165	72	85	100	88	107	95	110	69	139	71	177	97	201
csAB250_7	250	226	103	129	42	147	47	151	61	81	98	77	81	99	108	66	117	74	158	86	197
csAB250_8	250	186	82	86	45	75	32	97	67	55	58	66	86	68	86	50	86	60	159	65	162
csAB250_9	250	223	73	130	82	133	71	149	50	74	107	85	92	85	94	67	86	73	92	83	103
csAB250_10	250	222	81	146	58	153	70	155	69	82	120	88	79	96	79	67	113	90	116	82	110
csAB250_11	250	194	82	119	44	126	39	149	66	70	65	79	68	89	71	59	87	66	113	76	124
csAB250_12	250	195	99	115	46	124	142	42	69	71	55	70	58	86	70	64	105	63	99	74	114
csAB250_13	250	226	117	125	57	133	53	141	56	80	81	90	114	91	117	70	130	70	129	87	182
csAB250_14	250	194	90	120	40	128	58	128	59	77	121	80	84	85	80	70	126	76	181	76	169
csAB250_15	250	187	122	110	65	117	82	123	55	70	112	76	87	85	94	60	132	72	137	81	184
csAB250_16	250	210	185	139	66	137	74	142	63	70	88	78	101	87	83	60	141	66	81	76	106
csAB250_17	250	253	118	150	63	155	49	184	80	88	113	109	151	113	123	76	101	82	129	98	139
csAB250_18	250	237	106	150	60	163	63	172	110	78	110	88	149	106	120	69	119	76	110	89	156
csAB250_19	250	272	56	161	103	166	115	181	186	86	106	97	91	104	181	80	142	80	135	89	150
csAB250_20	250	182	71	105	37	110	30	117	41	66	105	68	57	85	96	57	92	65	102	67	113
Μέσος όρος:	250	212,8	88,5	127,3	56,7	133,4	65	138	69	73,9	88,9	80,9	88,3	89,5	97,2	63,8	107,4	69,9	123,1	79,9	139,7

Πίνακας 7-9: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBA250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n = 2)$						**MCG** $(n = 6)$						**MCG** $(n = 10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	*m*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csBA250_1	250	240	101	190	88	212	108	189	104	103	131	98	115	140	156	88	154	106	200	108	200
csBA250_2	250	205	111	191	98	194	95	211	102	118	233	126	120	117	109	104	169	109	183	125	180
csBA250_3	250	473	113	209	101	215	97	217	121	115	153	114	123	139	154	89	163	98	190	104	186
csBA250_4	250	144	83	161	88	173	66	176	72	98	130	109	153	115	144	89	167	93	178	92	197
csBA250_5	250	283	130	193	126	171	119	186	130	95	112	115	172	131	195	95	169	100	204	107	211
csBA250_6	250	212	92	199	78	204	113	211	86	118	155	126	158	134	215	100	234	115	213	120	178
csBA250_7	250	220	112	202	107	218	97	234	127	108	209	116	127	128	138	80	118	104	149	115	170
csBA250_8	250	225	91	213	85	203	86	193	89	112	127	114	124	126	151	93	145	96	216	116	216
csBA250_9	250	210	132	199	127	192	91	182	92	112	105	111	124	1234	140	92	170	93	156	117	264
csBA250_10	250	209	86	194	71	201	82	207	95	107	122	107	124	128	149	94	158	104	205	118	184
csBA250_11	250	229	131	219	122	214	90	220	143	111	157	111	119	128	174	96	167	94	148	110	180
csBA250_12	250	213	119	204	129	209	83	217	107	114	146	117	149	135	206	98	171	94	160	124	220
csBA250_13	250	211	120	190	81	153	64	202	86	105	136	108	138	134	141	97	142	99	207	112	208
csBA250_14	250	222	86	205	74	212	90	222	102	114	150	123	198	136	255	104	160	107	192	126	183
csBA250_15	250	243	181	216	165	198	92	210	141	107	110	114	128	130	146	101	154	104	253	96	165
csBA250_16	250	192	104	184	87	203	114	223	134	99	109	139	157	130	135	105	210	101	183	113	170
csBA250_17	250	246	173	215	154	179	232	213	146	107	105	106	133	133	158	90	207	98	168	111	169
csBA250_18	250	184	83	160	66	188	92	208	98	108	115	111	159	129	238	77	172	106	197	108	216
csBA250_19	250	196	109	164	93	203	177	198	125	100	126	119	170	121	172	87	147	101	188	116	202
csBA250_20	250	249	111	217	95	229	105	241	133	118	145	120	162	134	118	105	161	111	237	114	201
Μέσος όρος:	250	230,3	113	196,3	102	198,6	105	208	112	108,5	138,8	115,2	142,6	185,1	164,7	94,2	166,9	101,65	191,35	112,6	195

Πίνακας 7-10: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «csBB250» των T. Gschwind και S. Irnich [99] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n = 2)$						**MCG** $(n = 6)$						**MCG** $(n = 10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	m	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
csBB250_1	250	199	35	121	41	126	37	138	52	70	61	82	72	87	87	61	83	64	83	82	109
csBB250_2	250	232	61	130	46	139	55	144	76	72	63	83	94	93	85	64	92	77	105	87	113
csBB250_3	250	225	60	126	67	142	60	141	82	82	74	86	159	90	102	65	91	78	149	81	147
csBB250_4	250	225	101	140	63	138	41	142	58	74	80	81	74	91	110	71	205	71	135	85	151
csBB250_5	250	194	50	115	40	129	83	135	60	69	67	72	79	93	158	61	129	62	198	82	174
csBB250_6	250	209	172	121	45	125	64	130	67	72	64	82	109	91	103	61	113	64	118	85	180
csBB250_7	250	41	18	29	14	30	11	31	28	26	26	26	30	25	36	23	43	23	51	27	61
csBB250_8	250	244	121	142	43	153	73	166	78	80	105	87	129	95	117	74	171	74	152	73	98
csBB250_9	250	250	94	153	84	152	55	175	130	84	91	94	132	108	153	71	166	79	116	97	132
csBB250_10	250	272	59	160	92	163	120	171	90	93	118	92	125	91	78	65	92	76	103	88	134
csBB250_11	250	246	72	127	51	140	44	152	62	84	119	94	101	103	85	64	95	72	105	91	162
csBB250_12	250	119	29	82	39	87	34	86	37	51	51	55	56	69	55	51	78	56	67	60	105
csBB250_13*	250	15	6	10	5	10	3	11	5	10	10	10	8	11	7	10	10	10	12	11	16
csBB250_14	250	236	130	147	64	144	79	160	78	76	80	85	149	94	170	63	109	70	172	78	176
csBB250_15	250	286	132	166	67	173	60	194	74	87	128	98	128	114	125	86	175	85	152	98	134
csBB250_16	250	58	20	33	14	35	29	39	11	28	92	32	40	35	37	27	48	29	48	31	37
csBB250_17	250	270	94	150	44	167	73	176	97	80	99	87	112	99	152	79	197	80	197	93	133
csBB250_18	250	238	73	148	105	162	95	168	105	88	127	86	99	101	324	74	203	80	261	88	150
csBB250_19	250	36	28	23	14	27	16	23	24	23	24	21	24	21	86	25	58	22	35	21	40
csBB250_20	250	154	64	99	103	100	66	110	137	58	80	64	78	65	57	56	70	65	86	61	115
Μέσος όρος:	250	196,5	74,4	116,4	54,5	122,7	57,6	130,6	70,8	68,3	81,5	74,1	94,2	82,4	111,6	60,1	116,7	64,6	122,8	74,1	123,7

Πίνακας 7-11: Συγκεντρωτικός πίνακας των αριθμητικών αποτελεσμάτων των παραδειγμάτων δοκιμής του 1D-CSP που χρησιμοποιήθηκαν, με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

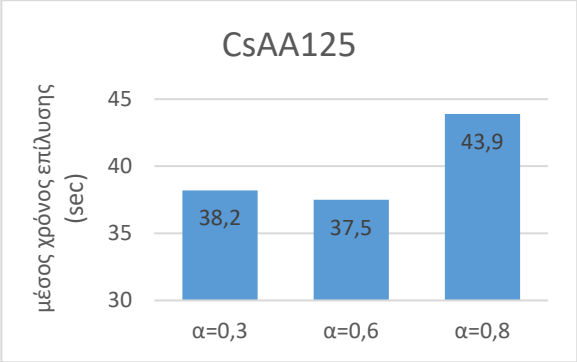
Μέθοδος:			**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$			1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Όνομα συνόλου παραδειγμάτων	m	N	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*
CsAA125	125	18	217,6	53,4	109,6	38,2	112,8	37,5	120,2	43,9	67,3	54,2	70,8	54,2	79,6	66,1	58,8	74,3	61,9	77,9	71,1	99,8
CsAB125	125	19	116,8	28,7	72,63	31,9	76,47	28,5	82,95	29,9	42,9	34,3	46	33,1	53,5	40,2	37,9	44,2	42,1	42,9	46,6	52,7
CsBA125	125	19	180,2	43,5	85,32	27,6	87,58	26,5	85,11	27,5	52,7	44,1	56,1	44,6	62	47,4	48,7	53,5	53	69,3	55,5	61,1
CsBB125	125	18	123	40,9	76,17	31,2	77,83	30,3	84,06	45,5	45,3	38,8	48,7	42,7	53,9	45,2	38,9	40,9	42,2	52,1	46,1	45,8
HARD10	199	10	219,5	226,4	85,8	43,3	88,6	40,9	92,4	48,4	70,6	80,5	74,4	66,5	77,5	77,8	60,5	219,5	70,3	111,5	74,1	106,8
CsAA250	250	20	324,6	149,6	135,9	63,5	138,9	64,7	146,9	97,2	80,8	100,3	83,35	102,7	90,5	113,4	72,5	139,7	77,4	160,1	86,5	186,9
CsAB250	250	20	212,8	88,5	127,3	56,7	133,4	65	138	69	73,9	88,9	80,9	88,3	89,5	97,2	63,8	107,4	69,9	123,1	79,9	139,7
CsBA250	250	20	230,3	113	196,3	102	198,6	105	208	112	108,5	138,8	115,2	142,6	185,1	164,7	94,2	166,9	101,65	191,35	112,6	195
CsBB250	250	19	196,5	74,4	116,4	54,5	122,7	57,6	130,6	70,8	68,3	81,5	74,1	94,2	82,4	111,6	60,1	116,7	64,6	122,8	74,1	123,7
συνολικά	-	163	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

7.2.2 Σχολιασμός των αποτελεσμάτων των υπολογιστικών πειραμάτων για το 1D-CSP

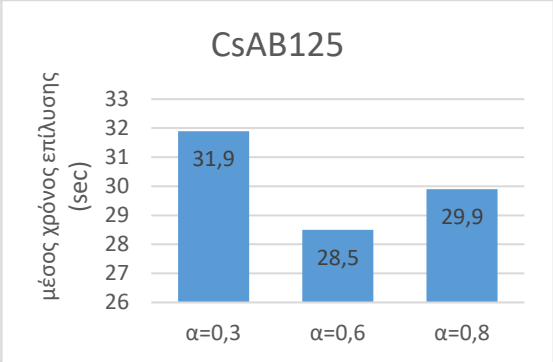
Από τα αποτελέσματα του **Πίνακας 7-11**, μπορούν να σημειωθούν οι ακόλουθες παρατηρήσεις:

Για την παράμετρο a:

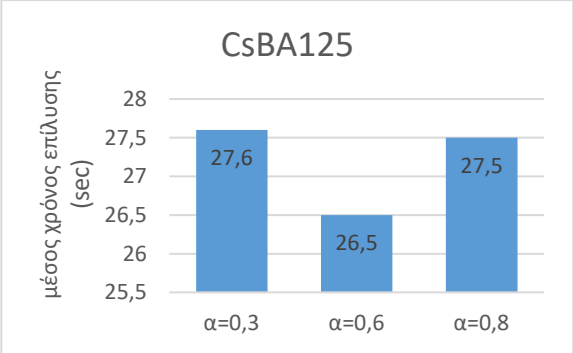
- Ο τρόπος με τον οποίο οι παράμετροι a και $n = k + 1$, επηρεάζουν τον χρόνο επίλυσης και τον αριθμό των επαναλήψεων, εξαρτάται από το πρόβλημα που αντιμετωπίζεται κάθε φορά.
- Όπως σημειώθηκε στην **§ 5.3.1.1**, για την κλασική μέθοδο δημιουργίας στήλης ισχύει $a = 1$. Η μείωση της τιμής μειώνει τον μέσο αριθμό των επαναλήψεων και τον μέσο χρόνο επίλυσης.
- Ο ελάχιστος μέσος χρόνος επίλυσης για την προτεινόμενη μέθοδο MCG με την προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, επιτυγχάνεται στα περισσότερα προβλήματα είτε για $a = 0{,}3$ είτε για $a = 0{,}6$ με μικρές διαφορές. Εξαίρεση αποτελεί η περίπτωση του συνόλου «HARD10» όπου με την προσθήκη 10 στηλών η τιμή αυτή είναι $a = 0{,}8$.
- Όσο αυξάνεται ο αριθμός των προστιθέμενων στηλών, φαίνεται ότι ο ελάχιστος μέσος χρόνος επίλυσης, στα περισσότερα σύνολα παραδειγμάτων επιτυγχάνεται για $a = 0{,}3$.



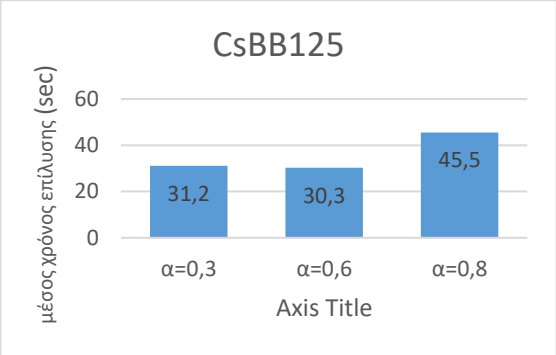
(α)



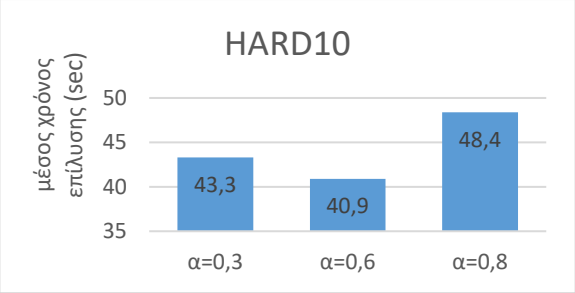
(β)



(γ)

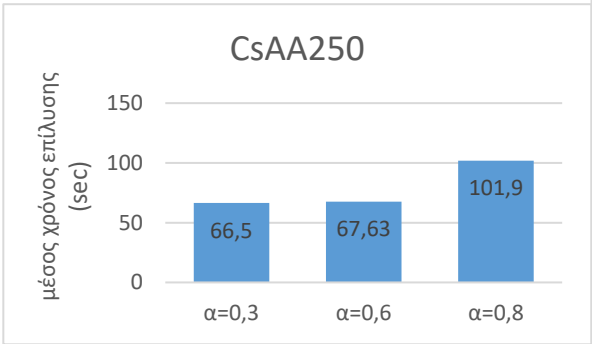


(δ)

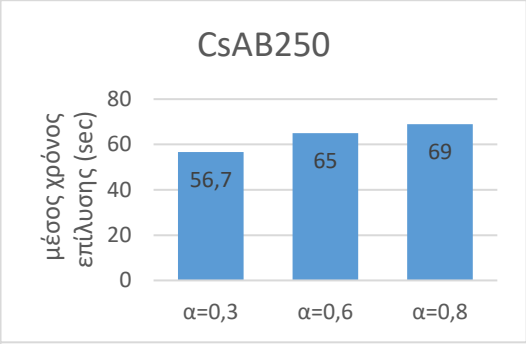


(ε)

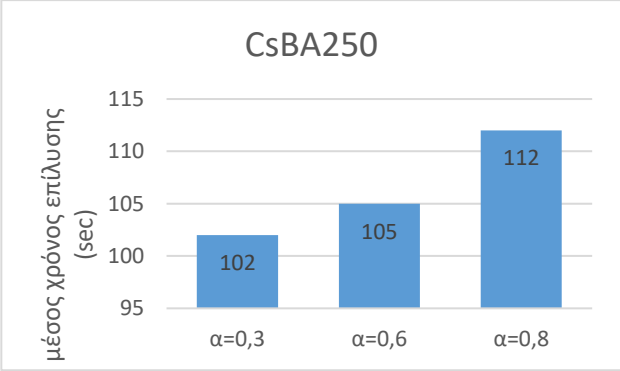
Εικόνα 7-1: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) csAA125, (β) csAB125, (γ) csBA125, (δ) csBB125 και (ε) HARD10, με την προσθήκη 2 στηλών στο RMP σε κάθε επανάληψη, για 3 διαφορετικές τιμές της παραμέτρου επίπεδου εξομάλυνσης α.



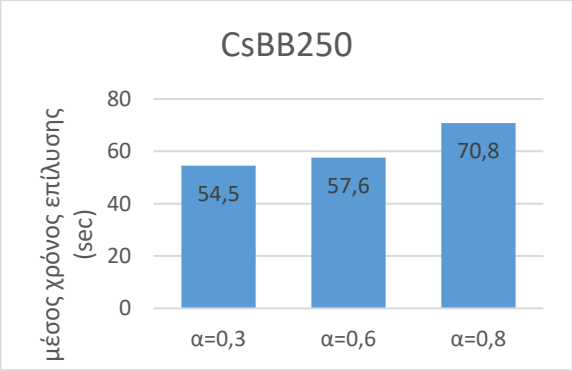
(α)



(β)



(γ)

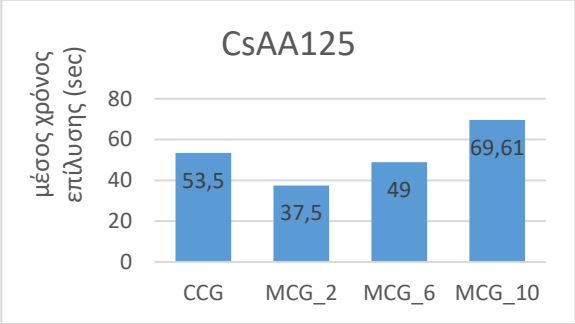


(δ)

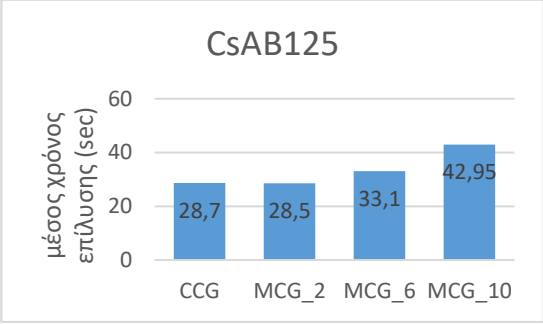
Εικόνα 7-2: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) csAA250, (β) csAB250, (γ) csBA250 και (δ) csBB250, με την προσθήκη 2 στηλών στο RMP σε κάθε επανάληψη, για 3 διαφορετικές τιμές της παραμέτρου επίπεδου εξομάλυνσης α.

