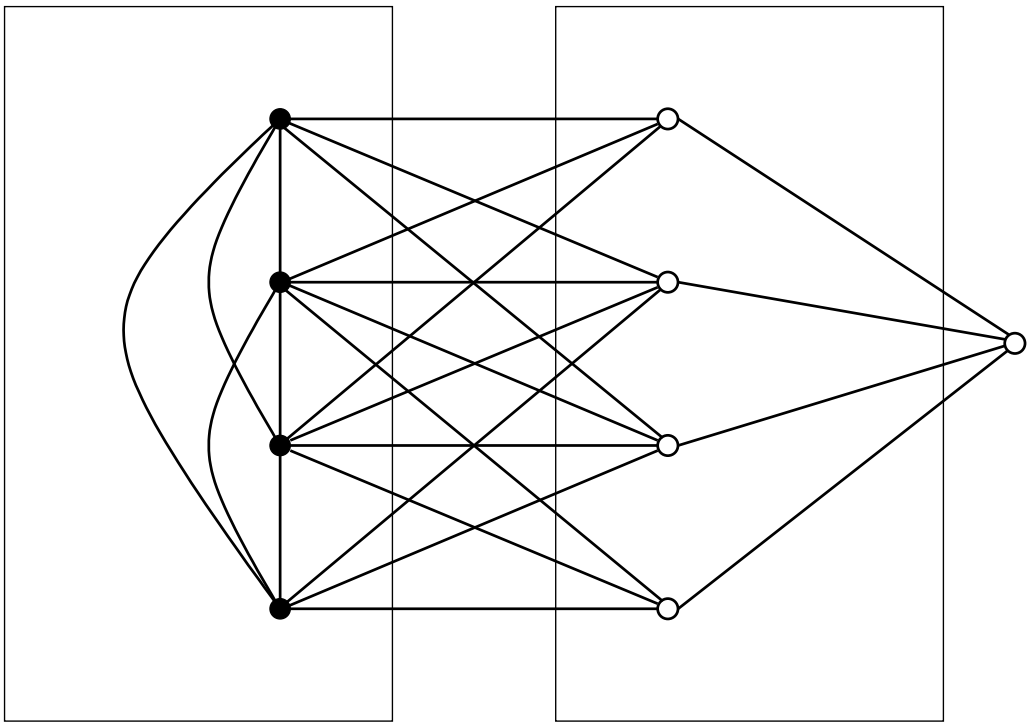
ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΟΜΕΑΣ ΘΕΩΡΗΤΙΚΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αλγοριθμική Επιχειρησιακή Έρευνα

Β. Ζησιμόπουλος



Αθήνα, 2016

Ευχαριστίες

Ευχαριστώ θερμά...

Β. Ζησιμόπουλος

Περιεχόμενα

Κεφάλαιο 1

Μοντέλα Επιχειρησιακής Έρευνας

Η Επιχειρησιακή Έρευνα (Operations Research) αποσκοπεί στην επίλυση προβλημάτων της εκτελεστικής λειτουργίας της διοίκησης με τη βοήθεια επιστημονικών μεθόδων. Οι συνθήκες, οι οποίες κατέστησαν αναγκαία την ανάπτυξη της Επιχειρησιακής Έρευνας (ΕΕ), άρχισαν να διαμορφώνονται κατά τη διάρκεια της πρώτης Βιομηχανικής Επανάστασης. Σήμερα οι εφαρμογές της ΕΕ καλύπτουν σχεδόν κάθε τομέα της ανθρώπινης δραστηριότητας όπως τις τηλεπικοινωνίες, τα εθνικά δίκτυα διαχείρισης υδάτινων πόρων, την κατανομή υδάτινων και ορυκτών πόρων στο σύστημα παραγωγής ηλεκτρικής ενέργειας, τη διαχείριση αποθεμάτων ανταλλακτικών, τη δρομολόγηση φορτηγών αυτοκινήτων, τις νοσηλευτικές μονάδες κ.λ.π.

Στόχος της ΕΕ είναι η επινόηση και η εφαρμογή επιστημονικών μεθόδων για την επίλυση προβλημάτων που αφορούν στη βελτιστοποίηση ενός συστήματος. Η λήψη αποφάσεων σχετικά με το βέλτιστο τρόπο λειτουργίας, τη βέλτιστη δομή ενός συστήματος ή τη βέλτιστη συμπεριφορά ενός φυσικού φαινομένου γίνεται με ένα μοντέλο, που είναι μια αναπαράσταση του πραγματικού συστήματος ή του φυσικού φαινομένου.

Οι βασικοί τύποι μοντέλων που χρησιμοποιούνται στην ΕΕ είναι τα μαθηματικά ή αναλυτικά μοντέλα και τα μοντέλα προσομοίωσης.

Τα μαθηματικά μοντέλα περιγράφουν τη δομή και τη λειτουργία του συστήματος με μαθηματικές σχέσεις, στις οποίες διακρίνονται τρεις ομάδες στοιχείων:

1. Παράμετροι (parameters) ή μεταβλητές απόφασης (decision variables) οι οποίες ποσοτικοποιούν κάποιες αποφάσεις που πρέπει να ληφθούν.

2. Περιορισμοί (constraints) οι οποίοι είναι ανισοεξισώσεις οι οποίες εκφράζουν τους

φυσικούς περιορισμούς.

3. Αντικειμενική συνάρτηση (objective function) η οποία περιγράφει ένα κριτήριο ή μέτρο απόδοσης (performance criterion) του συστήματος και η οποία πρόκειται να βελτιστοποιηθεί.

Η γενική μορφή ενός μαθηματικού μοντέλου είναι:

$$optimum(maximum \text{ or } minimum) \;\; z = f(x_1, x_2, \ldots, x_n)$$

με τους περιορισμούς

$$\begin{cases} H_i(x_1, x_2, \ldots, x_n) = 0, & i = 1, \ldots, m \\ G_j(x_1, x_2, \ldots, x_n) \leq 0, & j = 1, \ldots, k \end{cases}$$

και ο στόχος είναι να προσδιοριστούν οι τιμές των μεταβλητών $x_i, i = 1, \ldots, n$ έτσι ώστε η τιμή της αντικειμενικής συνάρτησης f να είναι βέλτιστη.

Ανάλογα με την φύση των περιορισμών και της προς βελτιστοποίηση αντικειμενικής συνάρτησης διακρίνουμε τις παρακάτω οικογένειες μαθηματικών μοντέλων:

1. Μοντέλο γραμμικού προγραμματισμού (linear programming model). Χαρακτηριστικό γνώρισμα του μοντέλου είναι η γραμμικότητα όλων των εμπλεκομένων συναρτήσεων ως προς τις μεταβλητές απόφασης.

   Παράδειγμα:

   maximize $z = 100x_1 + 150x_2 + 200x_3 + 400x_4$

   με τους περιορισμούς:

   $$\begin{aligned} &2x_1 + 4x_2 + 8x_3 + 10x_4 \leq 150 \\ &10x_1 + 12x_2 + 25x_3 + 20x_4 \leq 1000 \\ &x_i \geq 0, \;\; i = 1, 2, 3, 4 \end{aligned}$$

   Αν $x_i \in \mathbb{N}$ τότε έχουμε το μοντέλο του ακέραιου γραμμικού προγραμματισμού, ενώ αν $x_i \in \mathbb{R}$ τότε έχουμε το μοντέλο του συνεχούς γραμμικού προγραμματισμού.

2. Μοντέλο μη γραμμικού προγραμματισμού (non linear programming model). Χαρακτηριστικό γνώρισμα του μοντέλου είναι ότι τουλάχιστον μία από τις εμπλεκόμενες συναρτήσεις είναι μη γραμμική ως προς τις μεταβλητές απόφασης.

   minimize $z = f(x, y) = \sum_{i=1}^{n} [(x - x_i)^2 + (y - y_i)^2]^{\frac{1}{2}}$

   με τους περιορισμούς

   $$(x-1)^2 + (y-3)^2 \leq 0.5^2$$
   $$y + 2x \geq 4$$

   όπου x, y οι μεταβλητές απόφασης.

3. Μοντέλο τετραγωνικού προγραμματισμού (quadratic programming model).

   maximize $q(x) = r^t x - \mu x^t A x$

   με τους περιορισμούς

   $$\sum_{i=1}^{n} x_i = 1, \quad x_i \geq 0, i = 1, \ldots, n$$

   όπου $x^t = (x_1, x_2, \ldots, x_n)$ το διάνυσμα των μεταβλητών απόφασης,

   $r^t = (r_1, \ldots, r_n)$ ένα διάνυσμα γνωστών συντελεστών,

   μ ένας συντελεστής βαρύτητας και

   A ένας τετραγωνικός πίνακας $n \times n$.

   Παραδείγματα τέτοιων προβλημάτων συναντάμε στη βέλτιστη διαχείριση χαρτοφυλακίων (Markowitz βραβείο Νόμπελ Οικονομίας, 1990).

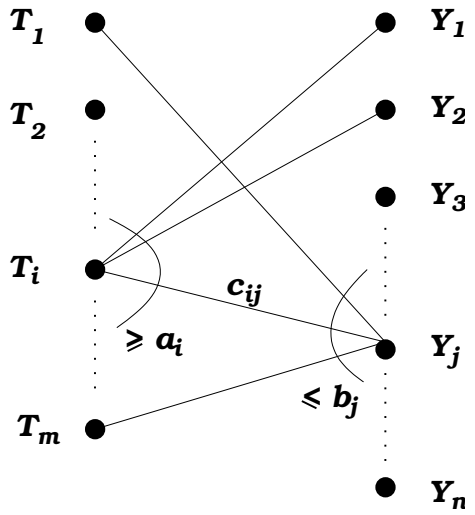
Αναμφίβολα, η μοντελοποίηση ενός συστήματος του πραγματικού κόσμου δεν είναι καθόλου εύκολη υπόθεση. Απαιτεί εμπειρία, γνώσεις και υπομονή.

Για να συλλάβουμε το πνεύμα μοντελοποίησης ας θεωρήσουμε το παρακάτω πρόβλημα το οποίο μοντελοποιείται σαν ένα πρόβλημα γραμμικού προγραμματισμού.

Ένας αριθμός τερματικών (σχήμα 1.1) πρόκειται να συνδεθεί σε ένα δίκτυο Η/Υ. Έστω, m ο αριθμός των τερματικών, n ο αριθμός των Η/Υ, b_j ο μέγιστος αριθμός τερματικών που επιτρέπεται να συνδεθεί με τον j-οστό Η/Υ, $j = 1, \ldots, n$, a_i ο ελάχιστος αριθμός Η/Υ με τους οποίους πρέπει, για λόγους ασφαλείας, να συνδεθεί το

i-οστό τερματικό, $i = 1, \ldots, m$ και c_{ij} το κόστος σύνδεσης του τερματικού i με τον Η/Υ j. Να διαμορφωθεί ένα μαθηματικό μοντέλο, του οποίου η λύση υποδεικνύει τη συνδεσμολογία που ελαχιστοποιεί το συνολικό κόστος του δικτύου.

Έστω, $T_1, \ldots, T_m$ τα τερματικά και $Y_1, Y_2, \ldots, Y_n$ οι υπολογιστές. Έστω, x_{ij}, η μεταβλητή που παριστάνει την απόφαση αν το i-οστό τερματικό θα συνδεθεί με τον j-οστό Η/Υ, όπου $x_{ij} \in \{0, 1\}$.



Σχήμα 1.1: m Τερματικά συνδεδεμένα με n Η/Υ. c_{ij} είναι το κόστος σύνδεσης του i-οστού Τερματικού με τον j-οστό Υπολογιστή.

Τότε οι περιορισμοί του προβλήματος εκφράζονται ως εξής:

$$x_{i1} + x_{i2} + \cdots + x_{ij} + \cdots + x_{in} \geq a_i, \quad i = 1, \ldots, m$$
$$x_{1j} + x_{2j} + \cdots + x_{ij} + \cdots + x_{mj} \leq b_j, \quad j = 1, \ldots, n$$

Εξάλλου, το κόστος του δικτύου επιβαρύνεται κατά c_{ij} μόνο όταν $x_{ij} = 1$ (δηλ. το i-οστό τερματικό συνδέεται με τον j-οστό υπολογιστή). Επομένως το συνολικό κόστος του δικτύου που πρόκειται να ελαχιστοποιηθεί, παρέχεται από τη συνάρτηση

$$f(x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{ij}, \ldots, x_{mn}) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

με τους περιορισμούς

$$x_{i1} + x_{i2} + \cdots + x_{in} \geq a_i, \quad i = 1, \ldots, m$$
$$x_{1j} + x_{2j} + \cdots + x_m \leq b_j, \quad j = 1, \ldots, n$$

το οποίο είναι ένα μοντέλο ακέραιου γραμμικού προγραμματισμού και επειδή οι μεταβλητές απόφασης είναι δυαδικές, είναι γνωστό σαν μοντέλο δυαδικού γραμμικού προγραμματισμού.

Όπως είπαμε προηγουμένως, η μοντελοποίηση με μαθηματικές σχέσεις γενικά είναι δύσκολη για ένα σύνθετο και πολύπλοκο σύστημα. Μερικές φορές μπορεί να είναι αδύνατη. Σε αυτήν την περίπτωση καταφεύγουμε σε μία απομίμηση της συμπεριφοράς του πραγματικού συστήματος με τη βοήθεια ενός Η/Υ. Δηλαδή, αναπτύσσουμε ένα πρόγραμμα που προσομοιώνει το πραγματικό σύστημα. Αυτή η αντιμετώπιση αποτελεί τη δεύτερη κατηγορία μοντέλων ΕΕ, τα μοντέλα προσομοίωσης τα οποία είναι πειραματικά.

1.1 Στάδια μελέτης στην Επιχειρησιακή Έρευνα

Η λήψη αποφάσεων με τη μεθοδολογία της ΕΕ ακολουθεί πέντε κύρια στάδια: αναγνώριση και περιγραφή του προβλήματος, διαμόρφωση του μοντέλου, επίλυση του μοντέλου, έλεγχος και επιβεβαίωση του μοντέλου, υλοποίηση της λύσης.

Στη συνέχεια θα ασχοληθούμε με τις πιο ενδιαφέρουσες τεχνικές για την επίλυση διαφόρων μαθηματικών μοντέλων και ειδικότερα των μοντέλων του συνεχούς γραμμικού προγραμματισμού (linear programming).

Θα μελετήσουμε τη μέθοδο Simplex για μοντέλα γραμμικού προγραμματισμού με παραμέτρους που δέχονται πραγματικές τιμές, τις μεθόδους Branch and Bound και δυναμικού προγραμματισμού για μοντέλα γραμμικού προγραμματισμού με ακέραιες παραμέτρους (μεταβλητές), καθώς και άλλες οικογένειες μεθόδων, όπως οι Ευριστικές (Heuristics), τοπικής αναζήτησης (local search) και Simulated annealing.

Οι τεχνικές αυτές θα περιγραφούν σε διάφορα απλά στη μοντελοποίηση προβλήματα, τα οποία όμως παίζουν σημαντικό ρόλο στις πραγματικές εφαρμογές, εμφανιζόμενα είτε ως έχουν σε πολλές περιπτώσεις είτε ως υποπροβλήματα σε πιο πολύπλοκες εφαρμογές.

Μερικά από τα προβλήματα που θα παρουσιαστούν είναι εύκολα, δηλαδή γνωρίζου-

με πολυωνυμικό αλγόριθμο που εξασφαλίζει τη βέλτιστη λύση (polynomial problems). Αυτά τα προβλήματα ανήκουν στην κλάση πολυπλοκότητας P (βλέπε κεφάλαιο 2). Άλλα προβλήματα χαρακτηρίζονται σαν δύσκολα δεδομένου ότι δεν γνωρίζουμε πολυωνυμικό αλγόριθμο που να επιτρέπει να υπολογίσουμε τη βέλτιστη λύση. Υπάρχουν αλγόριθμοι, που επιτρέπουν τον υπολογισμό της βέλτιστης λύσης, αλλά είναι εκθετικής πολυπλοκότητας (π.χ. Branch and Bound, Δυναμικός προγραμματισμός). Σε αυτή την περίπτωση όταν το μέγεθος του προβλήματος αυξάνει ''σημαντικά'' ο απαιτούμενος χρόνος εκτέλεσης γίνεται υπερβολικά υψηλός και η μέθοδος χάνει κάθε ενδιαφέρον. Επομένως, η διέξοδος είναι να αναζητήσουμε γρήγορες μεθόδους, οι οποίες επιτρέπουν να βρούμε μία προσεγγιστική λύση του προβλήματος (heuristics, local search, simulated annealing), για την οποία όμως, γενικά, δεν γνωρίζουμε πόσο κοντά είναι στη βέλτιστη λύση.

Για να καταλάβουμε γιατί αυτά τα προβλήματα δεν επιδέχονται επί του παρόντος έναν αποτελεσματικό αλγόριθμο (polynomial) δίνουμε στη συνέχεια την έννοια του προβλήματος NP-complete. Κατ΄ αρχήν ας υπενθυμίσουμε μερικές βασικές έννοιες της πολυπλοκότητας των αλγορίθμων.

1.2 Πολυπλοκότητα αλγορίθμων

Για να αποτιμήσουμε και να ταξινομήσουμε τους διάφορους αναπτυχθέντες αλγορίθμους για ένα πρόβλημα, χρειαζόμαστε ένα μέτρο αποδοτικότητας, ανεξάρτητα από τη γλώσσα και τον υπολογιστή που χρησιμοποιούμε.

Αυτό το μέτρο είναι ο αριθμός των βασικών πράξεων ή ακόμη των εντολών του αλγορίθμου, το οποίο ονομάζουμε πολυπλοκότητα του αλγορίθμου (algorithm complexity). Πρόκειται για έναν όρο ο οποίος δεν σχετίζεται με την πολυπλοκότητα της δομής ή της δυσκολίας προγραμματισμού.

Ορισμός 1.2.1 *Η πολυπλοκότητα ενός αλγορίθμου A είναι μία συνάρτηση $C_A(n)$, που δίνει τον αριθμό των χαρακτηριστικών εντολών που εκτελούνται από τον A στη χείριστη περίπτωση για δεδομένα διάστασης n.*

Η πολυπλοκότητα είναι μία συνάρτηση της διάστασης των δεδομένων για να επιτρέπει την πρόβλεψη διακύμανσης του χρόνου υπολογισμού, παραδείγματος χάριν όταν περνάμε από μικρά παραδείγματα ελέγχου σε μεγάλα πραγματικά στιγμιότυπα του προβλήματος.

Αυτό επιτρέπει επίσης να εκτιμήσουμε τη μέγιστη διάσταση των προβλημάτων που μπορούμε να επιλύσουμε στην πράξη.

Ένα δεδομένο πρέπει να θεωρηθεί εδώ με μία έννοια ομαδική. Ένα δεδομένο ενός αλγορίθμου εύρεσης του βέλτιστου μονοπατιού σ΄ ένα γράφο με βάρη $G = (V, E, W)$ θα περιέχει τον αριθμό n των κόμβων, τη λίστα των κόμβων V, τον αριθμό E των πλευρών και τη λίστα των πλευρών με την αριθμητική τιμή του βάρους για κάθε μία πλευρά.

Η διάσταση ενός δεδομένου είναι η απαιτούμενη ποσότητα μνήμης για την αποθήκευσή του. Για να είμαστε ακριβείς, θα έπρεπε να τη μετρήσουμε σε αριθμό bits. Γενικά όμως, αρκεί να μετρήσουμε τον αριθμό των λέξεων-μνήμης. Εξαρτώμενης της περίπτωσης, αυτές οι λέξεις μπορούν να είναι ακέραιοι (shorts or longs), πραγματικοί κ.λ.π. Έτσι ένα δεδομένο για το πρόβλημα της ταξινόμησης n ακεραίων μπορεί να κωδικοποιηθεί σαν ένας ακέραιος n, ακολουθούμενος από τους n προς ταξινόμηση ακεραίους. Η διάσταση των δεδομένων σε αυτή την περίπτωση είναι $n + 1$ λέξεισ-μνήμης.

Για τους γράφους, συνηθίζεται να εκφράζουμε τη διάσταση με τον αριθμό των κόμβων n και τον αριθμό των πλευρών m. Παραδείγματος χάριν, ο αλγόριθμος Dijkstra, για την εύρεση ενός μονοπατιού ελάχιστου κόστους σε έναν γράφο, έχει πολυπλοκότητα της τάξης n^2 αν τον προγραμματίσουμε χωρίς ιδιαίτερη προσοχή. Χρησιμοποιώντας όμως τη δομή δεδομένων σωρός (heap), η πολυπλοκότητα είναι της τάξης $m \log n$, η οποία είναι καλύτερη για την περίπτωση αραιών γράφων όπου $m = O(n)$.

1.2.1 Τάξη μίας συνάρτησης

Ας θεωρήσουμε δύο αλγορίθμους A_1 και A_2 για ένα πρόβλημα διάστασης n, με πολυπλοκότητα αντίστοιχα $C_{A_1}(n) = 0.5n^2$ και $C_{A_2}(n) = 5n$. Ο A_2 κάνει περισσότερες πράξεις από τον A_1, για $n = 5$ (π.χ. $C_{A_1}(5) = 12.5$, $C_{A_2}(5) = 25$) αλλά λιγότερες για $n \geq 10$ (π.χ. $C_{A_1}(20) = 200$, $C_{A_2}(20) = 100$). Πράγματι, οποιοιδήποτε και να είναι οι θετικοί συντελεστές του n και n^2 στις πολυπλοκότητες $C_{A_1}(n)$ και $C_{A_2}(n)$, ο αλγόριθμος A_2 θα εκτελείται πιο γρήγορα από μία ορισμένη και πάνω διάσταση ($\forall n \geq n_0$) του προβλήματος. Αυτή η ιδέα της ασυμπτωτικής αύξησης εκφράζεται από την τάξη μίας συνάρτησης.

Ορισμός 1.2.2 *Έστω f, g δύο συναρτήσεις από $\mathbb{R}$ στο $\mathbb{R}$. Θα λέμε ότι η f είναι τάξης μικρότερης ή ίσης της g, ή τάξης το πολύ g, αν μπορούμε να βρούμε έναν πραγματικό αριθμό x_0 και έναν πραγματικό θετικό c έτσι ώστε: $\forall x \geq x_0, f(x) \leq c \cdot g(x)$.*

Με άλλα λόγια η συνάρτηση g γίνεται πιο μεγάλη από την f κατά προσέγγιση ενός συντελεστή c, για όλες τις τιμές του x που είναι μεγαλύτερες από μία τιμή x_0. Γράφουμε: f είναι $O(g)$ ή $f = O(g)$. Αν f και g επιδέχονται όρια, τότε έχουμε άλλους ισοδύναμους ορισμούς.