<u>Για την παράμετρο k:</u>

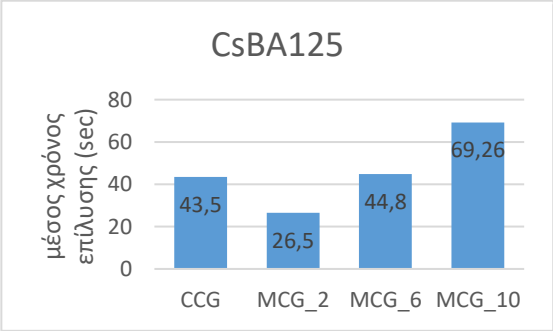
- Αν και ο ελάχιστος μέσος χρόνος επίλυσης όλων των εξεταζόμενων παραδειγμάτων δοκιμής επιτυγχάνεται με την προσθήκη δύο στηλών στο RMP σε κάθε επανάληψη, μέχρι και την προσθήκη έως και 6 στηλών, τα αποτελέσματα είναι καλύτερα από την κλασική μέθοδο δημιουργίας στηλών.
- Όσο αυξάνεται ο αριθμός των στηλών $n > 6$, που προστίθενται στο RMP, τόσο αυξάνεται και ο μέσος χρόνος επίλυσης. Αυτό φαίνεται από τα αποτελέσματα με προσθήκη 10 στηλών σε κάθε επανάληψη και στους πίνακες: **Πίνακας 7-12**,
- **Πίνακας** 7-13 και στον **Πίνακας 7-14**. Δηλαδή, η αύξηση του αριθμού των προστιθέμενων στηλών $n > 6$, «υπερφορτώνει» το RMP.


CsAA125
μέσος χρόνος επίλυσης (sec)
80
60
40
20
0
53,5
37,5
49
69,61
CCG
MCG_2
MCG_6
MCG_10

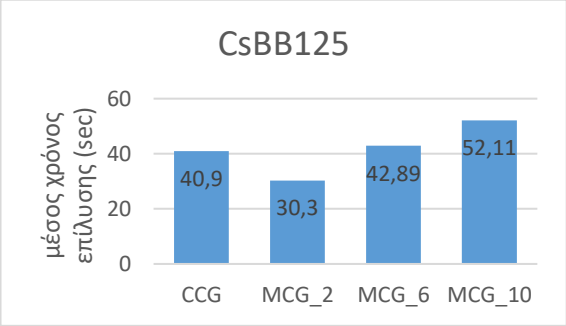
(α)


CsAB125
μέσος χρόνος επίλυσης (sec)
60
40
20
0
28,7
28,5
33,1
42,95
CCG
MCG_2
MCG_6
MCG_10

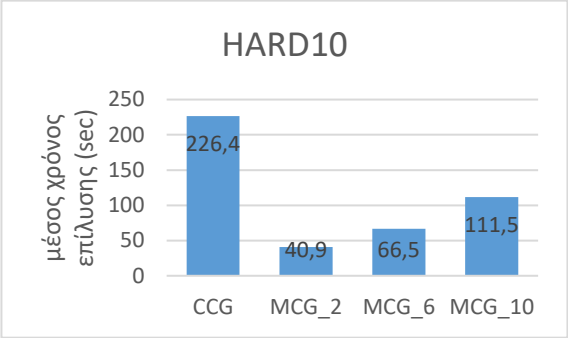
(β)


CsBA125
μέσος χρόνος επίλυσης (sec)
80
60
40
20
0
43,5
26,5
44,8
69,26
CCG
MCG_2
MCG_6
MCG_10

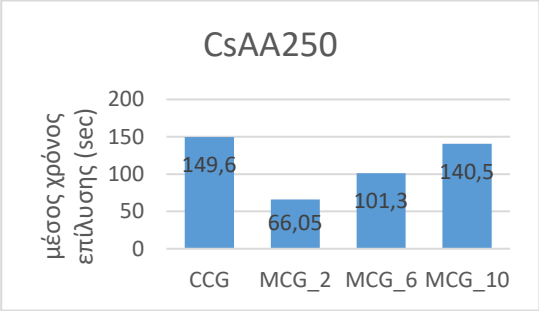
(γ)


CsBB125
μέσος χρόνος επίλυσης (sec)
60
40
20
0
40,9
30,3
42,89
52,11
CCG
MCG_2
MCG_6
MCG_10

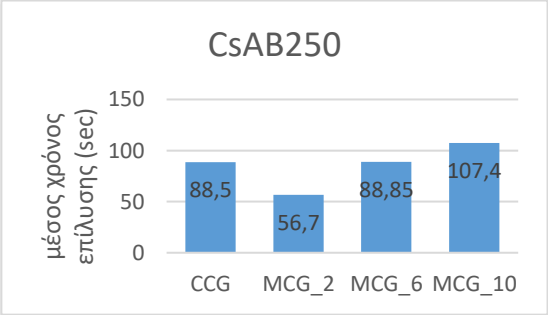
(δ)


HARD10
μέσος χρόνος επίλυσης (sec)
250
200
150
100
50
0
226,4
40,9
66,5
111,5
CCG
MCG_2
MCG_6
MCG_10

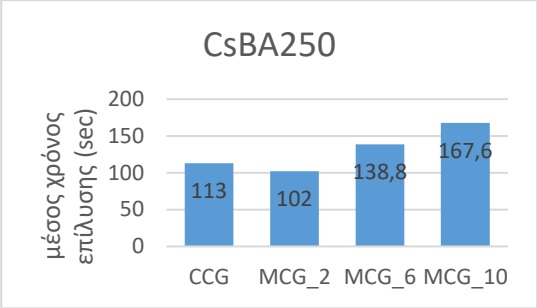
(ε)

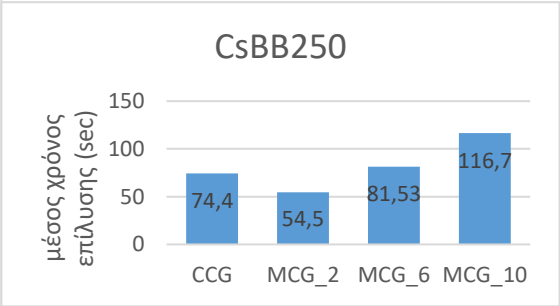

CsAA250
μέσος χρόνος επίλυσης (sec)
200
150
100
50
0
149,6
66,05
101,3
140,5
CCG
MCG_2
MCG_6
MCG_10

(στ)


CsAB250
μέσος χρόνος επίλυσης (sec)
150
100
50
0
88,5
56,7
88,85
107,4
CCG
MCG_2
MCG_6
MCG_10

(ζ)





(η) *(ϑ)*

Εικόνα 7-3: Μέσος χρόνος επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) csAA125, (β) csAB125, (γ) csBA125, (δ) csBB125, (ε) HARD10, (στ) csAA250, (ζ) csAB250, (η) csBA250 και (ϑ) csBB250, με την κλασική μέθοδο δημιουργίας στήλης (CCG) και της μεθόδου δημιουργίας πολλαπλών στηλών, με την προσθήκη 2 (MCG_2), 6 (MCG_6) και 10 (MCG_10) στηλών στο RMP σε κάθε επανάληψη αντίστοιχα, για την αντίστοιχη τιμή $\hat{a}$.

Πίνακας 7-12: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 2 στηλών στο RMP του 1D-CSP σε κάθε επανάληψη και με την τιμή της παραμέτρου $\hat{a}$ για κάθε σύνολο παραδειγμάτων).

Μέθοδος:				**CCG**	**MCG** $(n = 2)$	**(%) Ποσοστό μεταβολής του μέσου χρόνου επίλυσης**	**Μέσος όρος των ποσοστών μεταβολής του μέσου χρόνου επίλυσης**
Όνομα συνόλου παραδειγμάτων	m	N	$\hat{a}$	Μέσος χρόνος επίλυσης (seconds)	Μέσος χρόνος επίλυσης (seconds)		
CsAA125	125	18	0,6	53,4	37,5	-29,7 %	-34,1 %
CsAB125	125	19	0,6	28,7	28,5	-0,7 %	
CsBA125	125	19	0,6	43,5	26,5	-39,0 %	
CsBB125	125	18	0,6	40,9	30,3	-25,9 %	
HARD	199	10	0,6	226,4	40,9	-81,9 %	
CsAA250	250	20	0,3	149,6	63,5	-57,5 %	
CsAB250	250	20	0,3	88,5	56,7	-35,9 %	
CsBA250	250	20	0,3	113	102	-9,7 %	
CsBB250	250	19	0,3	74,4	54,5	-26,7 %	
Μέσος όρος των μέσων όρων	-	-	-	90,9	48,9	-	

Πίνακας 7-13: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 6 στηλών στο RMP του 1D-CSP σε κάθε επανάληψη και με την τιμή της παραμέτρου $\hat{a}$ για κάθε σύνολο παραδειγμάτων).

Μέθοδος:				**CCG**	**MCG** $(\boldsymbol{n = 6})$	**(%) Ποσοστό μεταβολής του μέσου χρόνου επίλυσης**	**Μέσος όρος των ποσοστών μεταβολής του μέσου χρόνου επίλυσης**
Όνομα συνόλου παραδειγμάτων	m	N	$\hat{a}$	Μέσος χρόνος επίλυσης (seconds)	Μέσος χρόνος επίλυσης (seconds)		
CsAA125	125	18	0,3 και 0,6	53,4	54,2	+1,4 %	-6,5 %
CsAB125	125	19	0,6	28,7	33,1	+15,3 %	
CsBA125	125	19	0,3	43,5	44,1	+1,3 %	
CsBB125	125	18	0,3	40,9	38,8	-5,1 %	
HARD	199	10	0,6	226,4	66,5	-70,6 %	
CsAA250	250	20	0,3	149,6	100,3	-32,9 %	
CsAB250	250	20	0,6	88,5	88,3	-0,2 %	
CsBA250	250	20	0,3	113	138,8	+22,8 %	
CsBB250	250	19	0,3	74,4	81,5	+9,5 %	
Μέσος όρος των μέσων όρων	-	-	-	90,9	71,7	-	

Πίνακας 7-14: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 10 στηλών στο RMP του 1D-CSP σε κάθε επανάληψη και με την τιμή της παραμέτρου $\hat{a}$, για κάθε σύνολο παραδειγμάτων).

Μέθοδος:				**CCG**	**MCG** $(\boldsymbol{n = 10})$	**(%) Ποσοστό μεταβολής του μέσου χρόνου επίλυσης**	**Μέσος όρος των ποσοστών μεταβολής του μέσου χρόνου επίλυσης**
Όνομα συνόλου παραδειγμάτων	m	N	$\hat{a}$	Μέσος χρόνος επίλυσης (seconds)	Μέσος χρόνος επίλυσης (seconds)		
CsAA125	125	18	0,3	53,4	74,3	+39,1 %	+19,7 %
CsAB125	125	19	0,6	28,7	42,9	+49,4 %	
CsBA125	125	19	0,3	43,5	53,5	+22,9 %	
CsBB125	125	18	0,3	40,9	40,9	0 %	
HARD	199	10	0,8	226,4	106,8	-52,8 %	
CsAA250	250	20	0,3	149,6	139,7	-6,6 %	
CsAB250	250	20	0,3	88,5	107,4	+21,3 %	
CsBA250	250	20	0,3	113,0	166,9	+47,6 %	
CsBB250	250	19	0,3	74,4	116,7	+56,8 %	
Μέσος όρος των μέσων όρων	-	-	-	90,9	94,3	-	

7.2.3 Αριθμητικά αποτελέσματα των υπολογιστικών πειραμάτων για το 1D-BPP

Πίνακας 7-15: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «t60» του E. Falkenauer [108] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	*m*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
t60_00	60	88	11	61	10	63	7	71	9	48	17	51	20	47	14	29	22	42	27	45	22
t60_01	60	99	19	60	13	66	19	64	15	37	19	40	23	47	25	29	18	26	19	32	27
t60_02	60	77	8	44	8	42	6	46	8	28	11	25	8	30	11	29	17	26	20	30	18
t60_03	60	66	9	64	18	60	14	67	15	35	12	42	17	39	13	34	19	36	25	33	26
t60_04	60	88	10	49	23	48	11	55	21	31	21	34	17	41	21	29	26	27	20	35	32
t60_05	60	86	25	53	13	60	7	46	23	46	57	45	22	49	30	30	22	39	47	48	58
t60_06	60	100	13	51	7	48	5	44	6	30	15	33	17	35	15	26	16	27	31	28	15
t60_07	60	78	11	47	7	41	5	49	7	31	13	28	14	33	27	28	18	26	14	28	14
t60_08	60	80	10	46	8	49	9	53	8	26	54	25	21	25	16	22	77	20	9	21	28
t60_09	60	77	10	47	7	47	14	50	14	24	39	26	11	30	16	34	30	22	13	23	13
t60_10	60	105	15	54	12	48	8	66	10	45	34	38	21	44	29	39	25	40	21	44	23
t60_11	60	92	12	40	12	53	11	58	14	29	15	31	11	36	17	25	21	25	27	31	28
t60_12	60	73	8	46	16	47	14	49	17	26	12	31	12	26	13	22	22	22	16	29	31
t60_13	60	72	9	49	10	44	9	40	7	33	17	29	13	31	11	29	27	23	15	26	29
t60_14	60	78	10	52	16	43	7	45	9	30	10	31	9	32	11	28	31	28	36	34	30
t60_15	60	69	9	51	11	49	10	58	10	25	8	33	28	35	21	24	22	29	39	33	47
t60_16	60	65	8	64	13	66	38	55	109	29	11	39	15	39	15	30	46	37	37	40	36
t60_17	60	84	11	47	48	58	15	50	10	31	10	28	10	41	69	31	21	24	15	33	27
t60_18	60	70	10	56	10	50	8	50	9	28	33	30	20	44	21	26	29	27	21	32	27
t60_19	60	81	12	63	8	67	8	65	10	38	22	42	21	36	13	28	25	34	37	35	29
Μέσος όρος:	60	81,4	11,5	52,2	13,5	52,45	11,25	54,05	16,55	32,5	21,5	34,1	16,5	37	20,4	28,6	26,7	29	24,5	33	28

Πίνακας 7-16: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «t120» του E. Falkenauer [108] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	m	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
t120_00	120	184	39	106	25	100	17	106	16	48	17	51	20	47	14	43	30	49	29	55	32
t120_01	120	169	43	96	17	95	35	95	16	37	19	40	23	47	25	49	37	50	38	58	38
t120_02	120	145	22	93	26	91	16	103	21	28	11	25	8	30	11	48	33	43	50	46	56
t120_03	120	148	22	85	25	102	44	93	37	35	12	42	17	39	13	50	60	47	33	48	34
t120_04	120	164	28	92	25	91	24	106	56	31	21	34	17	41	21	42	28	45	31	55	47
t120_05	120	160	26	86	28	80	39	94	17	46	57	45	22	49	30	51	41	41	30	41	44
t120_06	120	141	28	88	25	87	21	99	33	30	15	33	17	35	15	47	30	55	33	52	32
t120_07	120	163	26	96	22	104	32	101	20	31	13	28	14	33	27	44	36	63	44	68	94
t120_08	120	160	27	101	18	89	15	100	17	26	54	25	21	25	16	40	38	40	44	38	26
t120_09	120	166	38	89	14	106	55	118	27	24	39	26	11	30	16	46	33	39	28	44	33
t120_10	120	180	55	110	24	89	19	95	16	45	34	38	21	44	29	54	42	41	26	49	39
t120_11	120	191	62	106	38	93	79	105	31	29	15	31	11	36	17	62	45	53	32	71	51
t120_12	120	158	25	97	26	103	25	118	25	26	12	31	12	26	13	47	34	49	33	51	37
t120_13	120	164	26	98	24	105	28	100	24	33	17	29	13	31	11	33	54	35	28	44	37
t120_14	120	205	50	99	20	105	37	135	35	30	10	31	9	32	11	47	44	67	53	43	35
t120_15	120	164	26	56	13	59	33	67	15	28	8	33	28	35	21	34	33	39	32	36	29
t120_16	120	166	28	87	17	99	15	103	16	29	11	39	15	39	15	43	39	43	38	45	57
t120_17	120	177	30	92	17	93	18	101	21	31	10	28	10	41	69	61	39	66	88	70	50
t120_18	120	128	32	90	16	92	17	89	17	28	33	30	20	41	21	61	40	50	34	50	50
t120_19	120	178	33	102	22	126	37	130	31	38	22	42	21	36	13	49	34	73	56	65	84
Μέσος όρος:	120	165,55	33,3	93,45	22,1	95,45	30,3	102,9	24,55	32,7	21,5	34,1	16,5	36,9	20,4	47,6	38,5	49,4	39	51,5	45,3

Πίνακας 7-17: Αριθμητικά αποτελέσματα από τη επίλυση των 20 παραδειγμάτων του συνόλου «t249» του E. Falkenauer [108] *με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).*

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
α		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	*m*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
t249_00	249	377	149	179	65	211	84	180	67	92	79	97	92	112	92	84	64	71	85	91	143
t249_01	249	359	192	247	158	233	99	275	262	115	125	128	195	117	155	92	126	96	102	121	202
t249_02	249	478	322	213	104	213	77	215	145	122	94	117	105	127	182	73	109	90	98	103	106
t249_03	249	303	175	168	80	200	129	229	150	112	131	109	213	130	157	89	114	94	79	107	106
t249_04	249	257	205	197	54	201	58	174	46	114	139	82	204	140	142	98	141	76	116	86	124
t249_05	249	312	201	181	109	181	71	192	123	109	60	110	97	129	85	78	85	97	100	121	139
t249_06	249	321	98	201	70	190	60	206	52	128	85	90	150	94	63	87	101	87	77	82	76
t249_07	249	340	254	206	52	207	83	199	50	110	77	138	245	132	154	90	77	100	138	118	134
t249_08	249	322	233	184	79	187	84	199	130	113	138	129	92	146	136	112	123	119	233	134	186
t249_09	249	346	128	181	46	192	43	221	58	127	107	125	209	120	96	111	111	83	156	96	125
t249_10	249	306	168	162	68	173	83	177	71	107	84	103	139	106	161	80	94	85	129	96	116
t249_11	249	357	181	182	76	198	84	214	96	108	154	130	140	139	135	86	110	96	157	116	112
t249_12	249	314	110	193	44	183	63	186	45	112	63	127	74	147	124	85	132	83	68	107	123
t249_13	249	330	153	194	93	190	107	188	89	107	79	115	87	119	80	73	93	91	84	93	136
t249_14	249	331	106	206	54	212	56	220	59	114	105	117	104	143	133	102	101	91	101	115	138
t249_15	249	322	134	179	67	202	102	203	101	119	76	122	131	151	112	112	108	102	109	120	130
t249_16	249	318	148	185	47	172	44	179	201	93	79	110	104	122	88	78	99	87	89	109	102
t249_17	249	323	151	186	91	188	134	218	86	110	90	107	81	122	135	98	114	93	101	108	98
t249_18	249	299	71	188	100	176	139	185	84	100	77	99	97	122	71	77	67	92	105	98	117
t249_19	249	327	233	186	42	201	58	201	78	95	60	97	58	113	89	72	60	80	84	96	114
Μέσος όρος:	249	332,1	170,6	190,9	74,95	195,5	82,9	203,05	99,65	110,4	95,1	112,6	130,9	126,6	119,5	88,9	101,5	90,7	110,6	105,9	126,4

Πίνακας 7-18: Αριθμητικά αποτελέσματα από την επίλυση των 100 παραδειγμάτων του συνόλου «Schwerin 1» των P. Schwerin & G. Wäscher [111] με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

Μέθοδος:		**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$		1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Παράδειγμα	*m*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*	*iter.*	*time*
Schwerin1_BPP1	100	137	39	76	16	78	15	89	16	46	23	44	19	50	19	41	33	40	44	40	20
Schwerin1_BPP2	100	132	56	90	17	69	10	88	21	41	31	41	17	50	36	38	24	37	29	35	22
Schwerin1_BPP3	100	117	38	68	18	66	14	70	14	35	17	37	17	38	15	35	20	28	14	29	17
Schwerin1_BPP4	100	101	14	63	14	60	13	64	12	30	12	27	15	33	19	26	16	25	23	28	149
Schwerin1_BPP5	100	135	35	80	20	75	15	82	18	42	27	41	36	42	21	36	33	39	30	36	24
Schwerin1_BPP6	100	114	15	73	26	69	19	74	21	38	31	37	15	39	17	37	26	38	27	36	28
Schwerin1_BPP7	100	100	13	59	15	57	13	59	13	33	14	29	12	34	32	27	19	25	15	26	15
Schwerin1_BPP8	100	119	15	71	23	77	23	81	14	38	26	41	15	44	19	30	32	36	36	38	37
Schwerin1_BPP9	100	119	25	74	20	73	17	73	20	38	32	38	21	42	25	34	27	35	32	38	46
Schwerin1_BPP10	100	112	31	71	18	66	14	73	25	35	18	33	30	42	19	34	27	29	18	35	18
Schwerin1_BPP11	100	107	22	59	10	61	42	62	11	29	15	29	12	29	12	28	16	26	12	28	16
Schwerin1_BPP12	100	108	23	65	14	65	14	66	16	31	13	28	9	33	11	27	21	26	15	27	15
Schwerin1_BPP13	100	111	50	64	13	64	9	71	11	29	12	31	11	33	12	27	20	27	17	26	14
Schwerin1_BPP14	100	114	35	62	27	60	8	66	8	32	11	31	12	30	15	29	18	27	16	31	35
Schwerin1_BPP15	100	109	43	62	10	67	11	67	20	36	15	35	14	39	16	28	31	27	28	30	24
Schwerin1_BPP16	100	89	24	51	9	52	8	56	9	28	11	27	9	27	9	24	25	26	16	25	24
Schwerin1_BPP17	100	114	28	70	14	63	10	70	13	30	13	34	17	39	23	28	30	27	17	25	14
Schwerin1_BPP18	100	112	35	65	13	64	13	74	19	31	19	30	14	38	19	30	24	28	19	27	17
Schwerin1_BPP19	100	108	18	67	23	66	16	70	17	30	16	32	17	36	20	28	24	26	19	28	17
Schwerin1_BPP20	100	109	23	69	21	63	14	71	14	34	22	33	16	33	19	29	22	24	13	30	17
Schwerin1_BPP21	100	110	15	71	16	66	15	72	12	34	17	35	13	35	16	29	23	29	12	33	18
Schwerin1_BPP22	100	109	27	68	25	68	11	69	20	38	25	31	20	31	13	30	18	29	15	31	17
Schwerin1_BPP23	100	108	19	63	15	64	17	67	13	30	14	35	15	34	14	27	20	27	14	31	19
Schwerin1_BPP24	100	113	26	68	16	63	12	69	13	33	15	35	13	32	20	28	21	27	29	36	31
Schwerin1_BPP25	100	110	25	69	13	62	11	74	16	31	27	32	10	35	12	27	19	26	21	21	17
Schwerin1_BPP26	100	105	29	60	13	61	9	65	10	31	11	27	8	30	10	27	19	26	13	27	16
Schwerin1_BPP27	100	108	20	60	8	64	9	69	12	33	12	30	12	37	17	30	19	27	20	27	15
Schwerin1_BPP28	100	107	63	66	18	67	15	70	13	37	15	35	13	45	18	31	29	34	20	32	36
Schwerin1_BPP29	100	109	18	61	12	68	17	76	14	34	13	33	13	33	12	28	18	29	17	34	20
Schwerin1_BPP30	100	116	20	67	16	64	13	74	36	31	14	37	20	37	24	32	21	35	28	37	21
Schwerin1_BPP31	100	112	36	67	21	65	22	68	18	30	21	30	15	34	17	27	17	26	41	30	12
Schwerin1_BPP32	100	111	33	66	13	65	11	71	31	30	13	28	13	38	17	28	15	26	11	26	16
Schwerin1_BPP33	100	109	16	64	15	64	13	75	14	33	21	31	17	3	15	26	18	29	32	30	21

Schwerin1_BPP34	100	107	12	67	14	68	14	69	12	33	16	33	14	39	16	33	37	33	19	32	16
Schwerin1_BPP35	100	106	12	63	12	66	11	67	12	38	25	31	14	31	12	25	23	30	27	30	33
Schwerin1_BPP36	100	112	11	65	14	62	12	75	61	34	21	32	14	36	18	28	22	28	17	32	17
Schwerin1_BPP37	100	112	10	64	15	66	11	69	21	29	14	30	13	37	19	28	20	27	16	30	25
Schwerin1_BPP38	100	109	34	69	20	66	14	70	23	35	17	30	9	38	15	32	20	31	19	32	23
Schwerin1_BPP39	100	109	14	65	15	69	12	76	17	36	12	32	11	35	12	28	21	30	19	31	23
Schwerin1_BPP40	100	104	13	56	10	61	15	58	12	29	10	31	13	31	15	30	36	30	21	33	32
Schwerin1_BPP41	100	118	15	67	18	67	18	74	13	34	14	34	13	35	17	32	21	26	11	33	17
Schwerin1_BPP42	100	119	15	62	13	69	15	70	19	34	13	31	12	35	15	27	12	29	18	30	17
Schwerin1_BPP43	100	108	10	67	16	66	16	74	27	34	13	33	15	37	23	33	23	31	15	34	20
Schwerin1_BPP44	100	112	16	65	52	66	13	66	18	32	14	31	13	37	36	28	30	28	23	31	18
Schwerin1_BPP45	100	109	14	69	30	69	23	68	20	33	15	30	33	32	48	28	26	28	26	29	20
Schwerin1_BPP46	100	108	18	71	24	66	22	73	16	32	61	32	24	35	21	30	32	30	17	29	19
Schwerin1_BPP47	100	110	20	66	20	68	21	69	25	34	17	35	15	35	16	32	29	31	20	36	20
Schwerin1_BPP48	100	108	143	64	27	67	17	73	27	29	14	35	11	34	16	27	16	32	19	31	21
Schwerin1_BPP49	100	116	23	64	16	64	16	73	16	34	16	32	20	34	14	29	21	27	16	30	24
Schwerin1_BPP50	100	108	27	67	22	67	30	68	15	32	16	30	13	31	11	26	21	25	27	28	21
Schwerin1_BPP51	100	111	61	67	22	68	16	72	19	32	29	37	13	36	13	33	18	31	20	36	17
Schwerin1_BPP52	100	113	29	64	16	64	12	68	22	34	11	32	11	35	14	29	15	27	13	27	17
Schwerin1_BPP53	100	113	42	63	13	62	8	67	12	36	12	34	15	36	19	30	34	30	21	30	17
Schwerin1_BPP54	100	111	28	67	13	63	11	68	21	33	27	29	13	32	14	28	21	25	15	29	56
Schwerin1_BPP55	100	113	24	66	22	69	18	73	17	35	29	39	19	37	23	31	40	30	29	34	27
Schwerin1_BPP56	100	116	22	66	15	65	13	73	12	32	27	30	16	33	15	28	24	26	13	29	17
Schwerin1_BPP57	100	107	22	62	10	68	18	71	14	35	22	34	19	39	24	31	26	28	15	30	27
Schwerin1_BPP58	100	118	31	66	26	66	23	75	93	33	20	34	16	30	16	26	24	26	14	28	18
Schwerin1_BPP59	100	104	15	66	21	65	15	69	15	31	12	34	14	36	14	27	19	27	20	30	19
Schwerin1_BPP60	100	113	15	68	25	66	21	73	14	34	14	35	18	37	11	34	35	35	21	37	23
Schwerin1_BPP61	100	115	26	66	14	66	11	77	24	33	17	32	14	36	19	29	20	32	14	31	16
Schwerin1_BPP62	100	110	15	73	15	66	15	72	12	34	19	35	15	39	20	32	16	29	14	33	18
Schwerin1_BPP63	100	110	20	60	29	64	99	72	33	37	18	31	13	34	18	29	25	31	18	26	15
Schwerin1_BPP64	100	114	28	69	32	63	17	69	18	33	16	31	12	32	11	27	14	29	16	33	20
Schwerin1_BPP65	100	115	20	68	15	64	17	72	19	34	12	34	27	32	9	31	27	27	23	30	50
Schwerin1_BPP66	100	117	20	59	9	65	31	75	14	31	11	31	15	35	23	27	19	27	19	27	22
Schwerin1_BPP67	100	113	16	68	12	69	11	71	12	33	33	35	15	37	16	29	20	33	21	30	19
Schwerin1_BPP68	100	111	26	66	15	66	23	70	26	34	18	40	35	38	16	30	23	32	19	33	21
Schwerin1_BPP69	100	109	19	64	21	64	18	65	15	35	18	31	15	33	14	28	31	28	17	30	25
Schwerin1_BPP70	100	114	30	66	16	65	29	70	28	29	14	34	19	37	23	27	31	25	15	31	19
Schwerin1_BPP71	100	109	27	67	21	65	25	69	16	29	17	33	21	34	18	27	17	25	10	30	15
Schwerin1_BPP72	100	108	38	66	14	63	10	74	19	36	37	37	15	35	15	27	15	34	16	32	17
Schwerin1_BPP73	100	108	24	63	19	64	26	68	24	31	19	30	15	38	24	28	22	27	36	29	18
Schwerin1_BPP74	100	107	20	64	29	65	17	75	18	32	15	32	21	34	28	31	24	30	19	31	20

Schwerin1_BPP75	100	108	25	64	18	65	15	67	15	31	22	30	18	33	21	27	41	31	17	26	15
Schwerin1_BPP76	100	114	28	68	28	67	18	72	15	37	27	33	25	40	34	35	25	30	16	33	23
Schwerin1_BPP77	100	107	28	65	18	64	16	75	26	29	14	35	42	35	13	27	22	27	24	32	18
Schwerin1_BPP78	100	116	29	64	11	60	15	68	15	33	14	31	14	33	13	27	17	25	16	29	17
Schwerin1_BPP79	100	110	33	64	14	65	12	66	11	30	13	30	12	36	14	25	31	28	21	31	22
Schwerin1_BPP80	100	110	19	59	9	54	8	71	10	31	31	28	47	35	45	27	21	26	14	27	17
Schwerin1_BPP81	100	103	24	67	12	62	18	73	18	32	17	32	15	34	15	27	16	27	13	29	15
Schwerin1_BPP82	100	108	24	66	16	65	14	73	14	29	12	30	14	31	15	27	15	27	13	31	16
Schwerin1_BPP83	100	108	17	66	16	65	12	68	15	32	14	31	21	38	17	34	28	28	26	31	33
Schwerin1_BPP84	100	112	27	65	14	65	12	69	25	31	30	36	22	36	18	31	18	30	16	34	18
Schwerin1_BPP85	100	113	27	66	73	64	25	69	20	36	24	31	23	36	16	28	27	26	20	28	20
Schwerin1_BPP86	100	111	19	61	23	65	13	68	17	36	28	32	28	41	51	29	22	26	19	32	21
Schwerin1_BPP87	100	109	21	63	17	67	17	65	18	34	23	33	20	40	27	27	19	29	21	33	31
Schwerin1_BPP88	100	107	30	66	21	70	29	71	15	34	17	31	18	38	22	27	18	28	14	32	19
Schwerin1_BPP89	100	117	32	67	22	62	13	68	29	31	29	32	16	32	17	27	22	28	20	28	22
Schwerin1_BPP90	100	116	23	69	14	68	20	71	19	32	16	31	12	39	12	29	20	26	14	26	17
Schwerin1_BPP91	100	111	22	65	31	65	16	72	15	35	22	32	11	35	12	28	20	26	10	34	17
Schwerin1_BPP92	100	109	15	64	12	64	10	72	12	36	16	32	26	32	15	27	13	27	12	28	13
Schwerin1_BPP93	100	111	15	63	11	70	10	74	14	34	17	32	16	35	15	29	24	27	15	31	17
Schwerin1_BPP94	100	108	17	65	10	63	10	71	12	32	14	32	24	35	98	28	21	28	16	30	19
Schwerin1_BPP95	100	114	18	69	10	65	9	71	11	38	97	35	19	40	22	29	21	31	27	31	23
Schwerin1_BPP96	100	110	50	65	26	69	16	69	13	38	19	34	15	35	13	33	32	31	19	30	18
Schwerin1_BPP97	100	115	20	71	17	68	15	76	14	33	16	30	50	34	36	29	19	32	19	34	21
Schwerin1_BPP98	100	111	47	68	13	67	12	74	43	34	26	33	19	38	25	36	24	30	20	33	19
Schwerin1_BPP99	100	112	26	60	28	67	24	73	14	32	20	34	20	34	13	31	28	33	18	36	21
Schwerin1_BPP100	100	110	27	66	13	67	17	71	19	35	20	34	21	36	19	29	29	29	19	32	26
Μέσος όρος:	100	111,28	26,17	65,9	18,21	65,41	16,53	70,82	18,54	33,34	19,69	32,77	17,41	35,43	19,42	29,34	23,08	28,92	19,4	30,81	22,45

Πίνακας 7-19: Συγκεντρωτικός πίνακας των αριθμητικών αποτελεσμάτων των παραδειγμάτων δοκιμής του 1D-BPP που χρησιμοποιήθηκαν, με την CCG (προσθήκη 1 στήλης στο RMP σε κάθε επανάληψη) και με την προτεινόμενη MCG (προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, για τρεις διαφορετικές τιμές της παραμέτρου α).

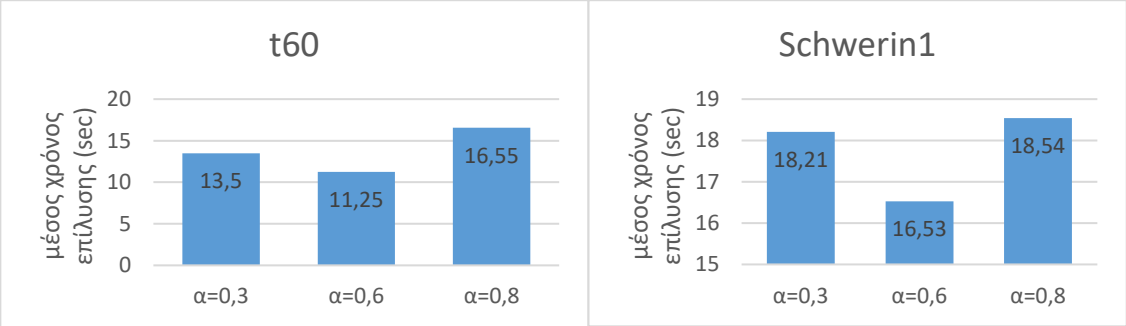
Μέθοδος:			**CCG**		**MCG** $(n=2)$						**MCG** $(n=6)$						**MCG** $(n=10)$					
$\boldsymbol{a}$			1		0,3		0,6		0,8		0,3		0,6		0,8		0,3		0,6		0,8	
Όνομα συνόλου παραδειγμάτων	m	N	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*	*Av.iter*	*AST*
t60	60	20	81,4	11,5	52,2	13,5	52,45	11,25	54,05	16,55	32,5	21,5	34,1	16,5	37	20,4	28,6	26,7	29	24,5	33	28
Schwerin1	100	100	111,28	26,17	65,9	18,21	65,41	16,53	70,82	18,54	33,34	19,69	32,77	17,41	35,43	19,42	29,34	23,08	28,92	19,4	30,81	22,45
t120	120	20	165,55	33,3	93,45	22,1	95,45	30,3	102,9	24,55	32,7	21,5	34,1	16,5	36,9	20,4	47,6	38,5	49,4	39	51,5	45,3
t249	249	20	332,1	170,6	190,9	74,95	195,5	82,9	203,05	99,65	110,4	95,1	112,6	130,9	126,6	119,5	88,9	101,5	90,7	110,6	105,9	126,4
συνολικά	-	160	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

7.2.4 Σχολιασμός των αριθμητικών αποτελεσμάτων των υπολογιστικών πειραμάτων για το 1D-BPP

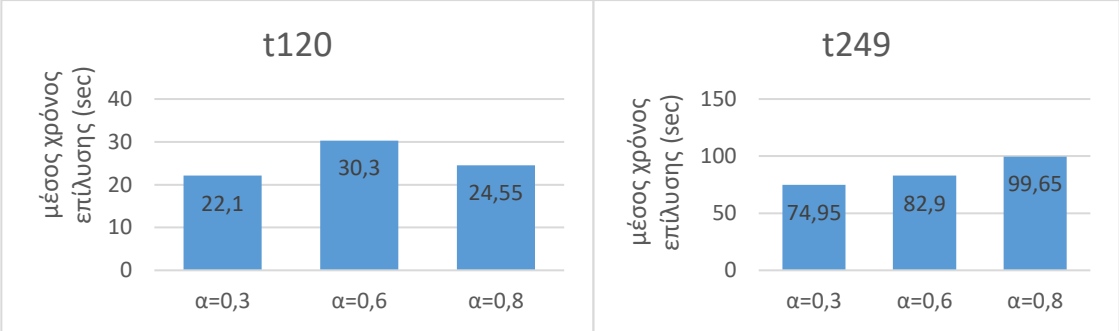
Από τα αποτελέσματα του **Πίνακας 7-19**, μπορούν να σημειωθούν οι ακόλουθες παρατηρήσεις:

Για την παράμετρο a:

- Ο τρόπος με τον οποίο οι παράμετροι a και $n = k + 1$, επηρεάζουν τον χρόνο επίλυσης και τον αριθμό των επαναλήψεων, εξαρτάται από το πρόβλημα που αντιμετωπίζεται κάθε φορά.
- Όπως σημειώθηκε στην **§ 5.3.1.1**, για την κλασική μέθοδο δημιουργίας στήλης ισχύει $a = 1$. Η μείωση της τιμής μειώνει τον μέσο αριθμό των επαναλήψεων και τον μέσο χρόνο επίλυσης.
- Ο ελάχιστος μέσος χρόνος επίλυσης για την προτεινόμενη μέθοδο MCG με την προσθήκη 2, 6 και 10 στηλών στο RMP σε κάθε επανάληψη, επιτυγχάνεται στα περισσότερα προβλήματα είτε για $a = 0{,}3$ είτε για $a = 0{,}6$ με μικρές διαφορές. Εξαίρεση αποτελεί η περίπτωση του συνόλου «HARD10» όπου με την προσθήκη 10 στηλών η βέλτιστη τιμή είναι $a = 0{,}8$.
- Όσο αυξάνεται ο αριθμός των προστιθέμενων στηλών, φαίνεται ότι ο ελάχιστος μέσος χρόνος επίλυσης, στα περισσότερα σύνολα παραδειγμάτων επιτυγχάνεται για $a = 0{,}3$.