Ορισμός 1.2.3 *Αν $\lim_{n\to\infty} \frac{f}{g} = c > 0$ τότε f και g είναι της ίδιας τάξης, $f = O(g)$ και $g = O(f)$, συμβολιζόμενο $f = \Theta(g)$.*

Ορισμός 1.2.4 *Αν $\lim_{n\to\infty} \frac{f}{g} = 0$, τότε $f = O(g)$.*

Ορισμός 1.2.5 *Αν $\lim_{n\to\infty} \frac{f}{g} = +\infty$ τότε η f είναι μεγαλύτερης τάξης από g και συμβολίζουμε $f = \Omega(g)$ (ή $g = O(f)$).*

Μπορούμε να παρατηρήσουμε τα εξής:

- $f = O(g)$ σημαίνει ότι g είναι ένα φράγμα της πραγματικής πολυπλοκότητας.
- Έχουμε ότι $10n^2 = O(0.5n^2)$ και αντιστρόφως. Δηλαδή δύο συναρτήσεις οι οποίες διαφέρουν κατά ένα σταθερό συντελεστή είναι της ίδιας τάξης. Αυτό επιτρέπει να ξεπεράσουμε τη διαφορετική σχετική ισχύ δύο μηχανών.
- Έχουμε ότι $n^2 + 3n + 4 = O(n^2)$. Δηλαδή ότι οι όροι μικρότερης τάξης μπορούν να αγνοηθούν. Έτσι, για n αρκετά μεγάλο, η αρχικοποίηση ενός πίνακα T από n στοιχεία σε χρόνο $O(n)$ είναι αμελητέα σε σχέση δύο ενσωματωμένους βρόχους από n επαναλήψεις στον T, που στοιχίζουν $O(n^2)$.
- Δεν είναι αναγκαίο να είμαστε ιδιαίτερα ακριβείς, στην έννοια της χαρακτηριστικής πράξης ενός αλγορίθμου. Ο ακριβής αριθμός των εντολών μέσα σε ένα βρόχο, π.χ., θα είναι χωρίς συνέπεια στην πολυπλοκότητα.

1.2.2 Αποδοτικοί και μη αποδοτικοί αλγόριθμοι

Μία κατηγοριοποίηση, γενικά αποδεκτή, διακρίνει τους αλγορίθμους σε πολυωνυμικούς (πολυπλοκότητας της τάξης ενός πολυωνύμου) και σε εκθετικούς. Παραδείγματα πολυωνυμικής πολυπλοκότητας είναι $\log\log n, \log n, \sqrt{n}, n, n\log n, n^2, n^2\log n, n^3, \ldots$. Μία εκθετική πολυπλοκότητα μπορεί να είναι είτε μία αληθινή, ως προς τη μαθηματική έννοια, εκθετική συνάρτηση $(e^n, 2^n)$, είτε μια συνάρτηση όπως η $n^{\log n}$ (sub-exponential), η παραγοντική συνάρτηση $n!$ και η n^n.

Ένας αποδοτικός αλγόριθμος είναι πολυωνυμικός. Οι εκθετικοί αλγόριθμοι χρειάζονται υπερβολικό χρόνο, ιδιαίτερα όταν το μέγεθος των δεδομένων αυξάνει και θα πρέπει να είμαστε δύσπιστοι ως προς την αποτελεσματικότητά τους.

1.2.3 Θεωρία πολυπλοκότητας

Μερικά προβλήματα βελτιστοποίησης έχουν επιλυθεί εδώ και πολύ καιρό με πολυωνυμικούς αλγορίθμους, ενώ μερικά άλλα όχι. Τα πρώτα χαρακτηρίζονται εύκολα και τα δεύτερα δύσκολα. Το ερώτημα που τίθεται είναι: '' *Υπάρχει πραγματικά μία κλάση προβλημάτων για τα οποία δε θα βρούμε ποτέ πολυωνυμικούς αλγορίθμους, ή μήπως προβλήματα χαρακτηριζόμενα δύσκολα επιδέχονται πολυωνυμικούς αλγορίθμους, αλλά όχι ακόμη ανακαλυφθέντες;* ''.

Η θεωρία πολυπλοκότητας αναπτύχθηκε γύρω στο 1970 για να απαντήσει σε αυτή την ερώτηση, από ειδικούς της θεωρητικής πληροφορικής.

Κεφάλαιο 2

Προβλήματα Συνδυαστικής Βελτιστοποίησης

Ένα Πρόβλημα Συνδυαστικής Βελτιστοποίησης (ΠΣΒ) συνίσταται στην αναζήτηση του βέλτιστου (μεγίστου ή ελαχίστου) s^* μιας συνάρτησης f, συνήθως ακεραίων ή πραγματικών τιμών, ορισμένη πάνω σε ένα πεπερασμένο σύνολο S:

$$f(s^*) = \underset{s \in S}{opt} \{f(s)\} \text{ με } opt = \max \text{ή} \min$$

όπου f είναι η αντικειμενική συνάρτηση (objective function, cost function). Το σύνολο S και η συνάρτηση f ορίζουν ένα στιγμιότυπο (instance) του προβλήματος. Στον ορισμό μπορούμε να θεωρήσουμε πάντα $opt \equiv \min$, παρατηρώντας ότι $\max f$ ισοδυναμεί με $\min -f$.

Τυπικά για ένα πρόβλημα Π ορίζουμε ως D_Π το σύνολο όλων των στιγμιοτύπων του. Για κάθε στιγμιότυπο $x \in D_\Pi$ υπάρχει ένα σύνολο $S_\Pi(x)$ ή απλά S_Π λύσεων που του αντιστοιχούν.

Ορισμός 2.0.1 *Ένα πρόβλημα, καλείται πρόβλημα συνδυαστικής βελτιστοποίησης αν* $\forall x \in S_\Pi, \exists\ f_\Pi(s,x)$ *η οποία ονομάζεται συνάρτηση κόστους. Το πρόβλημα δηλαδή, δοθέντος ενός στιγμιότυπου* $x \in D_\Pi$*, συνίσταται στην εύρεση της ολικά βέλτιστης λύσης (global optimum)* $s^* \in S_\Pi$ *έτσι ώστε* $f_\Pi(s^*,x) \leq f_\Pi(s,x), \forall s \in S_\Pi$*, δηλαδή η συνάρτηση κόστους να είναι ελάχιστη (global minimum), ή* $f_\Pi(s^*,x) \geq f_\Pi(s,x), \forall s \in S_\Pi$*, δηλαδή η συνάρτηση κόστους να είναι μέγιστη (global maximum).*

Συνήθως ενδιαφερόμαστε για τα σημεία του S (λύσεις) που ικανοποιούν κάποιους περιορισμούς του προβλήματος. Τα σημεία αυτά ονομάζονται εφικτές λύσεις (feasible solutions) του προβλήματος. Οι λύσεις που δεν ικανοποιούν τους περιορισμούς του προβλήματος χαρακτηρίζονται ως μη εφικτές λύσεις (infeasible solutions).

Τα προβλήματα συνδυαστικής βελτιστοποίησης είναι πολύ σημαντικά, όχι μόνο στην επιστήμη της πληροφορικής, όπου εμφανίζονται είτε αυτούσια είτε ως υποπροβλήματα σε πολλές εφαρμογές, π.χ. τηλεπικοινωνίες, χρονοπρογραμματισμός εργασιών, παράλληλα και κατανεμημένα συστήματα, αναγνώριση εικόνας, βάσεις δεδομένων κ.α., αλλά και σε άλλους τομείς όπως τα οικονομικά, η βιομηχανία, η αξιοπιστία συστημάτων παραγωγής κ.α. Συνεπώς η σπουδαιότητά τους μας οδηγεί στην ανάγκη εύρεσης αποδοτικών αλγορίθμων ως προς τον χρόνο και την ποιότητα της λύσης, όταν αυτή δεν είναι βέλτιστη.

2.1 Προβλήματα ύπαρξης

Η θεωρία πολυπλοκότητας για απλοποίηση της μελέτης και τη χρησιμοποίηση εργαλείων της μαθηματικής λογικής έχει βασιστεί στα προβλήματα ύπαρξης.

Ορισμός 2.1.1 *Σε κάθε πρόβλημα βελτιστοποίησης αντιστοιχεί ένα πρόβλημα ύπαρξης. Αρκεί να προστεθεί στα δεδομένα S και f ένας ακέραιος αριθμός k. Τότε δεν ψάχνουμε για μία λύση ελαχίστου κόστους, αλλά μία λύση κόστους το πολύ k. Δηλαδή το ερώτημά μας γίνεται, υπάρχει λύση s με κόστος $f(s) \leq k$;*

Η χρησιμοποίηση του προβλήματος ύπαρξης στη μελέτη της θεωρίας πολυπλοκότητας βέβαια δεν εμπεριέχει κανένα κίνδυνο για τα προβλήματα βελτιστοποίησης. Ένας αλγόριθμος αποδοτικός για ένα πρόβλημα βελτιστοποίησης Π επιλύει προφανώς το αντίστοιχο πρόβλημα ύπαρξης Q. Επίσης ένας αλγόριθμος αποδοτικός για το Q μπορεί να χρησιμοποιηθεί για να επιλύσει αποδοτικά το Π, με διχοτόμηση πάνω στις δυνατές τιμές της αντικειμενικής συνάρτησης.

Ένα πρόβλημα βελτιστοποίησης είναι λοιπόν το λιγότερο το ίδιο δύσκολο όσο το αντίστοιχό του πρόβλημα ύπαρξης. Επομένως, οποιοδήποτε αποτέλεσμα επιβεβαιώνουν τη δυσκολία του δεύτερου θα αφορά a fortiori το πρώτο.

2.2 Οι κλάσεις P και NP

Το σύνολο των προβλημάτων ύπαρξης που επιδέχονται πολυωνυμικούς αλγορίθμους συνιστούν την κλάση P. Αυτή η κλάση εμπεριέχει όλα τα προβλήματα ύπαρξης που αντιστοιχούν σε προβλήματα βελτιστοποίησης εύκολα, π.χ. συντομότερο μονοπάτι σε ένα γράφο, μέγιστη ροή, δένδρα επικάλυψης ελαχίστου ή μεγίστου βάρους (minimum or maximum spanning trees) κ.λ.π.

Η θεωρία πολυπλοκότητας επικεντρώνεται στα προβλήματα ύπαρξης που έχουν ένα πρακτικό ενδιαφέρον. Το κριτήριο πρακτικότητας, που έχει επιλεγεί, είναι η δυνατότητα εξακρίβωσης σε πολυωνυμικό χρόνο της απάντησης ΝΑΙ για ένα πρόβλημα ύπαρξης. Αυτό το κριτήριο ονομάζεται συνήθως αρχή του επιβλέποντος (supervisor): έχοντας βρει ότι η λύση σε ένα πρόβλημα ύπαρξης είναι ΝΑΙ (με ένα πολυωνυμικό ή ΟΧΙ αλγόριθμο), μπορούμε να πείσουμε ''εύκολα'' (πολυωνυμικά) ένα άλλο άτομο (supervisor) το οποίο δεν έψαξε τη λύση;

Η κλάση NP εμπεριέχει τα προβλήματα ύπαρξης των οποίων μία προτασσόμενη λύση ΝΑΙ είναι εξακριβώσιμη πολυωνυμικά. Προβλήματα που δεν είναι μέσα στην κλάση NP υπάρχουν, αλλά τα περισσότερα δεν ενδιαφέρουν παρά μόνο θεωρητικά. Προφανώς, η κλάση NP εμπεριέχει την κλάση P, για την οποία η εξακρίβωση είναι εύκολη: αφού ένα πρόβλημα της κλάσης P επιδέχεται έναν αλγόριθμο πολυωνυμικό, ο επιβλέπων θα μπορεί να εκτελέσει αυτό τον αλγόριθμο για να διαπιστώσει αν επιτυγχάνει την ίδια λύση με την προτεινόμενη. Επομένως κάθε πρόβλημα ύπαρξης με πολυωνυμικό αλγόριθμο είναι μέσα στην κλάση P, και άρα στην κλάση NP (σχήμα 2.1).

Για ένα πρόβλημα χωρίς γνωστό αποδοτικό αλγόριθμο, πρέπει να κάνουμε τα ακόλουθα για να αποδείξουμε την έγκλισή του στην κλάση NP.

- να προτείνουμε μία κωδικοποίηση της λύσης.
- να προτείνουμε έναν αλγόριθμο που θα εξακριβώνει τη λύση σε σχέση με τα δεδομένα και την κωδικοποίηση.
- να αποδείξουμε ότι ο αλγόριθμος εξακρίβωσης είναι πολυωνυμικός.