(α) (β)

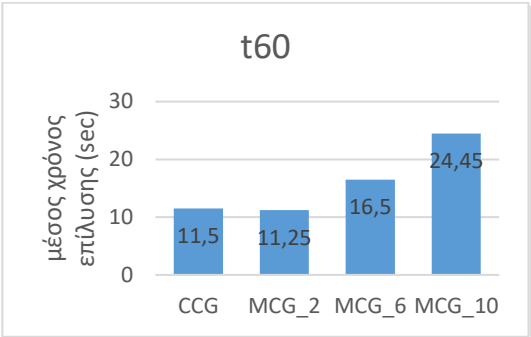


(γ) (δ)

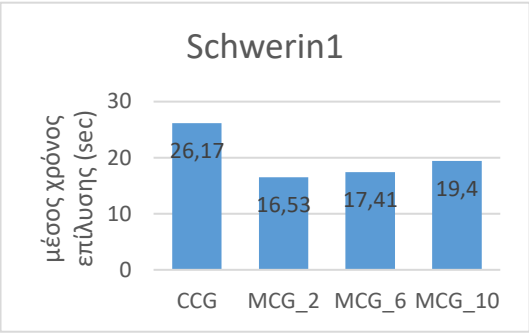
Εικόνα 7-4: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) t60, (β) Schwerin1, (γ) t120 και (δ) t249, για 3 διαφορετικές τιμές της παραμέτρου επίπεδου εξομάλυνσης α.

<u>Για την παράμετρο k:</u>

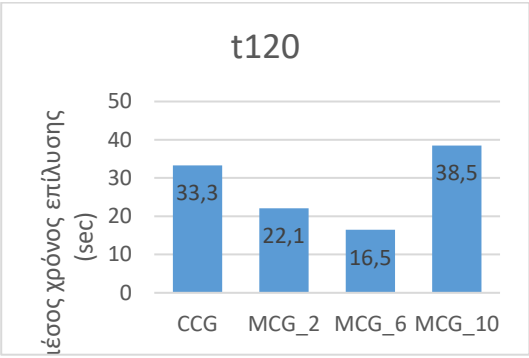
- Ο ελάχιστος μέσος χρόνος επίλυσης των περισσότερων εξεταζόμενων παραδειγμάτων δοκιμής επιτυγχάνεται με την προσθήκη δύο στηλών στο RMP σε κάθε επανάληψη αλλά και με την προσθήκη 6 στηλών σε κάθε επανάληψη (σύνολο «t120»).
- Όσο αυξάνεται ο αριθμός των στηλών $n > 6$, που προστίθενται στο RMP, τόσο αυξάνεται και ο μέσος χρόνος επίλυσης. Αυτό φαίνεται από τα αποτελέσματα με προσθήκη 10 στηλών σε κάθε επανάληψη και στους πίνακες: **Πίνακας 7-20**, **Πίνακας 7-21** και στον **Πίνακας 7-22**. Δηλαδή, η αύξηση του αριθμού των προστιθέμενων στηλών $n > 6$, «υπερφορτώνει» το RMP.
- Η προτεινόμενη μέθοδος λειτουργεί καλύτερα για μεγάλο αριθμό περιορισμών, m.



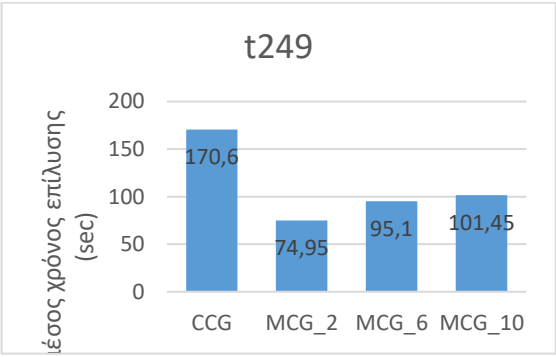
(α)



(β)



(γ)



(δ)

Εικόνα 7-5: Παράδειγμα επηρεασμού του μέσου χρόνου επίλυσης των συνόλων παραδειγμάτων δοκιμής (α) t60, (β) Schwerin1, (γ) t120 και (δ) t249 με την κλασική μέθοδο δημιουργίας στήλης (CCG) και της μεθόδου δημιουργίας πολλαπλών στηλών, με την προσθήκη 2 (MCG_2), 6 (MCG_6) και 10 (MCG_10) στηλών στο RMP σε κάθε επανάληψη αντίστοιχα, για την τιμή της παραμέτρου â.

Πίνακας 7-20: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 2 στηλών στο RMP του 1D-BPP σε κάθε επανάληψη και με την τιμή της παραμέτρου $\hat{a}$ για κάθε σύνολο παραδειγμάτων).

Μέθοδος:				**CCG**	**MCG** $(\boldsymbol{n = 2})$	**(%) Ποσοστό μεταβολής του μέσου χρόνου επίλυσης**	**Μέσος όρος των ποσοστών μεταβολής του μέσου χρόνου επίλυσης**
Όνομα συνόλου παραδειγμάτων	m	N	$\hat{a}$	Μέσος χρόνος επίλυσης (seconds)	Μέσος χρόνος επίλυσης (seconds)		
t60	60	20	0,6	11,5	11,2	-2,17 %	-32,1 %
Schwerin1	100	100	0,6	26,1	16,5	-36,8 %	
t120	120	20	0,3	33,3	22,1	-33,6 %	
t249	249	20	0,3	170,6	74,9	-56,1 %	
Μέσος όρος των μέσων όρων	-	-	-	60,3	31,1	-	

Πίνακας 7-21: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 6 στηλών στο RMP του 1D-BPP σε κάθε επανάληψη και με την τιμή της παραμέτρου $\hat{a}$ για κάθε σύνολο παραδειγμάτων).

Μέθοδος:				**CCG**	**MCG** $(\boldsymbol{n = 6})$	**(%) Ποσοστό μεταβολής του μέσου χρόνου επίλυσης**	**Μέσος όρος των ποσοστών μεταβολής του μέσου χρόνου επίλυσης**
Όνομα συνόλου παραδειγμάτων	m	N	$\hat{a}$	Μέσος χρόνος επίλυσης (seconds)	Μέσος χρόνος επίλυσης (seconds)		
t60	60	20	0,6	11,5	16,5	+43,4 %	-21,1 %
Schwerin1	100	100	0,6	26,1	17,4	-33,4 %	
t120	120	20	0,6	33,3	16,5	-50,4 %	
t249	249	20	0,3	170,6	95,1	-44,2 %	
Μέσος όρος των μέσων όρων	-	-	-	60,3	36,3	-	

Πίνακας 7-22: Σύγκριση της κλασικής μεθόδου δημιουργίας στήλης-προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών (με την προσθήκη 10 στηλών στο RMP του 1D-BPP σε κάθε επανάληψη και με την τιμή της παραμέτρου $\hat{a}$ για κάθε σύνολο παραδειγμάτων).

Μέθοδος:				**CCG**	**MCG** $(\boldsymbol{n = 10})$	**(%) Ποσοστό μεταβολής του μέσου χρόνου επίλυσης**	**Μέσος όρος των ποσοστών μεταβολής του μέσου χρόνου επίλυσης**
Όνομα συνόλου παραδειγμάτων	m	N	$\hat{a}$	Μέσος χρόνος επίλυσης (seconds)	Μέσος χρόνος επίλυσης (seconds)		
t60	60	20	0,6	11,5	24,5	+113,0 %	+15,5 %
Schwerin1	100	100	0,6	26,1	19,4	-25,8 %	
t120	120	20	0,3	33,3	38,5	+15,6 %	
t249	249	20	0,3	170,6	101,5	-40,5 %	
Μέσος όρος των μέσων όρων	-	-	-	60,3	45,9	-	

<u>Παρατήρηση που αφορά τα 1D-CSP & 1D-BPP:</u>

Σε κάθε επανάληψη της μεθόδου δημιουργίας πολλαπλών στηλών που προτείνεται, μπορούν να προστεθούν μέχρι k στήλες (μπορεί και λιγότερες) και όχι ακριβώς k. Αυτό συμβαίνει γιατί όταν βρεθεί μια στήλη, προστίθεται στο RMP μόνο όταν έχει αρνητικό μειωμένο κόστος σε σχέση με την τρέχουσα δυϊκή λύση.

Κεφάλαιο 8. Συμπεράσματα-προτάσεις

8.1 Συμπεράσματα της διπλωματικής εργασίας

Σε αυτή τη διπλωματική εργασία μελετήθηκε η μέθοδος μαθηματικής αποσύνθεσης Dantzig-Wolfe για προβλήματα γραμμικού προγραμματισμού. Η μέθοδος εφαρμόστηκε σε δύο προβλήματα μαθηματικής μοντελοποίησης: (α) στο 1D-CSP και (β) στο 1D-BPP. Στη συνέχεια, σημειώθηκε ότι η DWD επιταχύνεται, με επέμβαση στη μέθοδο δημιουργίας στήλης που αυτή χρησιμοποιεί. Αναφέρθηκαν διάφορες τεχνικές επιτάχυνσης της μεθόδου δημιουργίας στήλης. Ωστόσο, η εργασία, επικεντρώθηκε στην ανάπτυξη μίας νέας μεθόδου δημιουργίας πολλαπλών στηλών, όπου η μία στήλη παράγεται από την κλασική μέθοδο και οι άλλες k στήλες παράγονται με κυρτό συνδυασμό προηγούμενων δυϊκών λύσεων από τροποποιημένο/-α υπο-πρόβλημα/-ατα.

Για τη σύγκριση της απόδοσης της κλασικής μεθόδου δημιουργίας στήλης και της προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών, επιλύθηκαν τα απλοποιημένα μοντέλα Gilmore-Gomory, χρησιμοποιώντας 163 και 160 παραδείγματα δοκιμών της BPPLIB για το 1D-CSP και το 1D-BPP αντίστοιχα.

Από τα αποτελέσματα που παρουσιάστηκαν στο **Κεφάλαιο 7**, συνοπτικά, συμπεραίνονται τα ακόλουθα:

- Η μείωση της τιμής της παραμέτρου επίπεδου εξομάλυνσης $a \in (0,1)$, μειώνει τον αριθμό των επαναλήψεων.
- Η τιμή της παραμέτρου επίπεδου εξομάλυνσης a, για την οποία επιτυγχάνονται καλύτερα αποτελέσματα στα περισσότερα προβλήματα του 1D-CSP και του 1D-BPP που επιλύθηκαν είναι 0,3 ή 0,6.
- Η μέθοδος δημιουργίας πολλαπλών στηλών που προτείνεται, κρίνεται αποδοτικότερη με την προσθήκη **2 μέχρι και 6 στηλών** στο RMP σε κάθε επανάληψη.
- Η προτεινόμενη μέθοδος φαίνεται να έχει μικρή ανεκτικότητα σε προσθήκη στηλών με αριθμό $n > 6$, που προστίθενται σε κάθε επανάληψη στο RMP («υπερφόρτωση» του RMP).
- Η προτεινόμενη μέθοδος λειτουργεί καλύτερα για αριθμό περιορισμών, $m \geq 100$.

- Σε κάθε επανάληψη της προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών, μπορούν να προστεθούν μέχρι k επιπλέον στήλες (μπορεί και λιγότερες) και όχι ακριβώς k. Αυτό συμβαίνει γιατί όταν βρεθεί μια στήλη, προστίθεται στο RMP μόνο όταν έχει αρνητικό μειωμένο κόστος σε σχέση με την τρέχουσα δυϊκή λύση.

8.2 Προτάσεις για μελλοντική εργασία

Η μέθοδος δημιουργίας πολλαπλών στηλών που προτάθηκε στην παρούσα εργασία, εξετάστηκε με τη χρήση παραδειγμάτων δοκιμής. Σημαντική θα ήταν η εφαρμογή της με τη χρήση αληθινών δεδομένων, επιτρέποντας έτσι την αξιολόγηση της χρησιμότητάς της και τον εντοπισμό των αδυναμιών της.

Επίσης, η επίλυση του υπο-προβλήματος, που είναι το πρόβλημα σακιδίου και για τα δύο προβλήματα 1D-CDP & 1D-BPP, έγινε με τη χρήση της μεθόδου simplex (από το CPLEX). Για πιθανή βελτίωση των αποτελεσμάτων, κάτι στο οποίο δεν δόθηκε ιδιαίτερο βάρος, θα μπορούσε να χρησιμοποιηθεί κάποιος αλγόριθμος επιτάχυνσης του προβλήματος σακιδίου.

Στην εργασία επίσης έγινε η παραδοχή ότι όλοι οι κύλινδροι (για το 1D-CSP) και όλοι οι κάδοι (για το 1D-BPP) έχουν το ίδιο πλάτος και επομένως, χρησιμοποιήθηκε το απλοποιημένο μοντέλο των Gilmore-Gomory. Θα μπορούσε να εξεταστεί η απόδοση της μεθόδου δημιουργίας πολλαπλών στηλών που αναπτύχθηκε στην γενική περίπτωση διαφορετικού πλάτους των κυλίνδρων (για το 1D-CSP) και των κάδων (για το 1D-BPP).

Επιπλέον, θα μπορούσε να εξεταστεί η εφαρμογή της προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών, στην υβριδοποιημένη μέθοδο δημιουργίας στήλης - μεθόδου Branch and Bound (μέθοδος διακλάδωσης και τιμής, Branch and Price) για βελτίωση της απόδοσής της.

Τέλος, μία άλλη πρόταση θα ήταν η σύγκριση της προτεινόμενης μεθόδου δημιουργίας πολλαπλών στηλών με άλλες μεθόδους επιτάχυνσης της κλασικής μεθόδου δημιουργίας στήλης.

Βιβλιογραφία

[1] H. Mohammadi and E. Khorram, “A New Combination of Lagrangean Relaxation, Dantzig-Wolfe Decomposition and Benders Decomposition Methods for Exact Solution of the Mixed Integer Programming Problems,” *undefined*, vol. 10, no. 3, Sep. 2019, doi: 10.26713/CMA.V10I3.1116.

[2] U. Ngulo, “Decomposition Methods for Combinatorial Optimization Uledi Ngulo,” Linköping University, Linköping, Sweden, 2021.

[3] “The BIGGEST O of them all (notation) by Jamon Dixon | Medium.” https://jamondixon.medium.com/the-biggest-o-of-them-all-notation-93352775acc7 (accessed May 07, 2022).

[4] M. Galati, F. Barahona, J. C. Hartman, and J. T. Linderoth, “DECOMPOSITION METHODS FOR INTEGER LINEAR PROGRAMMING,” 2009.

[5] J. Ostrowski, “SYMMETRY IN INTEGER PROGRAMMING,” Lehigh University, 2008.

[6] K. Hadj Salem and Y. Kieffer, “An Experimental Study on Symmetry Breaking Constraints Im-pact for the One-Dimensional Bin-Packing Problem,” pp. 317–326, 2020, doi: 10.15439/2020f19ï.

[7] “Decomposition (computer science) - Wikipedia.” https://en.wikipedia.org/wiki/Decomposition_(computer_science) (accessed Mar. 13, 2022).

[8] S. Boyd, L. Xiao, and A. Mutapcic, “Notes on Decomposition Methods,” *Stanford University*, Oct. 2003.

[9] S. Boyd, L. Xiao, and A. Mutapcic, “Notes on Decomposition Methods,” Oct. 2003.

[10] M. Fischetti, D. Salvagnin, A. Zanette, M. Fischetti, D. Salvagnin, and A. Zanette, “Minimal Infeasible Subsystems and Benders cuts”.

[11] Jr. J. E. Kelley, “The Cutting-Plane Method for Solving Convex Programs,” *http://dx.doi.org/10.1137/0108053*, vol. 8, no. 4, pp. 703–712, Jul. 2006, doi: 10.1137/0108053.

[12] “Dantzig–Wolfe decomposition - Wikipedia.” https://en.wikipedia.org/wiki/Dantzig%E2%80%93Wolfe_decomposition (accessed Mar. 25, 2022).

[13] "Computer Pioneers - George Bernard Dantzig." https://history.computer.org/pioneers/dantzig.html (accessed Feb. 20, 2022).

[14] "Philip Starr Wolfe (1927–2016) ORMS Today." https://pubsonline.informs.org/do/10.1287/orms.2017.02.21in/full/ (accessed Feb. 20, 2022).

[15] G. B. Dantzig and P. Wolfe, "Decomposition Principle for Linear Programs," *Oper Res*, vol. 8, no. 1, pp. 101–111, Feb. 1960, doi: 10.1287/opre.8.1.101.

[16] G. B. Dantzig and P. Wolfe, "The Decomposition Algorithm for Linear Programs," *Econometrica*, vol. 29, no. 4, p. 767, Oct. 1961, doi: 10.2307/1911818.

[17] J. M. Valério De Carvalho, "Decomposition Methods for Integer Programming," 2010.

[18] J. R. Tebboth, "A Computational Study of Dantzig-Wolfe Decomposition," 2001.

[19] "Janet Lowe - INFORMS." https://www.informs.org/Recognizing-Excellence/Award-Recipients/Janet-Lowe (accessed Feb. 27, 2022).

[20] "BPPLIB – A Bin Packing Problem Library Operations Research Group Bologna." http://or.dei.unibo.it/library/bpplib (accessed Jul. 16, 2022).

[21] E. M. L. Beale, P. A. B. Hughes, and R. E. Small, "Experiences in Using a Decomposition Program.," *Comput. J.*, vol. 8, no. 1, pp. 13–18, Apr. 1965, doi: 10.1093/comjnl/8.1.13.

[22] H. P. Williams and A. C. Redwood, "STRUCTURED LINEAR PROGRAMMING MODEL IN THE FOOD INDUSTRY.," *Oper Res Q*, vol. 25, no. 4, pp. 517–527, 1974, doi: 10.2307/3008086.

[23] J. K. Ho and R. P. Sundarraj, "DECOMP: an Implementation of Dantzig-Wolfe Decomposition for Linear Programming," vol. 338, 1989, doi: 10.1007/978-1-4684-9397-9.

[24] J. Rios, "Algorithm 928: A general, parallel implementation of Dantzig-Wolfe decomposition," *ACM Trans. Math. Softw*, vol. 39, 2013, doi: 10.1145/2450153.2450159.

[25] F. Vanderbeck and L. A. Wolsey, "Reformulation and Decomposition of Integer Programs".

[26] G. Gamrath and M. E. Lübbecke, "Experiments with a Generic Dantzig-Wolfe Decomposition for Integer Programs," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6049 LNCS, pp. 239–252, 2010, doi: 10.1007/978-3-642-13193-6_21.

[27] F. Vanderbeck and M. W. P. Savelsbergh, "A generic view of Dantzig–Wolfe decomposition in mixed integer programming," *Operations Research Letters*, vol. 34, no. 3, pp. 296–306, May 2006, doi: 10.1016/J.ORL.2005.05.009.

[28] R. Jans, “Classification of Dantzig–Wolfe reformulations for binary mixed integer programming problems,” *Eur J Oper Res*, vol. 204, no. 2, pp. 251–254, Jul. 2010, doi: 10.1016/J.EJOR.2009.11.014.

[29] B. Kallehauge, J. Larsen, O. B. G. Madsen, and M. M. Solomon, “Chapter 3 VEHICLE ROUTING PROBLEM W I T H TIME WINDOWS”.

[30] T. Wu, Z. Shi, Z. Liang, X. Zhang, and C. Zhang, “Dantzig-Wolfe decomposition for the facility location and production planning problem,” *Comput Oper Res*, vol. 124, Dec. 2020, doi: 10.1016/J.COR.2020.105068.

[31] S. Gélinas and F. Soumis, “Dantzig-Wolfe decomposition for job shop scheduling,” *Column Generation*, pp. 271–302, 2005, doi: 10.1007/0-387-25486-2_10.

[32] G. Desaulniers, J. Desrosiers, M. Gamache, and F. Soumis, “Crew Scheduling in Air Transportation,” *Fleet Management and Logistics*, pp. 169–185, 1998, doi: 10.1007/978-1-4615-5755-5_8.

[33] J. Rios and K. Rosst, “Massively Parallel Dantzig-Wolfe Decomposition Applied to Traffic Flow Scheduling,” 2009.

[34] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen, “Stabilized column generation,” *Discrete Math*, vol. 194, no. 1–3, pp. 229–237, Jan. 1999, doi: 10.1016/S0012-365X(98)00213-1.

[35] O. Briant, C. Lemaréchal, S. Meurdesoif, N. Michel, and F. Perrot, “Comparison of Bundle and Classical Column Generation,” 2006, Accessed: Mar. 25, 2022. [Online]. Available: https://hal.inria.fr/inria-00070554

[36] M. E. Lübbecke, “Column Generation,” 2010.

[37] C. Lee and S. Park, “Chebyshev center-based column generation,” *Discrete Appl Math (1979)*, vol. 159, no. 18, pp. 2251–2265, Dec. 2011, doi: 10.1016/J.DAM.2011.08.009.

[38] B. Betrò, “An accelerated central cutting plane algorithm for linear semi-infinite programming,” *Mathematical Programming 2004 101:3*, vol. 101, no. 3, pp. 479–495, Jul. 2004, doi: 10.1007/S10107-003-0492-5.

[39] A. Frangioni and B. Gendron, “A Stabilized Structured Dantzig-Wolfe Decomposition Method,” 2000.

[40] P. Wentges, “Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming,” *International Transactions in Operational Research*, vol. 4, no. 2, pp. 151–162, Mar. 1997, doi: 10.1016/S0969-6016(97)00001-4.

[41] P. J. Neame, “Nonsmooth Dual Methods in Integer Programming - Philip James Neame - Google Books,” University of Melbourne, 1999. Accessed: Apr. 29, 2022. [Online]. Available: https://books.google.gr/books/about/Nonsmooth_Dual_Methods_in_Integer_Progra.html?id=WxdPNAAACAAJ&redir_esc=y

[42] Mahey P., “A subgradient algorithm for accelerating the Dantzig-Wolfe decomposition method,”1985. https://www.researchgate.net/publication/239063439_A_subgradient_algorithm_for_accelerating_the_Dantzig-Wolfe_decomposition_method (accessed Feb. 21, 2022).

[43] J. M. Valério De Carvalho, “Using Extra Dual Cuts to Accelerate Column Generation,” *https://doi.org/10.1287/ijoc.1030.0060,* vol. 17, no. 2, pp. 175–182, May 2005, doi: 10.1287/IJOC.1030.0060.

[44] N. Touati Moungla, L. Létocart, A. Nagih, N. Touati, and N. L. Touati Létocart A Nagih, “Solutions diversification in a column generation algorithm,” no. 5, Sep. 2010, Accessed: Jun. 14, 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00522354

[45] F. Castaño, M. Sevaux, and F. Castaño, “Combining Metaheuristics with Column Generation: Successful Approaches to Enhance Column Generation Algorithms Performance,” pp. 95–100, 2016, Accessed: Jul. 02, 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01367946

[46] “Scopus-Document-search.” https://www.scopus.com/search/form.uri?zone=TopNavBar&origin=&display=basic#basic (accessed Aug. 27, 2022).

[47] “Analyze Results.” https://www.webofscience.com/wos/woscc/analyze-results/20f24fae-aa10-49d4-9533-3ede768d28d9-5a4a72dc (accessed Oct. 31, 2022).

[48] B. Korte and J. Vygen, *Combinatorial Optimization*, Fifth., vol. 21. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-24488-9.

[49] G. B. (George B. Dantzig and M. Narain. Thapa, “Linear programming,” 1997, Accessed: Feb. 22, 2022. [Online]. Available: https://books.google.com/books/about/Linear_Programming_2.html?id=kWUKBwAAQBAJ

[50] A. Schrijver, “Theory of linear and integer programming,” p. 484, 1998.

[51] “3: An example of the difference between a convex and a non-convex region. | Download Scientific Diagram.” https://www.researchgate.net/figure/An-example-of-the-

difference-between-a-convex-and-a-non-convex-region_fig4_239926467 (accessed Feb. 22, 2022).

[52] Yiling Chen, "Introduction to Optimization Models and Methods," *Harvard SEAS*. http://am121.seas.harvard.edu/site/wp content/uploads/2011/03/lecture4.handout.pdf (accessed Feb. 22, 2022).

[53] J. Mitchell, "Extreme Points and Extreme Rays," Feb. 24, 1997. https://homepages.rpi.edu/~mitchj/handouts/extreme97/extreme.html (accessed Jul. 06, 2022).

[54] J. Vinther, "Automatic Dantzig-Wolfe Reformulation of Mixed Integer Linear Programs," Technical University of Denmark, 2021. Accessed: Jul. 06, 2022. [Online]. Available: www.man.dtu.dk

[55] "Hermann Minkowski - Wikidata." https://www.wikidata.org/wiki/Q57246 (accessed Feb. 22, 2022).

[56] "Hermann Weyl - Wikipedia." https://en.wikipedia.org/wiki/Hermann_Weyl (accessed Feb. 22, 2022).

[57] D. P. Bertsekas, *Convex Optimization Algorithms*. Belmont, Massachusetts: Athena Scientific, 2009. Accessed: Feb. 21, 2022. [Online]. Available: http://www.athenasc.comWWW:http://www.athenasc.com

[58] "Taha, Operations Research: An Introduction, 10th Edition | Pearson." https://www.pearson.com/us/higher-education/program/Taha-Operations-Research-An-Introduction-10th-Edition/PGM334070.html (accessed Feb. 25, 2022).

[59] "Reduced cost - Wikipedia." https://en.wikipedia.org/wiki/Reduced_cost (accessed Feb. 25, 2022).

[60] Jr. L. R. Ford and D. R. Fulkerson, "A Suggested Computation for Maximal Multi-Commodity Network Flows," *http://dx.doi.org/10.1287/mnsc.5.1.97*, vol. 5, no. 1, pp. 97–101, Oct. 1958, doi: 10.1287/MNSC.5.1.97.

[61] "What is column generation? - IBM Documentation." https://www.ibm.com/docs/en/icos/12.9.0?topic=example-what-is-column-generation (accessed Feb. 25, 2022).

[62] "Column generation - Wikipedia." https://en.wikipedia.org/wiki/Column_generation (accessed Feb. 25, 2022).

[63] "Column generation algorithms - optimization." https://optimization.mccormick.northwestern.edu/index.php/Column_generation_algorithms (accessed Feb. 25, 2022).

[64] J.-F. Cordeaú, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis, "The VRP with Time Windows," 1999.

[65] P. C. Gilmore and R. E. Gomory, "A Linear Programming Approach to the Cutting-Stock Problem," *https://doi.org/10.1287/opre.9.6.849*, vol. 9, no. 6, pp. 849–859, Dec. 1961, doi: 10.1287/OPRE.9.6.849.

[66] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting stock problem-Part II," *Oper Res*, vol. 11, pp. 863–888, 1963.

[67] M. Desrochers and F. Soumis, "Transportation Science A Column Generation Approach to the Urban Transit Crew Scheduling Problem," 1989, doi: 10.1287/trsc.23.1.1.

[68] Guy. Desaulniers, Jacques. Desrosiers, and M. M. Solomon, "Column generation," p. 358, 2005.

[69] G. Desaulniers, J. Desrosiers, and M. M. Solomon, "Column generation," *Column Generation*, pp. 1–358, 2005, doi: 10.1007/B135457.

[70] "Column_Generation_Flowchart.png (871×1056)." https://optimization.mccormick.northwestern.edu/images/c/cc/Column_Generation_Flowchart.png (accessed Oct. 31, 2022).

[71] P. González-Brevis, J. Gondzio, and P. Munari, "A natural stabilized column generation method".

[72] M. Chiarandini, "Dantzig-Wolfe Decomposition and Delayed Column Generation," 2022.

[73] D. Friberg, "An Implementation of the Branch-and-Price Algorithm Applied to Opportunistic Maintenance Planning," CHALMERS UNIVERSITY OF TECHNOLOGY, Gothenburg, Sweden, 2015.

[74] R. H. Kwon, *Introduction to linear optimization and extensions with MATLAB*. CRC Press, 2014. Accessed: Feb. 26, 2022. [Online]. Available: https://www.routledge.com/Introduction-to-Linear-Optimization-and-Extensions-with-MATLAB/Kwon/p/book/9781439862636

[75] F. Clautiaux, "Solving cutting and packing problems with extended integer programming formulations," Mexico City, Apr. 2019.

[76] L. E. Sokoler, L. Standardi, K. Edlund, N. K. Poulsen, H. Madsen, and J. B. Jørgensen, "A Dantzig–Wolfe decomposition algorithm for linear economic model predictive control of

dynamically decoupled subsystems," *J Process Control*, vol. 24, no. 8, pp. 1225–1236, Aug. 2014, doi: 10.1016/J.JPROCONT.2014.05.013.

[77] J. Tsitsiklis, "Dantzig–Wolfe decomposition," 2002. http://web.mit.edu/6.251/www/lectnotes/10-23-lect.pdf (accessed Jul. 04, 2022).

[78] M. E. Lübbecke and J. T. Witt, "Separation of Generic Cutting Planes in Branch-and-Price Using a Basis".

[79] F. Vanderbeck, "A generic view at the Dantzig-Wolfe decomposition approach in Mixed Integer Programming: paving the way for a generic code", Accessed: Jul. 16, 2022. [Online]. Available: www//math.u-bordeaux.fr/fv

[80] F. Vanderbeck, "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Oper Res*, vol. 48, no. 1, pp. 111–128, 2000, doi: 10.1287/OPRE.48.1.111.12453.

[81] J. M. Valério De Carvalho, "Decomposition Methods for Integer Programming," 2010.

[82] A. Ceselli, L. Létocart, and E. Traversi, "Dantzig Wolfe decomposition and objective function convexification for binary quadratic problems: the cardinality constrained quadratic knapsack case," 2017.

[83] J. Jobson, "A column generation approach to scheduling of parallel identical machines," Linköping University, Linköping, 2019. Accessed: Jun. 14, 2022. [Online]. Available: https://liu.diva-portal.org/smash/get/diva2:1350640/FULLTEXT01.pdf

[84] A. M. Geoffrion, "Lagrangian Relaxation for Integer Programming," *Mathematical Programming Studies*, pp. 82–114, 1974, doi: 10.1007/978-3-540-68279-0_9.

[85] P. G. Brevis, "Advances in Interior Point Methods and Column Generation," University of Edinburgh, 2013.

[86] "Cutting stock problem - Wikipedia." https://en.wikipedia.org/wiki/Cutting_stock_problem (accessed Jul. 16, 2022).

[87] "39,885 Steel coil Images, Stock Photos & Vectors | Shutterstock." https://www.shutterstock.com/search/steel-coil (accessed Jul. 16, 2022).

[88] G. Belov, "Problems, Models and Algorithms in One- and Two-Dimensional Cutting," Technischen Universitat¨ Dresden, Dresden, 2003. Accessed: Jul. 16, 2022. [Online]. Available: https://d-nb.info/970782489/34

[89] J. M. Val and E. de Carvalho, "LP models for bin packing and cutting stock problems", Accessed: Jul. 16, 2022. [Online]. Available: www.elsevier.com/locate/dsw

[90] S. Pal, “Column Generation MTech Seminar Report”.

[91] Silvano. Martello and Paolo. Toth, *Knapsack problems: algorithms and computer implementations*. Chichester; New York: J. Wiley & Sons, 1990.

[92] P. H. Vance, “Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem,” *Computational Optimization and Applications 1998 9:3*, vol. 9, no. 3, pp. 211–228, Mar. 1998, doi: 10.1023/A:1018346107246.

[93] J. M. Valério De Carvalho, “Exact solution of bin-packing problems using column generation and branch-and-bound,” *Ann Oper Res*, vol. 86, pp. 629–659, 1999, doi: 10.1023/A:1018952112615.

[94] H. ben Amor and J. Valério De Carvalho, “Cutting Stock Problems,” in *Column Generation*, Springer, Boston, MA, 2005, pp. 131–161. doi: 10.1007/0-387-25486-2_5.

[95] “Bin packing problem - Wikipedia.” https://en.wikipedia.org/wiki/Bin_packing_problem (accessed Aug. 16, 2022).

[96] M. Delorme, M. Iori, S. Martello, and G. Marconi, “Bin Packing and Cutting Stock Problems: Mathematical Models and Exact Algorithms,” 2015.

[97] M. Ataei, “A BRANCH-AND-PRICE ALGORITHM FOR BIN PACKING PROBLEM,” YORK UNIVERSITY, TORONTO, ONTARIO, 2015.

[98] A. Silva-Gálvez, E. Lara-Cárdenas, I. Amaya, J. M. Cruz-Duarte, and J. C. Ortiz-Bayliss, “A preliminary study on score-based hyper-heuristics for solving the bin packing problem,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12088 LNCS, pp. 318–327, 2020, doi: 10.1007/978-3-030-49076-8_30/FIGURES/3.

[99] T. Gschwind and S. Irnich, “Dual Inequalities for Stabilized Column Generation Revisited,” *https://doi.org/10.1287/ijoc.2015.0670*, vol. 28, no. 1, pp. 175–194, Feb. 2016, doi: 10.1287/IJOC.2015.0670.

[100] J. Elzinga and T. G. Moore, “A central cutting plane algorithm for the convex programming problem,” *Mathematical Programming 1975 8:1*, vol. 8, no. 1, pp. 134–145, Dec. 1975, doi: 10.1007/BF01580439.

[101] W. Ben-Ameur and J. Neto, “Acceleration of cutting-plane and column generation algorithms: Applications to network design,” *Networks*, vol. 49, no. 1, pp. 3–17, Jan. 2007, doi: 10.1002/NET.20137.

[102] R. E. Marsten, W. W. Hogan, and J. W. Blankenship, “The Boxstep Method for Large-Scale Optimization,” *https://doi.org/10.1287/opre.23.3.389*, vol. 23, no. 3, pp. 389–405, Jun. 1975, doi: 10.1287/OPRE.23.3.389.

[103] A. Pessoa, E. Uchoa, and M. Poggi De Aragão, “A new stabilization method for column generation”.

[104] A. A. S. Leão, L. H. Cherri, and M. N. Arenales, “Determining the K-best solutions of knapsack problems,” *Comput Oper Res*, vol. 49, pp. 71–82, Sep. 2014, doi: 10.1016/J.COR.2014.03.008.

[105] H. H. Yanasse, N. Y. Soma, and N. Maculan, “AN ALGORITHM FOR DETERMINING THE K-BEST SOLUTIONS OF THE ONE-DIMENSIONAL KNAPSACK PROBLEM,” vol. 20, no. 1, 2000.

[106] “BPPLIB - Mathematical software - swMATH.” https://swmath.org/software/23883 (accessed Aug. 16, 2022).