2.3 Προβλήματα NP-complete

Πρόκειται για τα πιο δύσκολα προβλήματα της κλάσης NP, δηλαδή ο ''σκληρός πυρήνας'' κατά κάποιον τρόπο (σχήμα 2.1). Η έννοια του προβλήματος NP-complete βασίζεται σε αυτή του πολυωνυμικού μετασχηματισμού ενός προβλήματος.

Ένα πρόβλημα ύπαρξης P_1 μετασχηματίζεται πολυωνυμικά σε ένα άλλο πρόβλημα P_2 αν υπάρχει ένας πολυωνυμικός αλγόριθμος A, που μετασχηματίζει κάθε δεδομένο του P_1 σε ένα δεδομένο του P_2 διατηρώντας την απάντηση ΝΑΙ ή ΟΧΙ.

Ένα πρόβλημα NP-Complete είναι ένα πρόβλημα της κλάσης NP, στο οποίο μετασχηματίζεται πολυωνυμικά κάθε άλλο πρόβλημα της κλάσης NP.

Ένα πρόβλημα συνδυαστικής βελτιστοποίησης είναι NP-hard αν το αντίστοιχό του, πρόβλημα ύπαρξης είναι NP-complete.

Τα προβλήματα NP-complete αντιπροσωπεύουν το σκληρό πυρήνα της κλάσης NP, γιατί αν βρίσκαμε έναν πολυωνυμικό αλγόριθμο A για ένα μόνο NP-complete πρόβλημα X, θα μπορούσαμε εξ αυτού να βρούμε έναν άλλο αλγόριθμο για κάθε άλλο δύσκολο πρόβλημα Y της κλάσης NP. Πράγματι, αυτός ο αλγόριθμος θα συνίσταται στον πολυωνυμικό μετασχηματισμό των δεδομένων του Y σε δεδομένα του X, και στη συνέχεια στην εκτέλεση του αλγορίθμου για το πρόβλημα X.

Το 1970, ο Cook απέδειξε ότι το πρόβλημα SAT (satisfiability)[1] είναι NP-complete: κάθε άλλο πρόβλημα της κλάσης NP μπορεί να μετασχηματιστεί πολυωνυμικά στο SAT. Από τότε, αποδείχθηκε ότι τα περισσότερα προβλήματα ύπαρξης, που αντιστοιχούν σε προβλήματα συνδυαστικής βελτιστοποίησης χωρίς γνωστούς πολυωνυμικούς αλγορίθμους, είναι NP-complete.

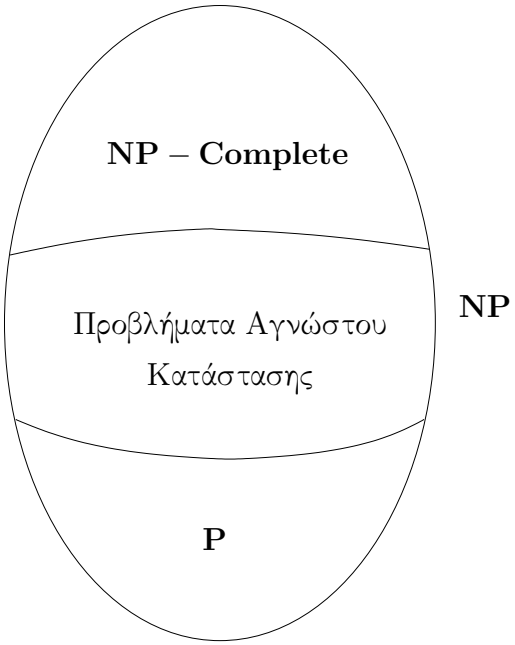
[1]SAT: Δίνεται μια λογική formula π.χ. $\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$. Υπάρχει ανάθεση των μεταβλητών x_i ώστε $\phi = True$;

2.4 Η εικασία $P \neq NP$

Η κρίσιμη ερώτηση της θεωρίας πολυπλοκότητας είναι να γνωρίσουμε αν τα NP-complete προβλήματα μπορούν να επιλυθούν πολυωνυμικά και επομένως $P = NP$, ή αν δεν θα βρούμε ποτέ πολυωνυμικούς αλγορίθμους, οπότε $P \neq NP$. Αυτή η ερώτηση παραμένει αναπάντητη.

Η ανακάλυψη ενός πολυωνυμικού αλγορίθμου για ένα μόνο πρόβλημα NP-complete θα επέτρεπε να επιλύσουμε εύκολα όλα τα υπόλοιπα. Επειδή όμως πάρα πολλά προβλήματα αντιστέκονται σθεναρά, εικάζουμε σήμερα ότι $P \neq NP$. Η οριστική απόδειξη αυτής της εικασίας παραμένει ένα ανοικτό ερώτημα και θα απαιτήσει πιθανώς την ανακάλυψη νέων μαθηματικών.

Μπορούμε να παρουσιάσουμε τη γεωγραφία της κλάσης NP όπως στο σχήμα 2.1. Στο κάτω τμήμα, τα πιο εύκολα προβλήματα σχηματίζουν την κλάση P. Επάνω ευρίσκονται τα προβλήματα NP-complete. Ανάμεσα στα δύο εκτείνεται ο χώρος μερικών προβλημάτων άγνωστου ακόμη χαρακτηρισμού, όπως για παράδειγμα το πρόβλημα ισομορφισμού δύο γράφων.



Σχήμα 2.1: Η κλάση NP των προβλημάτων ύπαρξης. Η υποκλάση P περιέχει τα πολυωνυμικά προβλήματα. Η υποκλάση NP-complete περιέχει τα δυσεπίλυτα προβλήματα.

Κεφάλαιο 3

Γραμμικός προγραμματισμός

Πολλά προβλήματα μπορούν να μοντελοποιηθούν ως μεγιστοποίηση ή ελαχιστοποίηση μιας αντικειμενικής συνάρτησης δεδομένων κάποιων περιορισμένων πόρων και ανταγωνιστικών περιορισμών. Αν μπορούμε να ορίσουμε την αντικειμενική συνάρτηση σαν μια γραμμική συνάρτηση συγκεκριμένων μεταβλητών και τους περιορισμούς σαν ισότητες ή ανισότητες αυτών των μεταβλητών, τότε έχουμε ένα πρόβλημα γραμμικού προγραμματισμού. Έτσι σε ένα πρόβλημα γραμμικού προγραμματισμού η αντικειμενική συνάρτηση (objective function ή cost function) και οι περιορισμοί (constraints) είναι γραμμικά.

Ας θεωρήσουμε το ακόλουθο πρόβλημα (βλ. πίνακα 3.1):

Μία εταιρία που διαθέτει 30 μηχανικούς, 24 τεχνικούς και 18h/ημέρα χρόνου υπολογιστή προτίθεται να επιλέξει ανάμεσα σε δύο τύπους συμβολαίων:

- Έναν αριθμό συμβολαίων του πρώτου τύπου το οποίο απαιτεί 5 μηχανικούς, 2 τεχνικούς και 1h υπολογιστή για ένα κέρδος 8 νομισματικών μονάδων ανά συμβόλαιο.

- Έναν αριθμό συμβολαίων του δεύτερου τύπου το οποίο απαιτεί 3 μηχανικούς, 3 τεχνικούς και 3h υπολογιστή για ένα κέρδος 6 νομισματικών μονάδων ανά συμβόλαιο.

Επί του παρόντος η εταιρία έχει περισσότερες προσφορές συμβολαίων απ΄ ότι μπορεί να ικανοποιήσει. Η εταιρεία θα πρέπει να αποφασίσει πόσα συμβόλαια τύπου ένα και πόσα τύπου δύο θα πρέπει να αποδεχθεί, κάτω από τους περιορισμούς των διαθέσιμων

πόρων. Επομένως θα πρέπει η εταιρία να αναζητήσει την καλύτερη λύση (optimum) ανάμεσα σε όλες τις δυνατές λύσεις. Στη συνέχεια θα πρέπει να αναλύσει τις συνέπειες στην ευρεθείσα λύση όταν υπεισέρχονται ελαφρές τροποποιήσεις μερικών δεδομένων, αλλά αυτός ο προβληματισμός δεν θα μας απασχολήσει εδώ.

3.1 Μοντελοποίηση – συνθήκες επίλυσης

Το πρώτο βήμα είναι:

- Να ποσοτικοποιήσουμε τον ζητούμενο στόχο και
- Να περιγράψουμε τους περιορισμούς (constraints) κάτω από τους οποίους λειτουργεί η εταιρία.

Γι΄ αυτό πρέπει να ορίσουμε:

- τις μεταβλητές που αντιπροσωπεύουν ποσότητες πόρων ή δραστηριοτήτων.
- μία αντικειμενική συνάρτηση ή συνάρτηση κόστους (ή κέρδους στην περίπτωσή μας).
- τους περιορισμούς που καθορίζουν τις δραστηριότητες.

Για να μπορέσουμε να χρησιμοποιήσουμε το γραμμικό προγραμματισμό πρέπει: η αντικειμενική συνάρτηση και οι περιορισμοί να είναι γραμμικοί σχετικά με τις μεταβλητές, δηλαδή κάθε στοιχειώδης δραστηριότητα να έχει ένα κόστος (μέσα στην αντικειμενική συνάρτηση) και ένα συντελεστή (μέσα στους περιορισμούς) σταθερό και ανεξάρτητο των άλλων δραστηριοτήτων και των τιμών τους.

Τα δεδομένα του προβλήματος μπορούν να ανακεφαλαιωθούν με τον παρακάτω πίνακα:

	Συμβόλαιο Τύπου 1	Συμβόλαιο Τύπου 2	Περιορισμοί
Μηχανικοί	5	3	30
Τεχνικοί	2	3	24
Ώρες υπολογιστή	1	3	18
Κέρδος	8	6	

Πίνακας 3.1: Δεδομένα του προβλήματος της εταιρείας

Αναζητούμε να επιτύχουμε το μέγιστο κέρδος, γι΄ αυτό δοκιμάζουμε να βρούμε ένα συνδυασμό από συμβόλαια του πρώτου τύπου και του δεύτερου τύπου, που αποδίδουν το μέγιστο κέρδος.

Οι μεταβλητές του προβλήματος είναι x_1 ο αριθμός συμβολαίων πρώτου τύπου και x_2 αριθμός συμβολαίων δεύτερου τύπου. Λογικά θα πρέπει να έχουμε $x_1, x_2 \in \mathbb{N}$, αλλά θα κάνουμε την απλοποιημένη υπόθεση προς το παρών ότι $x_1, x_2 \in \mathbb{R}$ δηλαδή αποδεχόμαστε να υπογράψουμε μέρη συμβολαίων.

Η αντικειμενική συνάρτηση η οποία εκφράζει το κέρδος της εταιρείας ορίζεται ως

$$z = 8x_1 + 6x_2.$$

Επομένως θα αναζητήσουμε την μεγιστοποίησή της.