[107] M. Delorme, M. Iori, and S. Martello, “BPPLIB: a library for bin packing and cutting stock problems,” *Optimization Letters 2017 12:2*, vol. 12, no. 2, pp. 235–250, Sep. 2017, doi: 10.1007/S11590-017-1192-Z.

[108] E. Falkenauer, “A hybrid grouping genetic algorithm for bin packing,” *Journal of Heuristics 1996 2:1*, vol. 2, no. 1, pp. 5–30, 1996, doi: 10.1007/BF00226291.

[109] A. Scholl, R. Klein, and C. Jürgens, “Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem,” *Comput Oper Res*, vol. 24, no. 7, pp. 627–645, Jul. 1997, doi: 10.1016/S0305-0548(96)00082-2.

[110] G. Wäscher and T. Gau, “Heuristics for the integer one-dimensional cutting stock problem: A computational study,” *Operations-Research-Spektrum 1996 18:3*, vol. 18, no. 3, pp. 131–144, 1996, doi: 10.1007/BF01539705.

[111] P. Schwerin and G. Wäscher, “The Bin-Packing Problem: A Problem Generator and Some Numerical Experiments with FFD Packing and MTP,” *International Transactions in Operational Research*, vol. 4, no. 5–6, pp. 377–389, Nov. 1997, doi: 10.1111/J.1475-3995. 1997.TB00093. X.

[112] J. E. Schoenfield, “Fast, exact solution of open bin packing problems without linear programming,” Huntsville, Alabama, USA, 2002.

[113] Suraj Sharma, “Performance comparison of Java and C++ when sorting integers and writing/reading files,” Blekinge Institute of Technology, Karlskrona, Sweden, 2019. Accessed: Oct. 25, 2022. [Online]. Available: www.bth.se

[114] "COIN-OR: Computational Infrastructure for Operations Research – Open-source Software for the Operations Research Community." https://www.coin-or.org/ (accessed Apr. 14, 2022).

[115] "Software – COR@L." https://coral.ise.lehigh.edu/software/ (accessed Apr. 14, 2022).

[116] J. Vinther, "Automatic Dantzig-Wolfe Reformulation of Mixed Integer Linear Programs," Technical University of Denmark, 2021. Accessed: Apr. 14, 2022. [Online]. Available: www.man.dtu.dk

[117] "ILOG CPLEX Optimization Studio | IBM." https://www.ibm.com/products/ilog-cplex-optimization-studio (accessed Apr. 14, 2022).

[118] "CPLEXhistory." https://web.archive.org/web/20090413054926/http://www.ilog.com/products/cplex/news/history.cfm (accessed Feb. 27, 2022).

[119] "CPLEX - Wikipedia." https://en.wikipedia.org/wiki/CPLEX (accessed Feb. 27, 2022).

[120] "Risking it All and Reaping the Rewards - Texas Engineer." https://magazine.engr.utexas.edu/2016/risking-it-all-and-reaping-the-rewards (accessed Mar. 08, 2022).

[121] "Downloading IBM ILOG CPLEX Optimization Studio 20.1.0." https://www.ibm.com/support/pages/downloading-ibm-ilog-cplex-optimization-studio-2010 (accessed Apr. 07, 2022).

[122] "IBM ILOG CPLEX Optimization Studio Getting Started with CPLEX," 2017.

Παράρτημα Α. Κατάλογος ακρωνύμων

B

Bin Packing Problem
BPP, 68
Bin Packing Problem Library
BPPLIB, 8
Binary Integer Linear Programming
BILP, 50

C

Central Processing Unit
CPU, 90
Column Generation
CG, 26
Cutting Ctock Problem
CSP, 8

D

Dantzig – Wolfe Decomposition
DWD, 6

I

Integer Linear Program
ILP, 1
Integrated Development Environment
IDE, 98
Intensified Column Generation
ICG, 86

J

Job Shop Scheduling
JSP, 10

L

Linear Program
LP, 1

M

Mixed Integer Linear Programming
MILP, 1
Multiple Column Generation, MCG, 8

O

Optimization Programming Language
OPL, 98

Q

Quadratic Programming problems
QP, 97

Παράρτημα Β. Κώδικες σε Wolfram Mathematica® που χρησιμοποιήθηκαν για τη δημιουργία εικόνων του 2ου Κεφαλαίου

Παράρτημα Β.1. Κώδικας για την δημιουργία της Εικόνα 2-1

```
In[ ]:= PolyhedronData["Dodecahedron"]
        πληροφορίες πολυέδρου
```

Παράρτημα Β.2. Κώδικας για την δημιουργία της Εικόνα 2-3 (α)

```
In[ ]:= pts = RandomReal[{-1, 1}, {50, 2}];
              τυχαίος πραγματικός αριθμός
        ConvexHullMesh[pts, Axes -> True, AxesLabel → {HoldForm[x], HoldForm[y]}]
        πλέγμα κυρτού περιβλή··· επιλογ··· αλη··· επικέτες αξόνων μορφή χωρίς αξ··· μορφή χωρίς αξιολό

In[ ]:= Show[%, Graphics[Point[pts]]]
        προβολ··· γραφικά σημείο
```

Παράρτημα Β.3. Κώδικας για την δημιουργία της Εικόνα 2-3 (β)

```
In[ ]:= pts = RandomReal[{-1, 1}, {50, 3}];
              τυχαίος πραγματικός αριθμός
        ConvexHullMesh[pts, Axes → True, AxesLabel → {HoldForm[x], HoldForm[y], HoldForm[z]},
        πλέγμα κυρτού περιβλή··· επιλο··· αλη··· επικέτες αξόνων μορφή χωρίς αξ··· μορφή χωρίς αξ··· μορφή χωρίς αξιολό
         Boxed → True]
         περίγρ··· αληθής

In[ ]:= Show[HighlightMesh[%, Style[2, Opacity[0.5]]], Graphics3D[Point[pts]]]
        πρ··· υπογράμμιση πλέγμ··· στυλ αδιαφάνεια 3D γραφικά σημείο
```

Παράρτημα Γ. Παράδειγμα πηγαίου κώδικα που επιλύει ένα πρόβλημα γραμμικού προγραμματισμού με τη χρήση της γλώσσας προγραμματισμού C++ και των βιβλιοθηκών βελτιστοποίησης του ILOG Optimization Studio®

```
#include <ilcplex/ilocplex.h>
#include <chrono>
ILOSTLBEGIN

int main(int argc, char** argv)
{
	auto start = chrono::steady_clock::now();//start of measuring time

	IloEnv env;//environment object

	/*To handle exceptions gracefully in a Concert Technology application, include all of
the code in a try/catch clause*/

	try {
		IloModel model(env);// model object

		//declaration of variables
		IloNumVar x1(env, 0.0, 40.0);//variable x1 on [0,40]
		IloNumVar x2(env, 0.0, IloInfinity,ILOFLOAT);//variable x2 on [0,infinity)
		IloNumVar x3(env, 0.0, IloInfinity, ILOFLOAT);//variable x3 on [0,infinity)

		//adding the variables to the model
		model.add(x1);
		model.add(x2);
		model.add(x3);

		//declaration and addition of the objective function to the model
		model.add(IloMaximize(env, x1 + 2 * x2+ 3 * x3));

		//declaration of the constraints
		IloRange cons_1(env, -IloInfinity, -x1 + x2 + x3, 20);
		IloRange cons_2(env, -IloInfinity, x1 - 3 * x2 + x3, 30);

		//adding the constraints to the model
		model.add(cons_1);
		model.add(cons_2);

		IloCplex cplex(model);// Model Solution
		cplex.setOut(env.getNullStream());//To avoid logging messages by CPLEX on screen
		if (!cplex.solve()) {
			env.error() << "Failed to optimize LP." << endl;
			throw(-1);
		}

		env.out() << "Solution status = " << cplex.getStatus() << endl;
		//get the objective value:
		env.out() << "Objective value = " << cplex.getObjValue() << endl;

		//To get the values of the variables:
		env.out() << "values of the variables:" << endl;
		env.out() << "x1 = " << cplex.getValue(x1) << endl;
		env.out() << "x2 = " << cplex.getValue(x2) << endl;
		env.out() << "x3 = " << cplex.getValue(x3) << endl;

		//additional information about the solution
		env.out() << " " << endl;
		env.out() << "additional information about the solution:" <<  endl;
		//To get the values of the dual variables corresponding to the constraints:
```

```
        env.out() << "dual variables"  << endl;
        env.out() << "dual of the first set = " << cplex.getDual(cons_1) << endl;
        env.out() << "dual of the second set = " << cplex.getDual(cons_2) << endl;
        env.out() << " " << endl;
        //To get the values of the reduced cost corresponding to the variables:
        env.out() << "reduced costs" << endl;
        env.out() << "reduced cost of x1 = " << cplex.getReducedCost(x1) << endl;
        env.out() << "reduced cost of x2 = " << cplex.getReducedCost(x2) << endl;
        env.out() << "reduced cost of x3 = " << cplex.getReducedCost(x3) << endl;
        env.out() << " " << endl;
        //To get the values of the slack variables corresponding to the constraints:
        env.out() << "slack variables" << endl;
        env.out() << "slack of the first set = " << cplex.getSlack(cons_1) << endl;
        env.out() << "slack of thw second set = " << cplex.getSlack(cons_2) << endl;
        env.out() << " " << endl;
    }
    catch (IloException& e) {
        cerr << "Concert exception caught: " << e << endl;
    }
    catch (...) {
        cerr << "Unknown exception caught" << endl;
    }

    auto end = chrono::steady_clock::now();//end of measuring time
    env.out() << "Elapsed time in seconds: "
        << chrono::duration_cast<chrono::seconds>(end - start).count()
        << " sec" << std::endl;
    env.end();

    return 0;
}
```

Παράρτημα Δ. Δείγμα πηγαίου κώδικα που υλοποιεί την προτεινόμενη μέθοδο δημιουργίας πολλαπλών στηλών, όπου επιλύει κάθε παράδειγμα 3 φορές, μία για κάθε τιμή της παραμέτρου a $(0.3, 0.6 \,\&\, 0.8)$, προσθέτοντας 2 στήλες στο RMP του 1D-CSP σε κάθε επανάληψη

```
#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
#include <ilcplex/ilocplex.h>
#include <stdio.h>
#include <ilconcert/iloexpression.h>
#include <vector>
#include <sys/stat.h>
#include <io.h>
#include <conio.h>
#include <process.h>
#include <direct.h>
#include <ctime>
#include <chrono>

ILOSTLBEGIN

ofstream outfile_1;//output file 1
ofstream outfile_2;//output file 2
ofstream outfile_3;//output file 3
ofstream outfile_4;//output file 4
ofstream outfile_5;//output file 5
ofstream outfile_6;//output file 6
ofstream outfile_7;//output file 7
ofstream outfile_8;//output file 8

//declarations
//--------------------------------------------------

double a;//parameterizes the level of smoothing
int k = 1;//Max number of extra columns added at each iteration

int n;//index for the number of column
double UB;//the upper bound value of the problem
double LB;//the lower bound value of the problem
int UB_exact_value;//the exact upper bound value of the problem
int LB_exact_value;//the exact lower bound value of the problem
double UB_error;//the error of the upper bound value
double LB_error;//the error of the lower bound value

int  m;//denotes the number of item type
int W;//denotes the width of rolls
int* w;//denotes the width of items
int* d;//denotes the demand for order width i
int i, p, h;//indexes
double epsilon;//convergence tolerance
vector <int*> Pattern;//possible cutting patterns

int MaxIterations = 10000;//Maximum nubmer of Column Generation Iterations
int iter_count;//iterations counter
int nc;//total generated columns number
//--------------------------------------------------

char FilePath_DataSet[1024] = "C:/.../.../.../.../Scholl_2";//path of
DataSet.txt
char FilePath_Data[1024] = "C:/.../.../.../.../Scholl_2";//path of folder
that contains the instaces
```

```
char FilePath_Results[1024] = "C:/.../.../.../.../Scholl_2/MCG2";//path of
folder that contains the results
char FileName_DataSet[512] = "DataSet";
char FileName_outfile_1[512] = "1.Initial patterns";
char FileName_outfile_2[512] = "2.RMP_values";
char FileName_outfile_3[512] = "3.Solution Summary";
char FileName_outfile_4[512] = "4.Final patterns";
char FileName_outfile_5[512] = "5.dual_values";
char FileName_outfile_6[512] = "6.primal_values";
char FileName_outfile_7[512] = "7.stab_dual_values";
char FileName_outfile_8[512] = "8.report";
const int TotalNoOfProblems = 10;//Total number of instances to be solved

string ProblemNames[TotalNoOfProblems];
double epsilon_array[TotalNoOfProblems];

int UB_exact_value_array[TotalNoOfProblems];
int LB_exact_value_array[TotalNoOfProblems];

char FileName_Problem[512];

double InsertDataSet(char* FilePath_Data_ptr)
{
	char filepath[1024];
	sprintf_s(filepath, "%s/%s.txt", FilePath_Data_ptr, FileName_DataSet);
	ifstream infile;
	infile.open(filepath);

	if (infile.fail())
	{
		cout << "DataSet file could not be opened. Try correcting the
file's location path." << endl;
		cout << filepath << endl;
		system("pause");
		exit(1);
	}
	for (int i = 0; i < TotalNoOfProblems; i++)
	{
		infile >> ProblemNames[i];
		infile >> UB_exact_value_array[i];
		infile >> LB_exact_value_array[i];
		infile >> epsilon_array[i];
	}
	infile.close();
}

int ReadDataFile(char* FilePath_Data_ptr, char* FileName_ptr) {
	char filepath[1024];
	sprintf_s(filepath, "%s/%s", FilePath_Data_ptr, FileName_ptr);
	ifstream infile;
	infile.open(filepath);
	if (infile.fail()) {
		cout << " File could not be opened. Try correcting the file's
location path." << endl;
		cout << filepath << endl;
		system("pause");
		exit(1);
	}
```

```
infile >> m;
	infile >> W;
	cout << "m=" << m << endl;
	cout << "W=" << W << endl;
	w = new int[m];
	d = new int[m];
	for (h = 0; h < m; h++)
	{
		infile >> w[h] >> d[h];
		cout << w[h] << '\t' << d[h] << endl;
	}

	infile.close();
	return 0;
}

//function containing the subproblem (slave)
double Subproblem(double* current_dual, int* NewPattern)
{
	IloEnv env;
	IloModel Slave(env);
	char filepath[1024];
	//declaration of the variable of the subproblem: a[i]=newPat
	IloNumVarArray newPat(env, m, 0, IloInfinity, ILOINT);

	//declaration of the objective function of the subproblem (equation 4-
25)
	IloExpr expr0(env);
	for (i = 0; i < m; i++)
	{
		expr0 += current_dual[i] * newPat[i];
	}
	Slave.add(IloMaximize(env, expr0));

	//declaration of the constraint of the subproblem (equation 4-26)
	IloExpr expr1(env);
	for (i = 0; i < m; i++)
	{
		expr1 += w[i] * newPat[i];
	}
	Slave.add(expr1 <= W);

	//solve the subproblem (slave)
	IloCplex cplex(Slave);
	//turn off logging output on the console window
	cplex.setOut(env.getNullStream());
	sprintf_s(filepath, "%s/%s/%f/CurrentSlave.lp", FilePath_Results,
FileName_Problem, a);
	cplex.exportModel(filepath);
	if (!cplex.solve())
	{
		env.error() << "Failed to optimize the Master Problem!" << endl;
		throw(-1);
	}
```

```
//New Pattern generation
	for (i = 0; i < m; i++)
	{
		NewPattern[i] = cplex.getValue(newPat[i]);
	}

	//get the objective function value of the SP (reduced cost)
	double reduced_cost = cplex.getObjValue();
	return reduced_cost;
}

double Subproblem_stab(double* stab_dual, int* NewPattern_stab, double*
current_dual)
{
	IloEnv env;
	IloModel Slave(env);
	char filepath[1024];
	//declaration of the variable of the subproblem: a[i]=newPat
	IloNumVarArray newPat(env, m, 0.0, IloInfinity, ILOINT);

	//declaration of the objective function of the modified subproblem
	IloExpr expr0(env);
	for (i = 0; i < m; i++)
	{
		expr0 += stab_dual[i] * newPat[i];
	}
	Slave.add(IloMaximize(env, expr0));

	//declaration of the constraint of the modified subproblem
	IloExpr expr1(env);
	for (i = 0; i < m; i++)
	{
		expr1 += w[i] * newPat[i];
	}
	Slave.add(expr1 <= W);

	//solve the modified subproblem (slave)
	IloCplex cplex(Slave);
	//turn off logging output on the console window
	cplex.setOut(env.getNullStream());
	sprintf_s(filepath, "%s/%s/%f/CurrentStabSlave.lp", FilePath_Results,
FileName_Problem, a);
	cplex.exportModel(filepath);
	if (!cplex.solve())
	{
		env.error() << "Failed to optimize the Restricted Master
Problem!" << endl;
		throw(-1);
	}

	//New Pattern generation
	for (i = 0; i < m; i++)
	{
		NewPattern_stab[i] = cplex.getValue(newPat[i]);
	}

	//get the objective function value of the (classical) SP (reduced cost)
	double reduced_cost_stabilized = 0;
```

```
//When a column is found, it is added to the RMP only when it has negative
reduced cost with respect to the current dual solution.
	for (i = 0; i < m; i++)
	{
		reduced_cost_stabilized = reduced_cost_stabilized +
current_dual[i] * NewPattern_stab[i];
	}

	return reduced_cost_stabilized;
}

//*********************************  Main program
*******************************
int main()
{
	int  stop, status;
	char filepath[1024];
	status = InsertDataSet(FilePath_DataSet);//Read the file with the names
of each instance
	if (status != 0) {
		cout << "File could not be opened! Try correcting the file's
location path." << endl;
		return -1;
	}

	for (int i_Problem = 0; i_Problem < TotalNoOfProblems; i_Problem++)
	{
		int u = 3;//Solve each problem u times
		for (int i_a = 0; i_a < u; i_a++) {

			if (i_a == 0) {
				a = 0.3;//Value of "a" for the 1st solution of the
problem
			}
			if (i_a == 1) {
				a = 0.6;//Value of "a" for the 2nd solution of the
problem
			}
			if (i_a == 2) {
				a = 0.8;//Value of "a" for the 3rd solution of the
problem
			}

			strcpy_s(FileName_Problem,
ProblemNames[i_Problem].c_str());//Assign the string value to a char value;
			UB_exact_value = UB_exact_value_array[i_Problem];//Assign
the exact value of UB
			LB_exact_value = LB_exact_value_array[i_Problem];//Assign
the exact value of LB
			epsilon = epsilon_array[i_Problem];//Assign the
convergence tolerance
			cout << "-----------------------------------------------
---------------" << endl;
			cout << "Solving test instance: " << FileName_Problem << "
(a= " << a << ",n= " << k+1 << ")" << endl;

			auto start = chrono::steady_clock::now();//start of
measuring time
```

```
//Read all data
                    status = ReadDataFile(FilePath_Data, FileName_Problem);
                    if (status != 0) {
                          cout << "File could not be opened! Try correcting
the file's location path." << endl;
                          return -1;
                    }


      //=================================================================
========

                    sprintf_s(filepath, "%s/%s", FilePath_Results,
FileName_Problem);
                    status = _mkdir(filepath);
                    sprintf_s(filepath, "%s/%s/%f", FilePath_Results,
FileName_Problem, a);
                    status = _mkdir(filepath);

                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_1);
                    outfile_1.open(filepath);
                    //open output file 1: contains the initial patterns
                    if (!outfile_1.is_open())
                    {
                          cout << "Error opening the output file 1!" << endl;
                          return 0;
                    }
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_2);
                    outfile_2.open(filepath);
                    //open output file 2: contains the values of the upper
bound of the RMP at each iteration
                    if (!outfile_2.is_open())
                    {
                          cout << "Error opening the output file 2!" << endl;
                          return 0;
                    }

                    //open output file 3: contains the number of iterations,
the value of the subproblem, the value of the upper bound of the RMP,
                    //the value of the lower bound of the RMP, the
corresponding error and the status at each iteration
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_3);
                    outfile_3.open(filepath);
                    if (!outfile_3.is_open())
                    {
                          cout << "Error opening the output file 3!" << endl;
                          return 0;
                    }

                    //open output file 4: contains the final patterns
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_4);
                    outfile_4.open(filepath);
                    if (!outfile_4.is_open())
                    {
```

```
        cout << "Error opening the output file 4!" << endl;
                        return 0;
                }

                //open output file 5: contains the values of the dual
variables at each iteration
                sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_5);
                outfile_5.open(filepath);
                if (!outfile_5.is_open())
                {
                        cout << "Error opening the output file 5!" << endl;
                        return 0;
                }

                //open output file 6: contains the values of the decision
variables (primal) lamda at each iteration
                sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_6);
                outfile_6.open(filepath);
                if (!outfile_6.is_open())
                {
                        cout << "Error opening the output file 6!" << endl;
                        return 0;
                }

                //open output file 7: contains the values of the
stabilized dual values at each iteration
                sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_7);
                outfile_7.open(filepath);
                if (!outfile_7.is_open())
                {
                        cout << "Error opening the output file 7!" << endl;
                        return 0;
                }

                sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_8);
                outfile_8.open(filepath);
                if (!outfile_8.is_open())
                {
                        cout << "Error opening the output file 8!" << endl;
                        return 0;
                }

                Pattern.clear();

                //=============== STEP 1: Initialization ===============
                // generate initial patterns
                for (i = 0; i < m; i++)
                {
                        int* Pat0 = new int[m]();
                        Pat0[i] = std::floor(W / w[i]);
                        Pattern.push_back(Pat0);
                }
```

```
//print initial patterns
                    outfile_1 << "The initial set of cutting patterns is the
following:" << endl;
                    for (p = 0; p < Pattern.size(); p++)
                    {
                           outfile_1 << "Pattern " << p << ": [";
                           for (i = 0; i < m; i++)
                           {
                                  outfile_1 << " " << Pattern[p][i];
                           }
                           outfile_1 << "]" << endl;
                    }

                    //-----------------------------------------------
                    outfile_3 << fixed;//set the decimal digits number
                    outfile_3 << setprecision(4);//4 digits
                    //-----------------------------------------------

                    outfile_3 << "
Solution Summary" << endl;
                    outfile_3 << "Problem: 1 Dimentional Cutting Stock Problem
(1D-CSP)" << endl;
                    outfile_3 << "test instance: " << FileName_Problem <<
endl;
                    if (k == 0 && a == 1)
                    {
                           outfile_3 << "Algorithm used: Classical Column
Generation" << endl;
                    }
                    else if (k >= 1 && a < 1)
                    {
                           outfile_3 << "Algorithm used: Multiple Column
Generation" << endl;
                    }
                    outfile_3 << "-------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------" << endl;
                    outfile_3 << "iteration" << "\tSP" << "\t\t\tUB" <<
"\t\t\t\tUB (IP)" << "\t\t\t\tUB_err" << "\t\t\t\tLB" << "\t\t\t\tLB (IP)" <<
"\t\t\t\tLB_err" << "\t\t\t\tSolution Status" << endl;
                    outfile_3 << "-------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------" << endl;

                    double** Dual_Vals;
                    Dual_Vals = new double* [MaxIterations];
                    for (i = 0; i < MaxIterations; i++)
                    {
                           Dual_Vals[i] = new double[m];
                    }
```

```
int iter_count = -1;
                    while (true)
                    {
                          iter_count++;
                          cout << "ITERATION:" << iter_count << endl;

                          //=============== STEP 2: Solve the restricted
master problem (RMP) and get the dual values ===============
                          IloEnv env;
                          IloModel Master(env);

                          //*************** Restricted Master Problem
***************
                          //
                          //define the one-dimensional decision variable
lamda
                          IloNumVarArray lamda(env, Pattern.size(), 0.0,
IloInfinity, ILOFLOAT);
                          IloRangeArray cons_set1(env); // name the
constraints (equation 4-23) to get the duals later

                          // define the objective function of the RMP
(equation 4-22)
                          IloExpr expr2(env);
                          for (p = 0; p < Pattern.size(); p++)
                          {
                                expr2 += lamda[p];
                          }
                          Master.add(IloMinimize(env, expr2));
                          expr2.end();

                          //constraint of the RMP (equation 4-23)
                          for (i = 0; i < m; i++)
                          {
                                IloExpr expr3(env);
                                for (p = 0; p < Pattern.size(); p++)
                                {
                                      expr3 += Pattern[p][i] * lamda[p];
                                }
                                cons_set1.add(expr3 >= d[i]);
                          }
                          Master.add(cons_set1);

                          //solve the RMP
                          IloCplex cplex(Master);
                          //cplex.extract(model);
                          sprintf_s(filepath, "%s/%s/%f/CurrentMaster.lp",
FilePath_Results, FileName_Problem, a);
                          cplex.exportModel(filepath);
                          cplex.setOut(env.getNullStream());
                          cplex.solve();
                          double UB = cplex.getObjValue();//UB=objective
value of the RMP

                          double* sol_lamda = new double[m]();
                          outfile_6 << "iter." << "\t\t\tlamda(#)" << "\t\t"
<< "lamda value" << endl;
                          for (i = 0; i < m; i++)
```

```
{
                                sol_lamda[i] = cplex.getValue(lamda[i]);
                                outfile_6 << iter_count << "\t\t\t" << i <<
"\t\t\t" << sol_lamda[i] << endl;
                        }
                        outfile_6 << "-----------------------------------
----------------------" << endl;

                        //print the value of the UB at each iteration
                        outfile_2 << iter_count << "\t\t\t" << UB << endl;

                        //absolute error and relative error of rounded UB
                        UB_error = round(UB) - UB_exact_value;
                        double* current_dual = new double[m]();
                        // get the dual variables of the constraint (eq.4-
23)
                        outfile_5 << "iter." << "\t\tdual(#)" << "\t\t\t"
<< "dual value" << endl;
                        for (i = 0; i < m; i++)
                        {
                                Dual_Vals[iter_count][i] =
cplex.getDual(cons_set1[i]);
                                current_dual[i] = Dual_Vals[iter_count][i];
                                outfile_5 << iter_count << "\t\t" << i <<
"\t\t\t" << Dual_Vals[iter_count][i] << endl;
                        }
                        outfile_5 << "-----------------------------------
----------------" << endl;

                        double* stab_dual = new double[m]();

                        outfile_7 << "iter." << "\t\tdual(#)" << "\t\t\t"
<< "dual value" << endl;
                        double reduced_cost = 0;
                        double reduced_cost_stab = 0;
                        int Extra_Col = 0;
                        if (iter_count > 0) {//0=the first iteration. The
stabilization method can not be applied at the first iteration because we
only have a single dual solution
                                //=============== STEP 3: Solve the
subproblem ===============
                                // to generate a new pattern from the
subproblem (knapsack problem):
                                //Use only current dual solution
                                int* NewPattern = new int[m];
                                reduced_cost = Subproblem(current_dual,
NewPattern);//The reduced cost for the algorithm convergence is only derived
by the current dual solution
                                if (1 - reduced_cost >= -epsilon)
                                {
                                        break;//optimal solution found
                                }
                                else
                                {
                                        Pattern.push_back(NewPattern);//add a
new column based on the current dual solution

                                }
```

```
//Use the dual solution derived from convex combination
                                for (n = 0; n < k; n++) {//add multiple extra
columns
                                        if (n < iter_count) {
                                                for (i = 0; i < m; i++)
                                                {
                                                        stab_dual[i] = (1 - a) *
Dual_Vals[iter_count - 1 - n][i] + a * Dual_Vals[iter_count][i];//combine
current with previous one
                                                        outfile_7 << iter_count
<< "\t\t" << i << "\t\t\t" << stab_dual[i] << endl;
                                                }
                                                int* NewPattern_stab = new
int[m];
                                                //call the function Subproblem()
                                                reduced_cost_stab =
Subproblem_stab(stab_dual, NewPattern_stab, current_dual);//The reduced cost
for the algorithm convergence is only derived by the current dual solution
                                                //=============== STEP 4: Check
the optimality condition/add the new pattern ===============

                                                if (iter_count > 0 && a != 1 &&
1 - reduced_cost_stab < 0) {//When a column is found it is added to the RMP
only when it has negative reduced cost with respect to the current dual
solution.

        Pattern.push_back(NewPattern_stab);//add a new column based on the
current dual solution
                                                        Extra_Col++;
                                                }
                                        }

                                }
                                cout << Extra_Col << " extra column(s) is/are
added in iteration " << iter_count << endl;
                            }
                            else {
                                //=============== STEP 3: Solve the
subproblem ===============
                                // to generate a new pattern from the
subproblem (knapsack problem):
                                int* NewPattern = new int[m];
                                //call the function Subproblem()
                                reduced_cost = Subproblem(current_dual,
NewPattern);//The reduced cost for the algorithm convergence is only derived
by the current dual solution
                                //=============== STEP 4: Check the
optimality condition/add the new pattern ===============
                                if (1 - reduced_cost >= -epsilon)
                                {
                                        break;//optimal solution found
                                }
                                else
                                {
                                        Pattern.push_back(NewPattern);//add a
new column based on the current dual solution
                                }
```

```
                        }
                        outfile_7 << "-------------------------------------
----------------" << endl;

                        //lower bound at each iteration
                        LB = UB + (1 - reduced_cost);
                        //error of rounded LB
                        LB_error = round(LB) - LB_exact_value;

                        //print the results at each iteration
                        outfile_3 << iter_count << "\t\t" << reduced_cost
<< "\t\t\t" << UB << "\t\t\t\t" << round(UB) << "\t\t\t\t" << UB_error <<
"\t\t\t\t" << LB << "\t\t\t\t" << round(LB) << "\t\t\t\t" << LB_error <<
"\t\t\t\t" << cplex.getStatus() << endl;
                }

                outfile_4 << "Final patterns:" << endl;

                int ncols;//number of columns
                ncols = 0;
                for (p = 0; p < Pattern.size(); p++)
                {
                        outfile_4 << "Pattern " << p << ": [";
                        for (i = 0; i < m; i++)
                        {
                                outfile_4 << " " << Pattern[p][i];
                        }
                        outfile_4 << " ]" << endl;
                        ncols++;
                }

                outfile_3 << "-------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
----------------------------------------------" << endl;

                auto end = chrono::steady_clock::now();//end of measuring
time

                outfile_3 << "size of orders                          : "
<< m << endl;
                outfile_3 << "width of rolls                          : "
<< W << endl;
                outfile_3 << "number of iterations                    : "
<< iter_count << endl;
                outfile_3 << "CPU time (seconds)                      : "
                        << chrono::duration_cast<chrono::seconds>(end -
start).count() << std::endl;
                outfile_3 << "total generated columns number          : "
<< ncols << endl;
                outfile_3 << "convergence tolerance (epsilon)         : "
<< epsilon << endl;
                outfile_3 << "number of columns added at each iteration: "
<< k + 1 << endl;
                outfile_3 << "level of smoothing                      : "
<< a << endl;
                outfile_3 << "Best upper bound                        : "
<< UB_exact_value << endl;
```

```
utfile_3 << "Best lower bound                        : " << LB_exact_value
<< endl;

                    //-------------------------------------------------------
--------------------------------
                    outfile_8 << "number of iterations                    : "
<< iter_count << endl;
                    outfile_8 << "CPU time (seconds)                      : "
                          << chrono::duration_cast<chrono::seconds>(end -
start).count() << std::endl;
                    outfile_8 << "total generated columns number          : "
<< ncols << endl;
                    //-------------------------------------------------------
--------------------------------

                    //close the input and the output files
                    outfile_1.close();
                    outfile_2.close();
                    outfile_3.close();
                    outfile_4.close();
                    outfile_5.close();
                    outfile_6.close();
                    outfile_7.close();
                    outfile_8.close();
             }//i_a loop

       }

       return 0;
}
```