Ένα συμβόλαιο τύπου 1 απαιτεί 2 τεχνικούς και επομένως x_1 συμβόλαια τύπου 1 απαιτούν $2x_1$ τεχνικούς, ενώ τα συμβόλαια τύπου 2 απαιτούν $3x_2$ τεχνικούς. Επομένως θα έχουμε τον περιορισμό

$$2x_1 + 3x_2 \leq 24$$

Ομοίως για τους μηχανικούς θα έχουμε:

$$5x_1 + 3x_2 \leq 30$$

και για τις ώρες υπολογισμού:

$$x_1 + 3x_2 \leq 18$$

Το πρόβλημα επομένως μοντελοποιείται ως εξής:

$$\begin{aligned} \max z = \quad & 8x_1 + 6x_2 && (1)\\ & 5x_1 + 3x_2 \leq 30 && (2)\\ & 2x_1 + 3x_2 \leq 24 && (3)\\ & x_1 + 3x_2 \leq 18 && (4)\\ & x_1, x_2 \geq 0 && (5) \end{aligned}$$

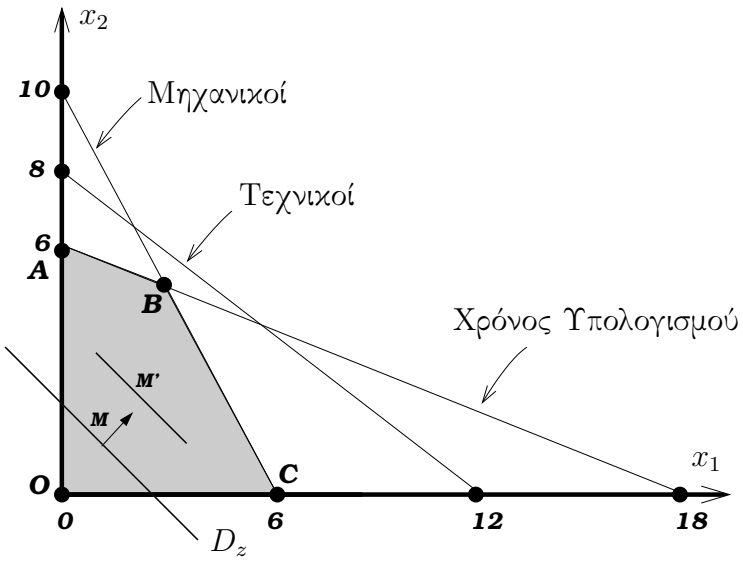
3.2 Γραφική επίλυση

Ας επανέλθουμε τώρα στο παράδειγμά μας. Κάθε περιορισμός (2 μέχρι 5) παριστάνεται από μία ευθεία στο $\mathbb{R}^2$ και καθορίζει ένα ημιεπίπεδο που τον ικανοποιεί.

Το σύνολο E των σημείων του πολυέδρου $OABC$ στο σχήμα 3.1 ικανοποιούν όλους τους περιορισμούς και είναι λοιπόν λύσεις του προβλήματος. Πρέπει να βρούμε ''την'' ή ''τις'' καλύτερες λύσεις ανάμεσα στα σημεία που ανήκουν στο E. Υπενθυμίζουμε ότι x_1, x_2 προς το παρών παίρνουν πραγματικές μη αρνητικές τιμές.

Η ευθεία $z = 8x_1 + 6x_2$ καθορίζει μία οικογένεια από ευθείες D_z παράλληλες μεταξύ τους. Ανάμεσα σε όλες τις ευθείες D_z που συναντούν το πολύγωνο $OABC$, αυτή η ευθεία για την οποία η $z = 8x_1 + 6x_2$ παίρνει τη μέγιστη τιμή, δίδει τη βέλτιστη λύση του προβλήματος. Αυτή η λύση επιτυγχάνεται σε ένα κόμβο του πολυγώνου, εδώ ο κόμβος B.

Πράγματι, έστω M ένα σημείο εσωτερικό του $OABC$. Μπορούμε να περάσουμε από αυτό το σημείο μία ευθεία D_{z_m} η οποία χωρίζει το πολύγωνο σε δύο περιοχές. Στην πρώτη περιοχή περιέχουσα την αρχή των αξόνων, όλα τα σημεία της δίνουν στην αντικειμενική συνάρτηση μία τιμή μικρότερη από αυτήν στο M.



Σχήμα 3.1: Το πολύγωνο $OABCO$ περιέχει όλες τις εφικτές λύσεις.

Η δεύτερη που είναι μη κενή, περιέχει το σύνολο των σημείων που δίνουν τιμή καλύτερη στην αντικειμενική συνάρτηση, αυτό είναι ειδικά αληθές για ένα σημείο M' εσωτερικά της δεύτερης περιοχής, και είναι δυνατό να ξαναρχίσουμε αυτό το συλλογισμό. Έτσι δείχνουμε ότι δεν είναι δυνατόν το βέλτιστο να είναι σημείο εσωτερικό.

Ας δείξουμε παρομοίως ότι δεν είναι δυνατόν να ανήκει, εκτός εξαίρεσης, σε μία πλευρά του πολυγώνου. Έστω M ένα σημείο ανάμεσα σε δύο κόμβους του πολυγώνου. Μία ευθεία D_z περνάει από το M η οποία χωρίζει πάλι το πολύγωνο σε δύο περιοχές, και οδηγούμαστε έτσι στην προηγούμενη περίπτωση.

Η μόνη εξαίρεση είναι όταν η αντικειμενική συνάρτηση είναι παράλληλη σε έναν περιορισμό. Σε αυτή την περίπτωση, υπάρχει μία απειρία λύσεων οι οποίες είναι βέλτιστες.

Μία άλλη ειδική περίπτωση είναι όταν το πολύγωνο (πολύεδρο στη γενική περίπτωση) είναι ''ανοικτό'' τότε η βέλτιστη λύση εκτείνεται στο άπειρο. Αργότερα θα δούμε τέτοια παραδείγματα.

Επιστρέφοντας στο παράδειγμά μας παρατηρούμε ότι η βέλτιστη λύση στο σημείο $B = (x_1, x_2)$ είναι μια ακέραια λύση $x_1, x_2 \in \mathbb{N}$ όπως θα επιθυμούσαμε για το πρόβλημά μας. Αυτό όμως είναι μια ευτυχής σύμπτωση!

3.3 Πολύεδρο - Κυρτότητα - Ακρότατα Σημεία

Ας θυμηθούμε σε αυτό το σημείο μερικούς ορισμούς.

- *Ευθεία*: Έστω $c \in \mathbb{R}^n$ και $c \neq 0$. Η ευθεία Δ που περνά από την αρχή των αξόνων O και παράγεται από το σημείο c είναι: $\Delta = \{x \in \mathbb{R}^n | x = \lambda c\}$, όπου $\lambda \in \mathbb{R}$.

- *Υπερεπίπεδο*: Έστω $a \in \mathbb{R}, c \in \mathbb{R}^n$. Το υπερεπίπεδο P ορίζεται ως εξής: $P = \{x \in \mathbb{R}^n | cx = a\}$

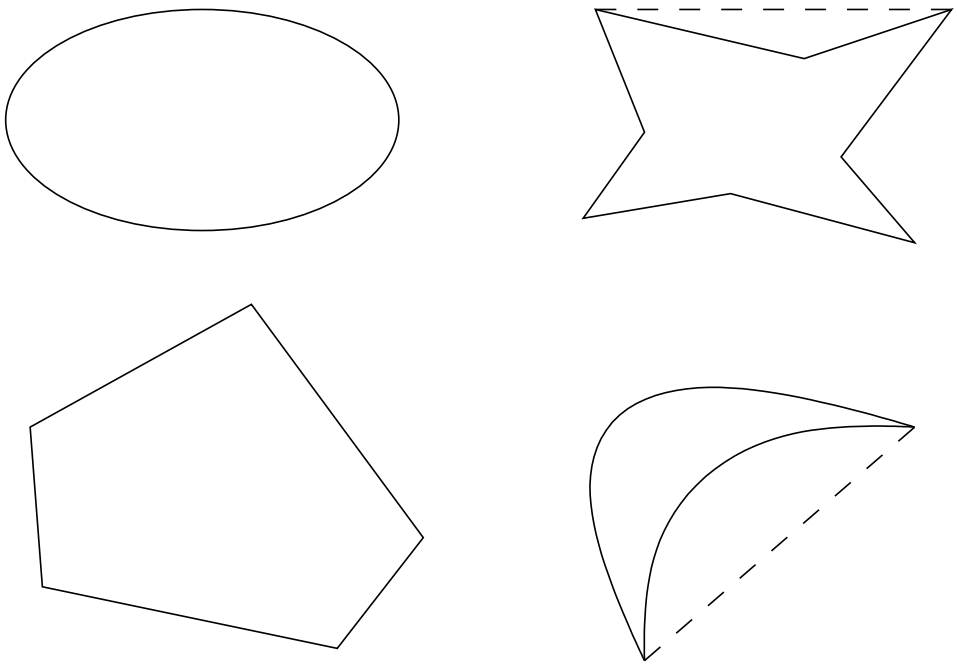
- *Ημιχώρος*: Έστω $a \in \mathbb{R}, c \in \mathbb{R}^n, c \neq 0$. Ο ημιχώρος ορίζεται από: $E = \{x \in \mathbb{R}^n | cx > a\}$

- *Ημιχώρος κλειστός*: Έστω $a \in \mathbb{R}, c \in \mathbb{R}^n, c \neq 0$. Ο κλειστός ημιχώρος F ορίζεται από: $F = \{x \in \mathbb{R}^n | cx \geq a\}$

- *Κυρτό σύνολο*: Ένα σύνολο C είναι κυρτό σύνολο αν και μόνο αν, η μία από τις δύο παρακάτω συνθήκες ικανοποιείται:

1. $\forall x_1, x_2 \in C, \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1 \Rightarrow \lambda x_1 + (1-\lambda)x_2 \in C$
2. $\forall p \in \mathbb{N},\ x_i \in C,\ i = 1, \ldots, p$ και $\mu_i \in \mathbb{R},\ \sum_{i=1}^{p} \mu_i = 1,\ \mu_i \geq 0,\ i = 1, \ldots, p$ έχουμε $\sum_{i=1}^{p} \mu_i x_i \in C$.

Με άλλα λόγια, εάν ένα κυρτό υποσύνολο περιέχει δύο σημεία, τότε θα περιέχει και το ευθύγραμμο τμήμα που έχει άκρα αυτά τα δύο σημεία (σχήμα 3.2).



Κυρτά Σύνολα Μη Κυρτά Σύνολα

Σχήμα 3.2: Αριστερά έχουμε δύο κυρτά σύνολα και δεξιά δύο μη κυρτά σύνολα.

- *Ακρότατο σημείο*: Ένα σημείο x είναι ακρότατο σημείο ενός κυρτού συνόλου C αν δεν είναι δυνατό να εκφραστεί σαν γραμμικός συνδυασμός, με μη μηδενικούς συντελεστές, δύο διακεκριμένων σημείων του κυρτού συνόλου.

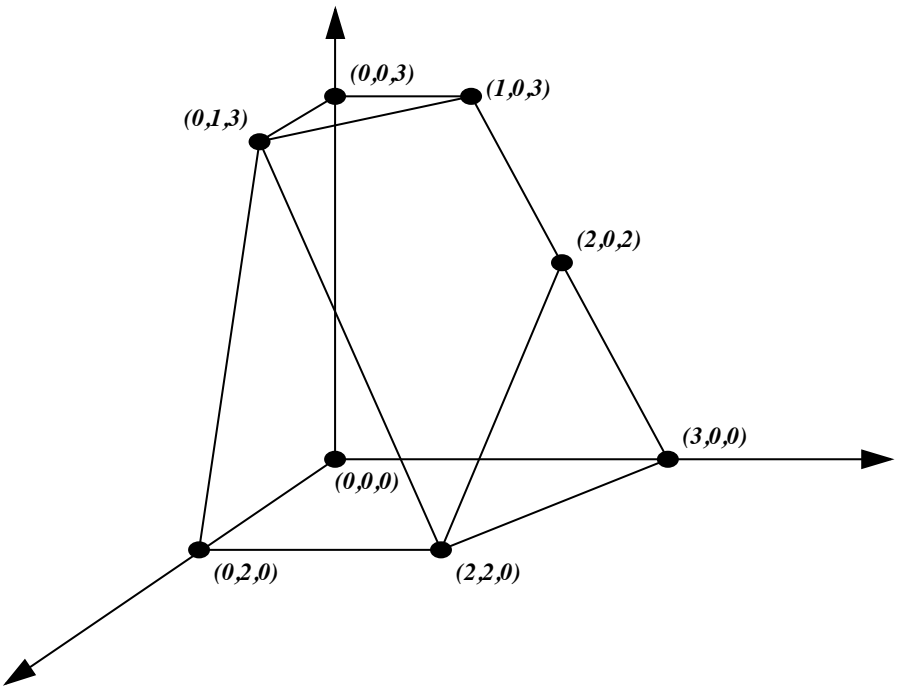
$$x \text{ ακρότατο σημείο του } C \Leftrightarrow \nexists x_1 \text{ και } x_2 \in C \text{ έτσι ώστε:}$$
$$x = \lambda x_1 + (1-\lambda)x_2, \text{ με } \lambda \in (0,1).$$

- *Πολύεδρο*: Ένα πολύεδρο P ορίζεται ως εξής: $P = \{x \in \mathbb{R}^n | Ax \geq \alpha\}$, όπου A είναι ένας $m \times n$ πίνακας και α ένα m-διάνυσμα κολόνα.

  Το P είναι λοιπόν η τομή ενός πεπερασμένου αριθμού κλειστών ημιχώρων (σχήμα 3.3).

  Παρατήρηση: Ένα πολύεδρο P είναι κυρτό. Πράγματι:

  $\forall x_1 \in P, x_2 \in P$ έχουμε $Ax_1 \geq a$ και $Ax_2 \geq a$
  $\forall \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1$ έχουμε ότι
  $\lambda Ax_1 + (1-\lambda)Ax_2 \geq \lambda a + (1-\lambda)a = a.$
  Άρα $A(\lambda x_1 + (1-\lambda)x_2) \geq a$ και επομένως
  $\lambda x_1 + (1-\lambda)x_2 \in P$



Σχήμα 3.3: Πολύεδρο

Το παρακάτω θεώρημα είναι σημαντικό για την επίλυση προβλημάτων γραμμικού προγραμματισμού.

3.4 Θεώρημα βέλτιστου

Κάθε γραμμική συνάρτηση $f : \mathbb{P} \to \mathbb{R}, \quad \mathbb{P} \subseteq \mathbb{R}^n$, επιτυγχάνει τη βέλτιστη τιμή της σε ένα ακρότατο σημείο του P.

Απόδειξη

Έστω $x \in \mathbb{P}$ και $y_i,\ i = 1, \ldots, k$ τα k ακρότατα σημεία του $\mathbb{P}$. Το σημείο x μπορεί να γραφεί σαν γραμμικός συνδυασμός των σημείων y_i:

$$x = \sum_{i=1}^{k} \lambda_i y_i \text{ με } \begin{cases} \sum_{i=1}^{k} \lambda_i = 1 \\ \lambda_i \geq 0, \forall i \end{cases}$$

Έχουμε $f(x) = f\left(\sum_{i=1}^{k} \lambda_i y_i\right) = \sum_{i=1}^{k} \lambda_i f(y_i)$ (αφού η f είναι γραμμική συνάρτηση).

Άρα $f(x) \leq \sum_{i=1}^{k} \lambda_i (\max_i f(y_i)) = f(\hat{x}) \sum_{i=1}^{k} \lambda_i = f(\hat{x})$ με $\hat{x} = y_{i^*}$ και $i^* = argmax\{f(y_i)\}$.

Στη συνέχεια θα δούμε μερικά ακόμη παραδείγματα μοντελοποίησης και την επίλυσή τους με γραφική παράσταση. Αυτό βέβαια είναι εφικτό όσο η διάσταση του προβλήματός μας παραμένει μικρή.

3.5 Παραδείγματα

3.5.1 Παράδειγμα παραγωγής

Ένα εργοστάσιο παράγει δύο προϊόντα P_1 και P_2 με τη βοήθεια τριών πρώτων υλών M_1, M_2 και M_3 οι οποίες υπάρχουν σε περιορισμένες ποσότητες, 18 μονάδες για την M_1, 8 μονάδες για την M_2 και 14 μονάδες για την M_3. Τα τεχνικά χαρακτηριστικά της παραγωγής των προϊόντων δίδονται στον παρακάτω πίνακα (πίνακας 3.2), όπου βλέπουμε τις αναλογίες στις ποσότητες της πρώτης ύλης οι οποίες είναι απαραίτητες για την παραγωγή μίας μονάδας ενός προϊόντος $P_i, (i = 1, 2)$.

	M_1	M_2	M_3
P_1	1	1	2
P_2	3	1	1

Πίνακας 3.2: Το προϊόν P_1 χρειάζεται 1 μονάδα από την πρώτη ύλη M_1, 1 από την M_2 και 2 μονάδες από την M_3.

Το κέρδος του εργοστασίου εκφράζεται από ένα μοναδιαίο κέρδος c_1 από το προϊόν P_1 και ένα μοναδιαίο κέρδος c_2 από το προϊόν P_2.

Να δοθεί ένα πρόγραμμα παραγωγής που ικανοποιεί τις διαθέσιμες πρώτες ύλες και να μεγιστοποιεί το κέρδος του εργοστασίου.

Ας υποθέσουμε ότι x_1 μονάδες παραγωγής του προϊόντος P_1 θα χρειαστούν x_1 μονάδες από την πρώτη ύλη M_1, x_1 μονάδες από την M_2 και $2x_1$ μονάδες από την M_3 (γραμμικότητα). Παρομοίως για το προϊόν P_2.

Ένα σχήμα παραγωγής θα προσδιορίζεται από τις μεταβλητές x_1, x_2 οι τιμές των οποίων θα δίδουν τις ποσότητες των προϊόντων P_1 και P_2 που θα παραχθούν. Προφανώς $x_1, x_2 \geq 0$. Το πρόβλημα που πρέπει να επιλυθεί είναι:

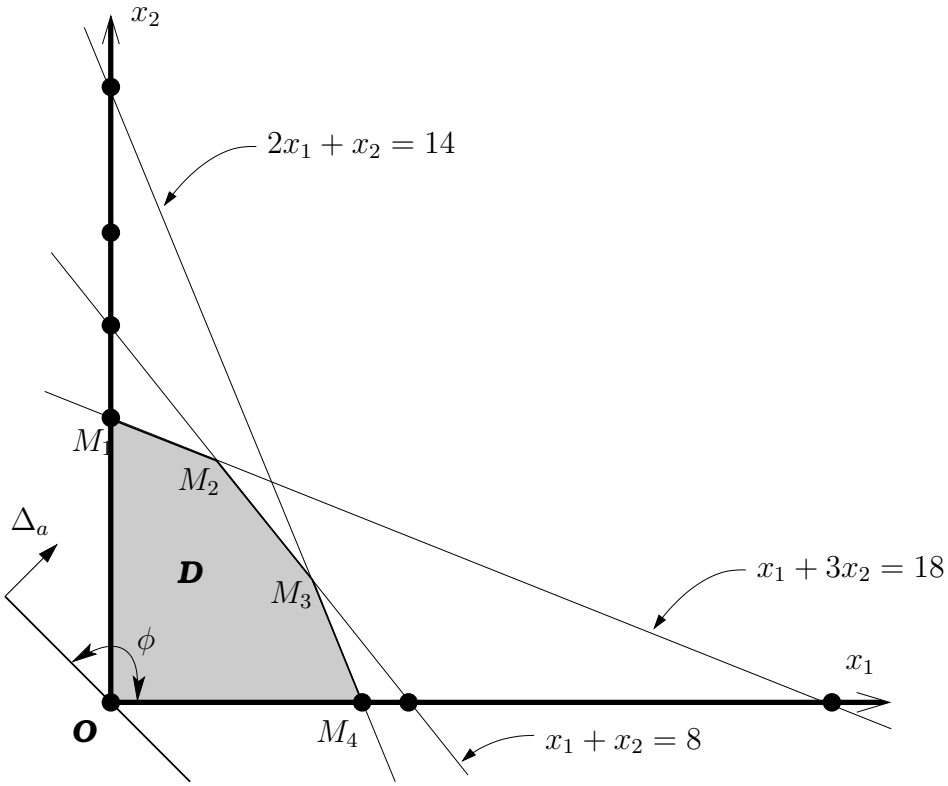
$$\begin{aligned} \max z = c_1x_1 + c_2x_2 \\ x_1 + 3x_2 \leq 18 \\ x_1 + x_2 \leq 8 \\ 2x_1 + x_2 \leq 14 \\ x_1, x_2 \geq 0 \end{aligned}$$

Μπορούμε να γράψουμε το πρόγραμμα αυτό χρησιμοποιώντας τους πίνακες x, c, A και b ως εξής:

$$\begin{aligned} &\max z = c^t x \\ &Ax \leq b \\ &x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, A = \begin{pmatrix} 1 & 3 \\ 1 & 1 \\ 2 & 1 \end{pmatrix}, b = \begin{pmatrix} 18 \\ 8 \\ 14 \end{pmatrix} \\ &c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \end{aligned}$$

Γραφική επίλυση παραδείγματος παραγωγής

α) Ας θεωρήσουμε κατ΄ αρχήν $c_1 = c_2 = 1$, για απλοποίηση της επίλυσης.



Σχήμα 3.4: Η γωνία ϕ έχει εφ(ϕ)=-1 και ο χώρος των εφικτών λύσεων λαμβάνοντας υπ΄ όψιν τους περιορισμούς και την μη αρνητικότητα των μεταβλητών x_1, x_2 είναι ο γραμμοσκιασμένος χώρος D.

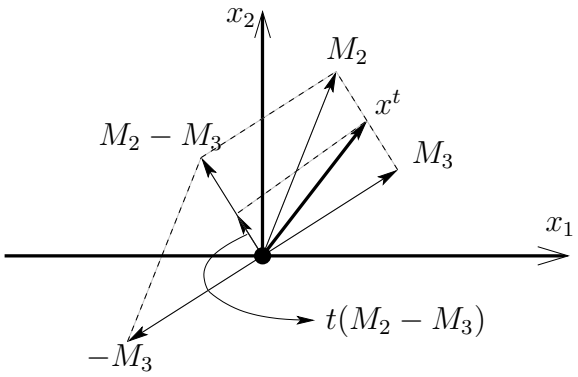
Η ευθεία $\Delta_a : x_1 + x_2 = a$ έχει κλίση ϕ ως προς τον άξονα x_1 με εφ$(\varphi) = -1$. Πράγματι έχουμε $x_2 = a - x_1 \Rightarrow dx_2/dx_1 = -1$ άρα εφ$(\varphi) = -1$. Έστω D το σύνολο των εφικτών λύσεων. Για τα σημεία $\Delta_a \bigcap D$ έχουμε $z = a$, βλέπε σχήμα 3.4.

Αυξάνοντας την τιμή a, η ευθεία Δ_a μετατοπίζεται παράλληλα προς τα δεξιά. Όταν η τιμή a γίνει μέγιστη και $\Delta_a \bigcap D \neq \emptyset$ τότε $z = a$, είναι η βέλτιστη τιμή της αντικειμενικής συνάρτησης.

Έχουμε τότε $a = 8$, με $\Delta_a \bigcap D = [M_2, M_3]$ όπου $M_2 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$ και $M_3 = \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$.

Έχουμε λοιπόν δύο βέλτιστες λύσεις στα σημεία M_2 και M_3 με $c_1x_1 + c_2x_2 = 8$ και $c_1x_1' + c_2x_2' = 8$.

Επίσης κάθε σημείο του τμήματος $[M_2, M_3]$ είναι βέλτιστη λύση. Πράγματι, κάθε σημείο $x^t \in [M_2, M_3]$ γράφεται: $x^t = tM_2 + (1-t)M_3$ με $t \in [0, 1]$. Έστω $x^t = \begin{pmatrix} x_1^t \\ x_2^t \end{pmatrix}$. Ας θεωρήσουμε το διάνυσμα $M_2 - M_3$ που είναι παράλληλο στο τμήμα $[M_2, M_3]$ (βλέπε σχήμα 3.5). Έχουμε $\vec{x^t} = t(M_2 - M_3) + M_3 \Rightarrow \vec{x^t} = tM_2 + M_3 - tM_3 \Rightarrow \vec{x^t} = tM_2 + (1-t)M_3$.