Παράρτημα Ε. Δείγμα πηγαίου κώδικα που υλοποιεί την προτεινόμενη μέθοδο δημιουργίας πολλαπλών στηλών, όπου επιλύει κάθε παράδειγμα 3 φορές, μία για κάθε τιμή της παραμέτρου a $(0.3, 0.6 \,\&\, 0.8)$, προσθέτοντας 10 στήλες στο RMP του 1D-BPP σε κάθε επανάληψη

```
#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
#include <ilcplex/ilocplex.h>
#include <stdio.h>
#include <ilconcert/iloexpression.h>
#include <vector>
#include <sys/stat.h>
#include <io.h>
#include <conio.h>
#include <process.h>
#include <direct.h>
#include <ctime>
#include <chrono>

ILOSTLBEGIN

ofstream outfile_1;//output file 1
ofstream outfile_2;//output file 2
ofstream outfile_3;//output file 3
ofstream outfile_4;//output file 4
ofstream outfile_5;//output file 5
ofstream outfile_6;//output file 6
ofstream outfile_7;//output file 7
ofstream outfile_8;//output file 8

//declarations
//--------------------------------------------------

double a;//parameterizes the level of smoothing
int k = 9;//Max number of extra columns added at each iteration

int n;//index for the number of column
double UB;//the upper bound value of the problem
double LB;//the lower bound value of the problem
int UB_exact_value;//the exact upper bound value of the problem
int LB_exact_value;//the exact lower bound value of the problem
double UB_error;//the error of the upper bound value
double LB_error;//the error of the lower bound value

int  m;//denotes the number of item type
int W;//denotes the capacity of bins
int* w;//denotes the width of items
int i, p, h;
double epsilon;//convergence tolerance
vector <int*> Pattern;//possible cutting patterns

int MaxIterations = 10000;//Maximum nubmer of Column Generation Iterations
int iter_count;//iterations counter
int nc;//total generated columns number
//--------------------------------------------------

char FilePath_DataSet[1024] = "C:/.../.../...";//path of DataSet.txt
char FilePath_Data[1024] = "C:/.../.../...";//path of folder that contains
the instaces
```

```
char FilePath_Results[1024] = "C:/.../.../...";//path of folder that contains
the results
char FileName_DataSet[512] = "DataSet";
char FileName_outfile_1[512] = "1.Initial patterns";
char FileName_outfile_2[512] = "2.RMP_values";
char FileName_outfile_3[512] = "3.Solution Summary";
char FileName_outfile_4[512] = "4.Final patterns";
char FileName_outfile_5[512] = "5.dual_values";
char FileName_outfile_6[512] = "6.primal_values";
char FileName_outfile_7[512] = "7.stab_dual_values";
char FileName_outfile_8[512] = "8.report";
const int TotalNoOfProblems = 20;//Total number of instances to be solved

string ProblemNames[TotalNoOfProblems];
double epsilon_array[TotalNoOfProblems];

int UB_exact_value_array[TotalNoOfProblems];
int LB_exact_value_array[TotalNoOfProblems];

char FileName_Problem[512];

double InsertDataSet(char* FilePath_Data_ptr)
{
	char filepath[1024];
	sprintf_s(filepath, "%s/%s.txt", FilePath_Data_ptr, FileName_DataSet);
	ifstream infile;
	infile.open(filepath);

	if (infile.fail())
	{
		cout << "DataSet file could not be opened. Try correcting the
file's location path." << endl;
		cout << filepath << endl;
		system("pause");
		exit(1);
	}
	for (int i = 0; i < TotalNoOfProblems; i++)
	{
		infile >> ProblemNames[i];
		infile >> UB_exact_value_array[i];
		infile >> LB_exact_value_array[i];
		infile >> epsilon_array[i];
	}
	infile.close();
}
```

```
int ReadDataFile(char* FilePath_Data_ptr, char* FileName_ptr) {
	char filepath[1024];
	sprintf_s(filepath, "%s/%s", FilePath_Data_ptr, FileName_ptr);
	ifstream infile;
	infile.open(filepath);
	if (infile.fail()) {
		cout << " file could not be opened. Try correcting the file's
location path." << endl;
		cout << filepath << endl;
		system("pause");
		exit(1);
	}

	infile >> m;
	infile >> W;
	cout << "m=" << m << endl;
	cout << "W=" << W << endl;
	w = new int[m];
	for (h = 0; h < m; h++)
	{
		infile >> w[h];
		cout << w[h] << endl;
	}

	infile.close();
	return 0;
}

//function containing the subproblem (slave)
double Subproblem(double* current_dual, int* NewPattern)
{
	IloEnv env;
	IloModel Slave(env);
	char filepath[1024];
	//declaration of the variable of the subproblem: a[i]=newPat
	IloNumVarArray newPat(env, m, 0, 1, ILOINT);

	//declaration of the objective function of the subproblem (equation 4-
60)
	IloExpr expr0(env);
	for (i = 0; i < m; i++)
	{
		expr0 += current_dual[i] * newPat[i];
	}
	Slave.add(IloMaximize(env, expr0));

	//declaration of the constraint of the subproblem (equation 4-61)
	IloExpr expr1(env);
	for (i = 0; i < m; i++)
	{
		expr1 += w[i] * newPat[i];
	}
	Slave.add(expr1 <= W);

	//solve the subproblem (slave)
	IloCplex cplex(Slave);
	//turn off logging output on the console window
	cplex.setOut(env.getNullStream());
```

```
sprintf_s(filepath, "%s/%s/%f/CurrentSlave.lp", FilePath_Results,
FileName_Problem, a);
	cplex.exportModel(filepath);
	if (!cplex.solve())
	{
		env.error() << "Failed to optimize the Master Problem!" << endl;
		throw(-1);
	}

	//New Pattern generation
	for (i = 0; i < m; i++)
	{
		NewPattern[i] = cplex.getValue(newPat[i]);
	}

	//get the objective function value of the SP (reduced cost)
	double reduced_cost = cplex.getObjValue();
	return reduced_cost;
}

double Subproblem_stab(double* stab_dual, int* NewPattern_stab, double*
current_dual)
{
	IloEnv env;
	IloModel Slave(env);
	char filepath[1024];
	//declaration of the variable of the modified subproblem: a[i]=newPat
	IloNumVarArray newPat(env, m, 0.0, 1, ILOINT);

	IloExpr expr0(env);
	for (i = 0; i < m; i++)
	{
		expr0 += stab_dual[i] * newPat[i];
	}
	Slave.add(IloMaximize(env, expr0));

	IloExpr expr1(env);
	for (i = 0; i < m; i++)
	{
		expr1 += w[i] * newPat[i];
	}
	Slave.add(expr1 <= W);

	//solve the subproblem (slave)
	IloCplex cplex(Slave);
	//turn off logging output on the console window
	cplex.setOut(env.getNullStream());
	sprintf_s(filepath, "%s/%s/%f/CurrentStabSlave.lp", FilePath_Results,
FileName_Problem, a);
	cplex.exportModel(filepath);
	if (!cplex.solve())
	{
		env.error() << "Failed to optimize the Master Problem!" << endl;
		throw(-1);
	}

	//New Pattern generation
	for (i = 0; i < m; i++)
```

```
{
		NewPattern[i] = cplex.getValue(newPat[i]);
	}

	//get the objective function value of the SP (reduced cost)
	double reduced_cost = cplex.getObjValue();
	return reduced_cost;
}

double Subproblem_stab(double* stab_dual, int* NewPattern_stab, double*
current_dual)
{
	IloEnv env;
	IloModel Slave(env);
	char filepath[1024];
	//declaration of the variable of the modified subproblem: a[i]=newPat
	IloNumVarArray newPat(env, m, 0.0, 1, ILOINT);

	IloExpr expr0(env);
	for (i = 0; i < m; i++)
	{
		expr0 += stab_dual[i] * newPat[i];
	}
	Slave.add(IloMaximize(env, expr0));

	IloExpr expr1(env);
	for (i = 0; i < m; i++)
	{
		expr1 += w[i] * newPat[i];
	}
	Slave.add(expr1 <= W);

	//solve the subproblem (slave)
	IloCplex cplex(Slave);
	//turn off logging output on the console window
	cplex.setOut(env.getNullStream());
	sprintf_s(filepath, "%s/%s/%f/CurrentStabSlave.lp", FilePath_Results,
FileName_Problem, a);
	cplex.exportModel(filepath);
	if (!cplex.solve())
	{
		env.error() << "Failed to optimize the Master Problem!" << endl;
		throw(-1);
	}



/New Pattern generation
	for (i = 0; i < m; i++)
	{
		NewPattern_stab[i] = cplex.getValue(newPat[i]);
	}

	//get the objective function value of the SP (reduced cost)
	double reduced_cost_stabilized = 0;
	//When a column is fo14nd it is added to the RMP only when it has
negative reduced cost with respect to the current dual solution.
	for (i = 0; i < m; i++)
	{
		reduced_cost_stabilized = reduced_cost_stabilized +
current_dual[i] * NewPattern_stab[i];
	}
```

```
return reduced_cost_stabilized;
}

//*********************************  Main program
*********************************
int main()
{
	int  stop, status;
	char filepath[1024];
	status = InsertDataSet(FilePath_DataSet);//Read the file with the names
of each instance
	if (status != 0) {
		cout << "An error has occured! Try correcting the file's
location path." << endl; //Found_Error("InsertDataSet");
		return -1;
	}

	for (int i_Problem = 0; i_Problem < TotalNoOfProblems; i_Problem++)
	{
		for (int i_a = 0; i_a < 3; i_a++) {//Solve each problem 3 times

			if (i_a == 0) {
				a = 0.3;//Value of "a" for the 1st solution of the
problem
			}
			if (i_a == 1) {
				a = 0.6;//Value of "a" for the 2nd solution of the
problem
			}
			if (i_a == 2) {
				a = 0.8;//Value of "a" for the 3rd solution of the
problem
			}

			strcpy_s(FileName_Problem,
ProblemNames[i_Problem].c_str());//Assign the string value to a char value;
			UB_exact_value = UB_exact_value_array[i_Problem];//Assign
the exact value of UB
			LB_exact_value = LB_exact_value_array[i_Problem];//Assign
the exact value of LB
			epsilon = epsilon_array[i_Problem];//Assign the
convergence tolerance
			cout << "-------------------------------------------------
---------------" << endl;
			cout << "Solving test instance: " << FileName_Problem << "
(a= " << a << "," << "n= " << k + 1 << ")" << endl;

			auto start = chrono::steady_clock::now();//start of
measuring time


    //Read all data
                    status = ReadDataFile(FilePath_Data, FileName_Problem);
                    if (status != 0) {
                            cout << "An error has occured! Try correcting the
    file's location path." << endl; //Found_Error("ReadDataFile");
                            return -1;
                    }


            //=================================================================
    ========
```

```
sprintf_s(filepath, "%s/%s", FilePath_Results, FileName_Problem);
                    status = _mkdir(filepath);
                    sprintf_s(filepath, "%s/%s/%f", FilePath_Results,
FileName_Problem, a);
                    status = _mkdir(filepath);

                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_1);
                    outfile_1.open(filepath);
                    //open output file 1: contains the initial patterns
                    if (!outfile_1.is_open())
                    {
                          cout << "Error opening the output file 1!" << endl;
                          return 0;
                    }
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_2);
                    outfile_2.open(filepath);
                    //open output file 2: contains the values of the upper
bound of the RMP at each iteration
                    if (!outfile_2.is_open())
                    {
                          cout << "Error opening the output file 2!" << endl;
                          return 0;
                    }

                    //open output file 3: contains the number of iterations,
the value of the subproblem, the value of the upper bound,
                    //the value of the lower bound of the RMP, the
corresponding relative error and the status at each iteration
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_3);
                    outfile_3.open(filepath);
                    if (!outfile_3.is_open())
                    {
                          cout << "Error opening the output file 3!" << endl;
                          return 0;
                    }

                    //open output file 4: contains the final patterns
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_4);
                    outfile_4.open(filepath);
                    if (!outfile_4.is_open())
                    {
                          cout << "Error opening the output file 4!" << endl;
                          return 0;
                    }

                    //open output file 5: contains the values of the dual
variables at each iteration
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_5);
                    outfile_5.open(filepath);
                    if (!outfile_5.is_open())
                    {
                          cout << "Error opening the output file 5!" << endl;
                          return 0;
```

```
}

                    //open output file 6: contains the values of the decision
variables lamda at each iteration
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_6);
                    outfile_6.open(filepath);
                    if (!outfile_6.is_open())
                    {
                          cout << "Error opening the output file 6!" << endl;
                          return 0;
                    }

                    //open output file 7: contains the values of the
stabilized dual values at each iteration
                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_7);
                    outfile_7.open(filepath);
                    if (!outfile_7.is_open())
                    {
                          cout << "Error opening the output file 7!" << endl;
                          return 0;
                    }

                    sprintf_s(filepath, "%s/%s/%f/%s.txt", FilePath_Results,
FileName_Problem, a, FileName_outfile_8);
                    outfile_8.open(filepath);
                    if (!outfile_8.is_open())
                    {
                          cout << "Error opening the output file 8!" << endl;
                          return 0;
                    }


                    Pattern.clear();

                    //=============== STEP 1: Initialization ===============
                    // generate initial patterns
                    for (i = 0; i < m; i++)
                    {
                          int* Pat0 = new int[m]();
                          Pat0[i] = std::floor(w[i] / w[i]);
                          Pattern.push_back(Pat0);
                    }

                    //print initial patterns
                    outfile_1 << "The initial set of cutting patterns is the
following:" << endl;
                    for (p = 0; p < Pattern.size(); p++)
                    {
                          outfile_1 << "Pattern " << p << ": [";
                          for (i = 0; i < m; i++)
                          {
                                outfile_1 << " " << Pattern[p][i];
                          }
                          outfile_1 << "]" << endl;
                    }
```

```
//-----------------------------------------------
                    outfile_3 << fixed;//set the decimal digits number
                    outfile_3 << setprecision(4);//4 digits
                    //-----------------------------------------------

                    outfile_3 << "
Solution Summary" << endl;
                    outfile_3 << "Problem: 1 Dimentional Bin Packing Problem
(1D-BPP)" << endl;
                    outfile_3 << "test instance: " << FileName_Problem <<
endl;
                    if (k == 0 && a == 1)
                    {
                          outfile_3 << "Algorithm used: Classical Column
Generation" << endl;
                    }
                    else if (k >= 1 && a < 1)
                    {
                          outfile_3 << "Algorithm used: Multiple Column
Generation" << endl;
                    }
                    outfile_3 << "-------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------" << endl;
                    outfile_3 << "iteration" << "\tSP" << "\t\t\tUB" <<
"\t\t\t\tUB (IP)" << "\t\t\t\tUB_err" << "\t\t\t\tLB" << "\t\t\t\tLB (IP)" <<
"\t\t\t\tLB_err" << "\t\t\t\tSolution Status" << endl;
                    outfile_3 << "-------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------" << endl;

                    double** Dual_Vals;
                    Dual_Vals = new double* [MaxIterations];
                    for (i = 0; i < MaxIterations; i++)
                    {
                          Dual_Vals[i] = new double[m];
                    }

                    int iter_count = -1;
                    while (true)
                    {
                          iter_count++;
                          cout << "ITERATION:" << iter_count << endl;

                          //=============== STEP 2: Solve the restricted
master problem (RMP) and get the dual values ===============
                          IloEnv env;
                          IloModel Master(env);

                          //*************** Restricted Master Problem
***************
                          //
                          //define the one-dimensional decision variable
lamda
                          IloNumVarArray lamda(env, Pattern.size(), 0.0,
IloInfinity, ILOINT);
```

```
IloRangeArray cons_set1(env); // name constraints (equation 4-58) to get the
duals later

                            // define the objective function of the RMP
(equation 4-57)
                            IloExpr expr2(env);
                            for (p = 0; p < Pattern.size(); p++)
                            {
                                  expr2 += lamda[p];
                            }
                            Master.add(IloMinimize(env, expr2));
                            expr2.end();

                            //constraint of the RMP (equation 4-58)
                            for (i = 0; i < m; i++)
                            {
                                  IloExpr expr3(env);
                                  for (p = 0; p < Pattern.size(); p++)
                                  {
                                        expr3 += Pattern[p][i] * lamda[p];
                                  }
                                  cons_set1.add(expr3 >= 1);
                            }
                            Master.add(cons_set1);

                            //solve the RMP
                            IloCplex cplex(Master);
                            //cplex.extract(model);
                            sprintf_s(filepath, "%s/%s/%f/CurrentMaster.lp",
FilePath_Results, FileName_Problem, a);
                            cplex.exportModel(filepath);
                            cplex.setOut(env.getNullStream());
                            cplex.solve();
                            double UB = cplex.getObjValue();//UB=objective
value of the RMP

                            double* sol_lamda = new double[m]();
                            outfile_6 << "iter." << "\t\t\tlamda(#)" << "\t\t"
<< "lamda value" << endl;
                            for (i = 0; i < m; i++)
                            {
                                  sol_lamda[i] = cplex.getValue(lamda[i]);
                                  outfile_6 << iter_count << "\t\t\t" << i <<
"\t\t\t" << sol_lamda[i] << endl;
                            }
                            outfile_6 << "-------------------------------------
----------------------" << endl;

                            //print the value of the UB at each iteration
                            outfile_2 << iter_count << "\t\t\t" << UB << endl;

                            //absolute error and relative error of rounded UB
                            UB_error = round(UB) - UB_exact_value;
                            double* current_dual = new double[m]();
                            // get the dual variables of the constraint (eq.4-
58)
                            outfile_5 << "iter." << "\t\tdual(#)" << "\t\t\t"
<< "dual value" << endl;
```

```
for (i = 0; i < m; i++)
                        {
                                Dual_Vals[iter_count][i] =
cplex.getDual(cons_set1[i]);
                                current_dual[i] = Dual_Vals[iter_count][i];
                                outfile_5 << iter_count << "\t\t" << i <<
"\t\t\t" << Dual_Vals[iter_count][i] << endl;
                        }
                        outfile_5 << "-------------------------------------
----------------" << endl;

                        double* stab_dual = new double[m]();

                        outfile_7 << "iter." << "\t\tdual(#)" << "\t\t\t"
<< "dual value" << endl;
                        double reduced_cost = 0;
                        double reduced_cost_stab = 0;
                        int Extra_Col = 0;
                        if (iter_count > 0) {//0=the first iteration. The
stabilization method can not be applied at the first iteration because we
only have a single dual solution
                                //=============== STEP 3: Solve the
subproblem ===============
                                // to generate a new pattern from the
subproblem (knapsack problem):
                                //Use only current dual solution
                                int* NewPattern = new int[m];
                                reduced_cost = Subproblem(current_dual,
NewPattern);//The reduced cost for the algorithm convergence is only derived
by the current dual solution
                                if (1 - reduced_cost >= -epsilon)
                                {
                                        break;//optimal solution found
                                }
                                else
                                {
                                        Pattern.push_back(NewPattern);//add a
new column based on the current dual solution

                                }
                                //Use dual solution derived from convex
combination
                                for (n = 0; n < k; n++) {//add multiple extra
columns
                                        if (n < iter_count) {
                                                for (i = 0; i < m; i++)
                                                {
                                                        stab_dual[i] = (1 - a) *
Dual_Vals[iter_count - 1 - n][i] + a * Dual_Vals[iter_count][i];//combine
current with previous one
                                                        outfile_7 << iter_count
<< "\t\t" << i << "\t\t\t" << stab_dual[i] << endl;
                                                }
                                                int* NewPattern_stab = new
int[m];
                                                //call the function Subproblem()
```

```
reduced_cost_stab = Subproblem_stab(stab_dual, NewPattern_stab,
current_dual);//The reduced cost for the algorithm convergence is only
derived by the current dual solution
                                                //=============== STEP 4: Check
the optimality condition/add the new pattern ===============

                                                if (iter_count > 0 && a != 1 &&
1 - reduced_cost_stab < 0) {//When a column is found it is added to the RMP
only when it has negative reduced cost with respect to the current dual
solution.

      Pattern.push_back(NewPattern_stab);//add a new column based on the
current dual solution
                                                      Extra_Col++;
                                                }
                                          }

                                    }
                                    cout << Extra_Col << " extra column(s) is/are
added in iteration " << iter_count << endl;
                              }
                              else {
                                    //=============== STEP 3: Solve the
subproblem ===============
                                    // to generate a new pattern from the
subproblem (knapsack problem):
                                    int* NewPattern = new int[m];
                                    //call the function Subproblem()
                                    reduced_cost = Subproblem(current_dual,
NewPattern);//The reduced cost for the algorithm convergence is only derived
by the current dual solution
                                    //=============== STEP 4: Check the
optimality condition/add the new pattern ===============
                                    if (1 - reduced_cost >= -epsilon)
                                    {
                                          break;//optimal solution found
                                    }
                                    else
                                    {
                                          Pattern.push_back(NewPattern);//add a
new column based on the current dual solution
                                    }

                              }
                              outfile_7 << "-------------------------------------
----------------" << endl;
```

```
//lower bound at each iteration
                        LB = UB + (1 - reduced_cost);
                        //error of rounded LB
                        LB_error = round(LB) - LB_exact_value;

                        //print the results at each iteration
                        outfile_3 << iter_count << "\t\t" << reduced_cost
<< "\t\t\t" << UB << "\t\t\t\t" << round(UB) << "\t\t\t\t" << UB_error <<
"\t\t\t\t" << LB << "\t\t\t\t" << round(LB) << "\t\t\t\t" << LB_error <<
"\t\t\t\t" << cplex.getStatus() << endl;
                }

                outfile_4 << "Final patterns:" << endl;

                int ncols;//nuber of columns
                ncols = 0;
                for (p = 0; p < Pattern.size(); p++)
                {
                        outfile_4 << "Pattern " << p << ": [";
                        for (i = 0; i < m; i++)
                        {
                                outfile_4 << " " << Pattern[p][i];
                        }
                        outfile_4 << " ]" << endl;
                        ncols++;
                }

                outfile_3 << "-----------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------------------------" << endl;

                auto end = chrono::steady_clock::now();//end of measuring
time

                outfile_3 << "size of orders                          : "
<< m << endl;
                outfile_3 << "width of rolls                          : "
<< W << endl;
                outfile_3 << "number of iterations                    : "
<< iter_count << endl;
                outfile_3 << "CPU time (seconds)                      : "
                        << chrono::duration_cast<chrono::seconds>(end -
start).count() << std::endl;
                outfile_3 << "total generated columns number          : "
<< ncols << endl;
                outfile_3 << "convergence tolerance (epsilon)         : "
<< epsilon << endl;
                outfile_3 << "number of columns added at each iteration: "
<< k + 1 << endl;
                outfile_3 << "level of smoothing                      : "
<< a << endl;
                outfile_3 << "Best upper bound                        : "
<< UB_exact_value << endl;
                outfile_3 << "Best lower bound                        : "
<< LB_exact_value << endl;
                //-------------------------------------------------------
-----------------------------
```

```
//report
                    outfile_8 << "number of iterations                    : "
<< iter_count << endl;
                    outfile_8 << "CPU time                                : "
                          << chrono::duration_cast<chrono::seconds>(end -
start).count()
                          << " seconds" << std::endl;
                    outfile_8 << "total generated columns number          : "
<< ncols << endl;
                    //-------------------------------------------------------
-----------------------------
                    //close the input and the output files
                    outfile_1.close();
                    outfile_2.close();
                    outfile_3.close();
                    outfile_4.close();
                    outfile_5.close();
                    outfile_6.close();
                    outfile_7.close();
                    outfile_8.close();
             }//i_a loop

      }

      return 0;
}
```