Σχήμα 3.5: Το σημείο x^t πάνω στο τμήμα $[M_2, M_3]$ γράφεται $t(M_2 - M_3) + M_3$.

Η τιμή Z για το σημείο x^t είναι:

$$Z_{x^t} = c_1x_1^t + c_2x_2^t = x_1^t + x_2^t = (3t + (1-t)6) + (5t + 2(1-t)) = 3t + 6 - 6t + 5t + 2 - 2t = 8$$

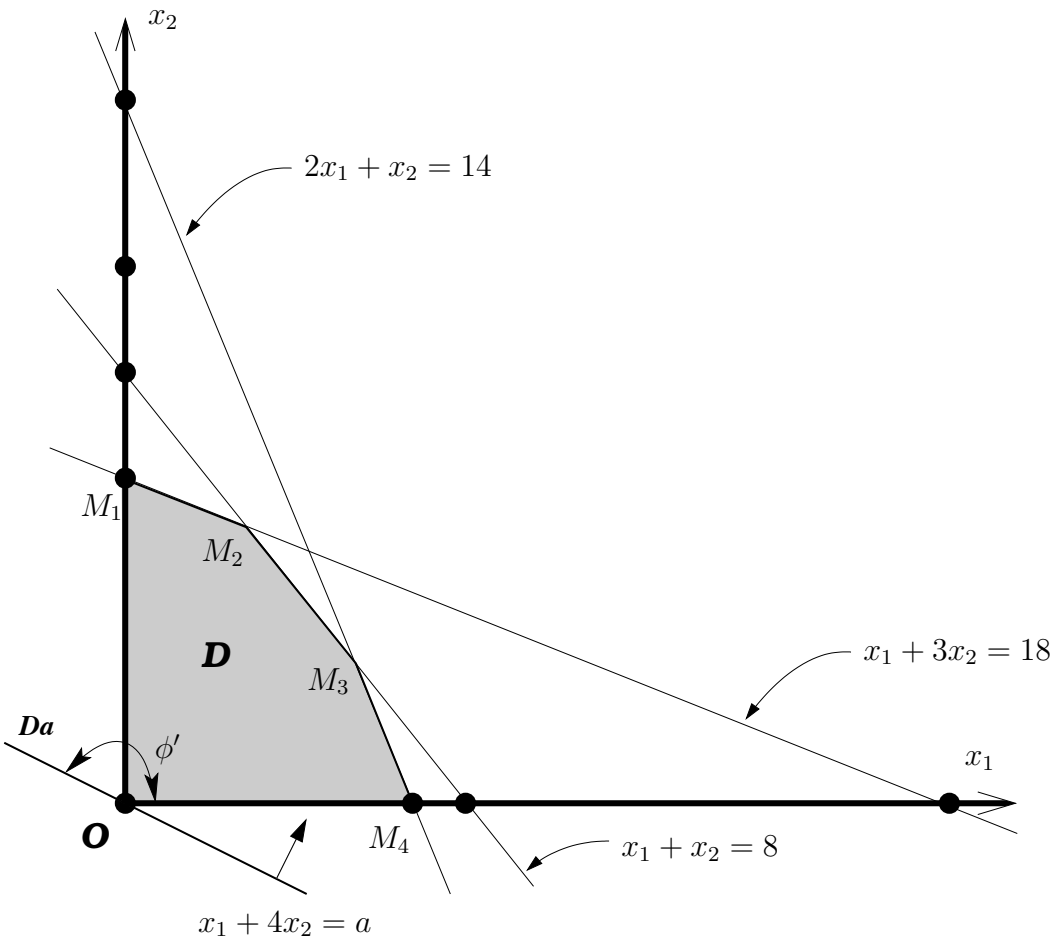
Επομένως έχουμε μία απειρία λύσεων πάνω στο $[M_2, M_3]$ οι οποίες είναι βέλτιστες και έχουν όλες τιμή 8.

β) Ας θεωρήσουμε τώρα $c_1 = 1, c_2 = 4$ (βλέπε σχήμα 3.6).

Έχουμε $\Delta_a' = x_1 + 4x_2 = a$ η οποία έχει κλίση ϕ' με $\varepsilon\varphi(\phi') = \frac{dx_2}{dx_1} = \frac{d(\frac{a-x_1}{4})}{dx_1} = -\frac{1}{4}$

Για όλα τα σημεία της τομής $\Delta_a' \bigcap D$ έχουμε την τιμή της αντικειμενικής συνάρτησης

$z = a$. Το a γίνεται μέγιστο στην τιμή 24 όπου $\Delta'_{24} \bigcap D = \{M_1\}$ με $M_1 = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$ η οποία είναι η μόνη βέλτιστη λύση.



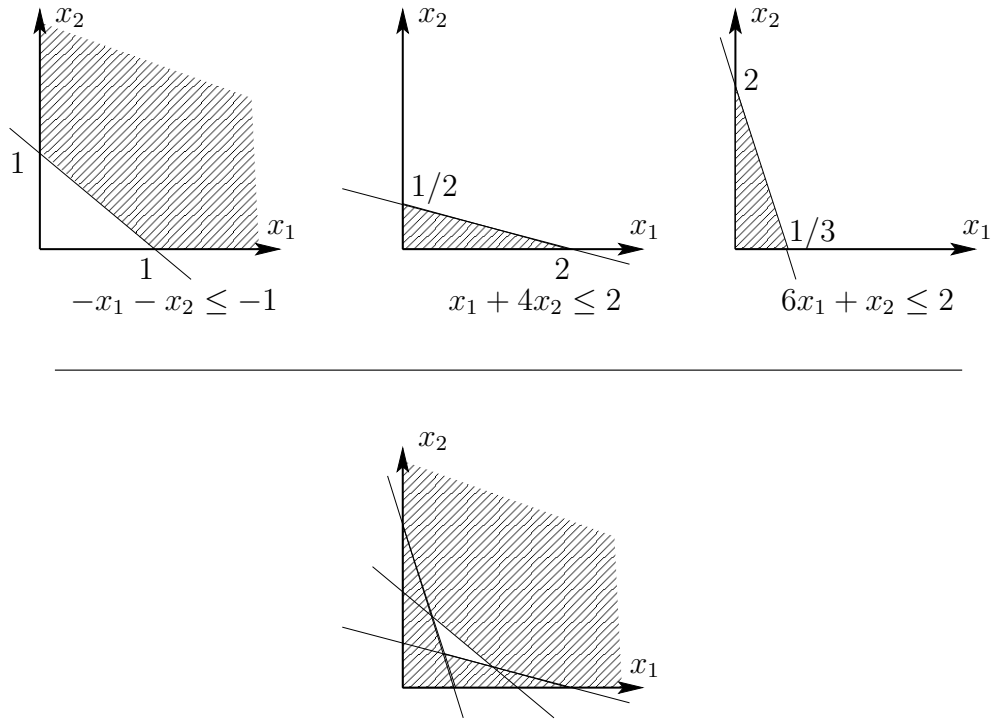
Σχήμα 3.6: Ο χώρος λύσεων παραμένει ο ίδιος D αλλά η κλίση της συνάρτησης είναι ϕ' με $\varepsilon\varphi(\phi')=-\frac{1}{4}$.

3.5.2 Παράδειγμα γραμμικού προγράμματος χωρίς εφικτή λύση

Πολλές φορές ο χώρος των εφικτών λύσεων είναι το κενό σύνολο. Ας δούμε το παρακάτω γραμμικό πρόγραμμα:

$$\begin{aligned}
&\max z = 2x_1 + 3x_2 \\
&-x_1 - x_2 \leq -1 \\
&x_1 + 4x_2 \leq 2 \\
&6x_1 + x_2 \leq 2 \\
&x_i \geq 0
\end{aligned}$$

Ας επιχειρήσουμε να επιλύσουμε το γραμμικό αυτό πρόγραμμα με τη μέθοδο της γραφικής παράστασης (βλέπε σχήμα 3.7).



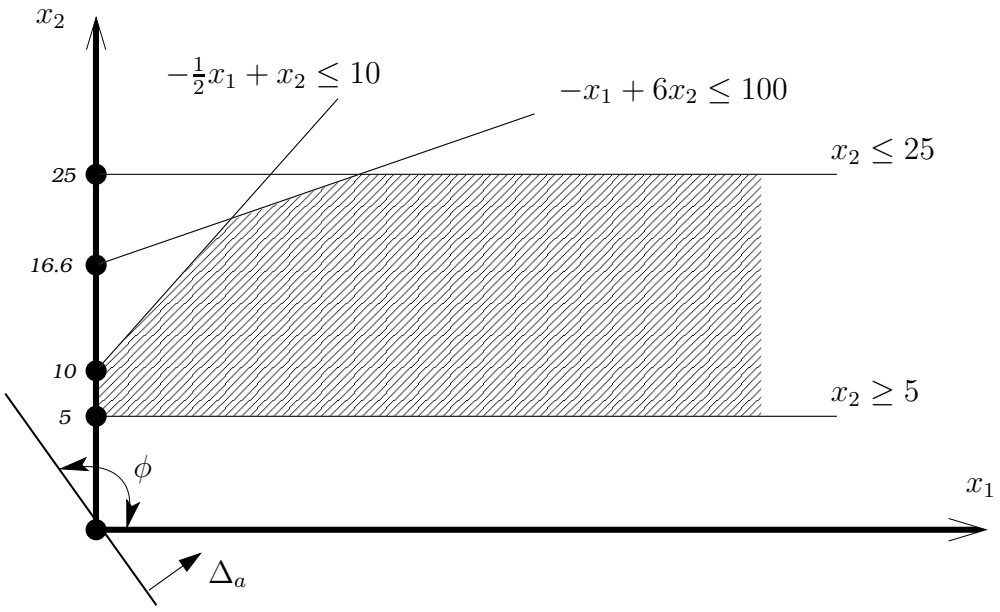
Σχήμα 3.7: Πάνω βλέπουμε τους γραμμοσκιασμένους ημιχώρους που ικανοποιούν τους αντίστοιχους περιορισμούς. Στο κάτω σχήμα παρατηρούμε ότι ο χώρος των εφικτών λύσεων είναι το κενό σύνολο.

Έχουμε ήδη σχολιάσει στο πρώτο παράδειγμα ότι η λύση ενός γραμμικού προγράμματος μπορεί να εκτείνεται στο άπειρο. Αυτό συμβαίνει όταν ο χώρος των εφικτών λύσεων είναι ένα πολύεδρο μη φραγμένο. Ας δούμε το παρακάτω παράδειγμα.

3.5.3 Παράδειγμα γραμμικού προγράμματος με μη φραγμένη αντικειμενική συνάρτηση

Να επιλυθεί γραφικά το γραμμικό πρόγραμμα:

$$\begin{aligned}
&\max z = 2x_1 + 2x_2 \\
&-\tfrac{1}{2}x_1 + x_2 \leq 10 \\
&x_2 \leq 25 \\
&x_2 \geq 5 \\
&-x_1 + 6x_2 \leq 100 \\
&x_1 \geq 0, x_2 \geq 0
\end{aligned}$$



Σχήμα 3.8: Ο γραμμοσκιασμένος χώρος είναι ο χώρος των εφικτών λύσεων και είναι ένα ανοικτό πολύεδρο. Η αντικειμενική συνάρτηση εκτείνεται στο άπειρο.

Το κυρτό σύνολο είναι μη φραγμένο (βλέπε σχ. 3.8). Η αντικειμενική συνάρτηση μπορεί να αυξηθεί και να πάρει τιμές αυθαίρετα μεγάλες.

Στη συνέχεια θα δούμε δύο παραδείγματα ακεραίου γραμμικού προγραμματισμού. Είναι δύο βασικά προβλήματα της θεωρητικής πληροφορικής με πολλές πρακτικές εφαρμογές. Θα επανέλθουμε και αργότερα στα προβλήματα αυτά και τις μεθόδους επίλυσης τους.

3.5.4 Μέγιστο ανεξάρτητο σύνολο σε ένα γράφο (Maximum Independent Set)

Δίδεται ένας γράφος $G = (V, E)$ με V το σύνολο των κόμβων και E το σύνολο των πλευρών. Ένα ανεξάρτητο υποσύνολο $V' \subseteq V$ είναι ένα υποσύνολο των κόμβων οι οποίοι ανά δύο δεν είναι γειτονικοί. Να βρεθεί ένα ανεξάρτητο υποσύνολο με μέγιστο πληθικό αριθμό. Να μοντελοποιηθεί το πρόβλημα σαν ένα πρόβλημα γραμμικού προγραμματισμού.

Μοντελοποίηση

Έστω $|V| = n$ και $|E| = m$.

Θεωρούμε τον πίνακα πρόσπτωσης A του γράφου G, με

$$A = (a_{ij}) = \begin{cases} 1 & \text{αν η πλευρά } e_i \text{ περιέχει τον κόμβο } j \\ 0 & \text{διαφορετικά} \end{cases}$$

Κάθε γραμμή του πίνακα A περιέχει ακριβώς δύο 1.

Έστω $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, με $x_i = \begin{cases} 1 & \text{αν ο κόμβος } i \text{ είναι στη λύση} \\ 0 & \text{διαφορετικά} \end{cases}$

Τότε το πρόβλημα γράφεται:

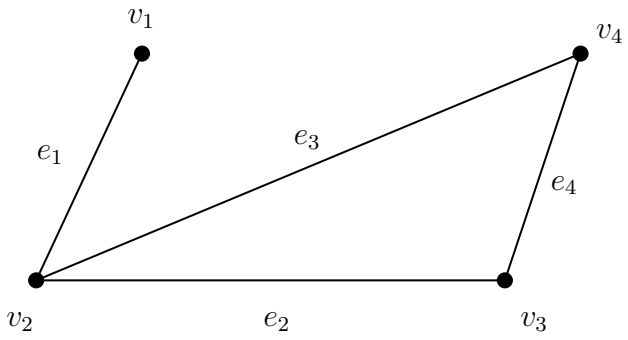
$$\boxed{\begin{array}{rl} Ax & \leq \mathbb{1} \\ \max z & = \mathbb{1}^t x \end{array}}$$

όπου $\mathbb{1}$ είναι το μοναδιαίο διάνυσμα κολόνα $\mathbb{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$ διάστασης m και $\mathbb{1}^t =$

$(1, \ldots, 1)$ το μοναδιαίο διάνυσμα γραμμή διάστασης n.

Παράδειγμα

Έστω ο γράφος $G = (V, E)$ με $V = \{v_1, v_2, v_3, v_4\}$ και $E = \{e_1, e_2, e_3, e_4\}$ (βλ. σχήμα 3.9)



Σχήμα 3.9: Ένας απλός γράφος $G = (V, E)$ και μια αυθαίρετη αρίθμηση των πλευρών του

Ο πίνακας προσπτώσεως A και η μοντελοποίηση του προβλήματος του μέγιστου ανεξαρτήτου υποσυνόλου με την μορφή πινάκων και αναλυτικά έχουν ως εξής:

$$A = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline e_1 & 1 & 1 & 0 & 0 \\ e_2 & 0 & 1 & 1 & 0 \\ e_3 & 0 & 1 & 0 & 1 \\ e_4 & 0 & 0 & 1 & 1 \end{array} \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \qquad \mathbb{1} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\boxed{\begin{array}{c} Ax \leq \mathbb{1} \\ \max \mathbb{1}^t x \\ x_i \in \{0, 1\} \end{array}} \qquad \begin{array}{c} x_1 + x_2 \leq 1 \\ x_2 + x_3 \leq 1 \\ x_2 + x_4 \leq 1 \\ x_3 + x_4 \leq 1 \\ x_i \in \{0, 1\} \\ \hline \max z = x_1 + x_2 + x_3 + x_4 \end{array}$$

Δύο ισοδύναμες βέλτιστες λύσεις του προβλήματος για το στιγμιότυπο του σχήματος 3.9 είναι οι $\{v_1, v_3\}$ και $\{v_1, v_4\}$.

Ας δούμε τώρα ένα άλλο παράδειγμα ακεραίου γραμμικού προγραμματισμού το οποίο είναι συμπληρωματικό του προβλήματος του ανεξαρτήτου υποσυνόλου. Η συμπληρωματικότητα έγκειται στο γεγονός ότι το συμπλήρωμα της βέλτιστης λύσης του ενός, είναι η βέλτιστη λύση του άλλου.

3.5.5 Πρόβλημα ελάχιστης κομβικής επικάλυψης (Vertex Covering)

Δίδεται ένας γράφος $G = (V, E)$ με V το σύνολο των κόμβων και E το σύνολο των πλευρών. Μία κομβική επικάλυψη $V' \subseteq V$ είναι ένα υποσύνολο κόμβων τέτοιο ώστε κάθε πλευρά του γράφου έχει το λιγότερο ένα άκρο μέσα στο V'. Να βρεθεί μία κομβική επικάλυψη με ελάχιστο πληθικό αριθμό. Να μοντελοποιηθεί το πρόβλημα σαν ένα πρόβλημα γραμμικού προγραμματισμού.

Θεωρούμε πάλι τον πίνακα πρόσπτωσης A, με $A = (a_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}}$ όπου $m = |E|$ και $n = |V|$.

Τότε το πρόβλημα υπό μορφή πινάκων γράφεται:

$$\boxed{\begin{array}{c} Ax \geq \mathbb{1} \\ \min \mathbb{1}^t x \\ x_i \in \{0, 1\} \end{array}}$$

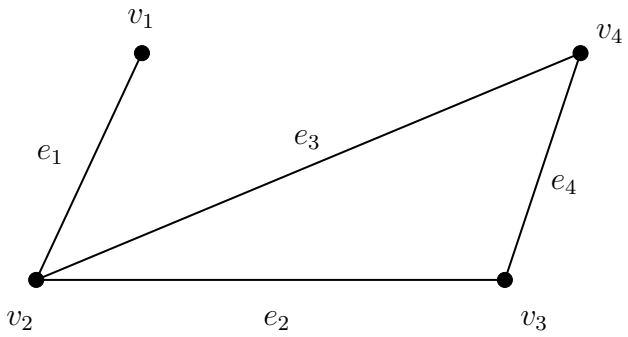
όπου $\mathbb{1} = (1, 1, \ldots, 1)^t$ διάστασης m και $\mathbb{1}^t = (1, 1, \ldots, 1)$ διάστασης n.

Παράδειγμα

$$\begin{aligned} x_1 + x_2 &\geq 1 \\ x_2 + x_3 &\geq 1 \\ x_2 + x_4 &\geq 1 \\ x_3 + x_4 &\geq 1 \end{aligned}$$

$$\min z = x_1 + x_2 + x_3 + x_4$$

και υπό μορφή πινάκων:



Σχήμα 3.10: Δύο ισοδύναμες βέλτιστες λύσεις του προβλήματος της κομβικής επικάλυψης είναι $\{v_2, v_3\}$ και $\{v_2, v_4\}$.

$$\boxed{\begin{array}{c} Ax \geq \mathbb{1} \\ \min \mathbb{1}^t x \\ x_i \in \{0,1\} \end{array}}$$

όπου:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad \mathbb{1} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Παρατηρούμε ότι αν θεωρήσουμε μια βέλτιστη λύση του προβλήματος της κομβικής επικάλυψης π.χ. την $\{v_2, v_3\}$ τότε το συμπλήρωμά της $\{v_1, v_2, v_3, v_4\} - \{v_2, v_3\} = \{v_1, v_4\}$ είναι μια βέλτιστη λύση του προβλήματος μέγιστου ανεξαρτήτου (υπο)συνόλου στο γράφο G. Οι βέλτιστες λύσεις και των δύο προβλημάτων στο στιγμιότυπο των σχημάτων 3.9 και 3.10 επισημαίνονται εύκολα εποπτικά. Μπορούμε όμως να χρησιμοποιήσουμε και την γραφική μέθοδο, αλλά θέλει προσοχή στο ότι η βέλτιστη λύση δεν βρίσκεται πάντα σε κορυφή του πολυέδρου όπως στον συνεχή γραμμικό προγραμματισμό. Αυτό δημιουργεί και τη δυσκολία των προβλημάτων του ακέραιου γραμμικού προγραμματισμού, αλλά θα ασχοληθούμε αργότερα με αυτά τα προβλήματα. Προς το παρόν θα επικεντρωθούμε στο συνεχή γραμμικό προγραμματισμό και ένα βασικό αλγόριθμο επίλυσής του, ο οποίος ας σημειώσουμε από τώρα δεν είναι αποδοτικός αλγόριθμος

(είναι εκθετικός στην χείριστη περίπτωση) αλλά στην πράξη συμπεριφέρεται πάρα πολύ καλά.

3.6 Γενική Μορφή Γραμμικού Προγράμματος

Έστω $\{a_{ij}\}, \{b_j\}, \{c_j\}$ δεδομένοι πραγματικοί αριθμοί με $1 \leq i \leq m$ και $1 \leq j \leq n$.

Αναζητούμε να προσδιορίσουμε τους n πραγματικούς αριθμούς $x_1, \ldots, x_n$ που ικανοποιούν τους $n + m$ περιορισμούς (m ανισώσεις ή εξισώσεις και n περιορισμοί μη αρνητικότητας για τις μεταβλητές):

$$\begin{array}{c} a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1 \\ \vdots \\ a_{m1}x_1 + \ldots + a_{mn}x_n \leq b_m x_1 \geq 0, i = 1, \ldots, n \end{array}$$

ή

$$\begin{array}{c} a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ \vdots \\ a_{m1}x_1 + \ldots + a_{mn}x_n = b_m x_1 \geq 0, i = 1, \ldots, n \end{array}$$

μεγιστοποιώντας τη συνάρτηση $z = c_1x_1 + \ldots + c_nx_n$. μακε Η συνάρτηση ονομάζεται αντικειμενική συνάρτηση. Οι άγνωστοι $x_i, i = 1, \ldots, n$ ονομάζονται μεταβλητές απόφασης. Έστω A ο πίνακας (a_{ij}) με m γραμμές και n κολόνες, b το διάνυσμα κολόνα $(b_i)_{1 \leq i \leq m}$, c το διάνυσμα γραμμή $(c_j)_{1 \leq j \leq n}$ και x το άγνωστο διάνυσμα κολόνα $(x_j)_{1 \leq j \leq n}$. Το πρόβλημα γράφεται τότε υπό μορφή πινάκων:

$$\begin{array}{c} \max z = cx \\ x \geq 0 \\ Ax \leq b \text{ ή } Ax = b \end{array}$$

Βέβαια, διάφορες παραλλαγές του γραμμικού προγράμματος μπορούν να αναχθούν στο ίδιο πρόβλημα, παραδείγματος χάρη:

1. Οι δύο τύποι των περιορισμών $Ax \leq b$ ή $Ax = b$ μπορούν να μετατραπούν ο ένας στον άλλο. Πράγματι, η ισότητα $Ax = b$ είναι ισοδύναμη με ένα διπλό σύστημα ανισοτήτων $Ax \leq b$ και $Ax \geq b$, δηλαδή $Ax \leq b$ και $-Ax \leq -b$.

   Αντίστροφα, εισάγοντας m επιπλέον μεταβλητές $x_{n+i}, 1 \leq i \leq m$, ονομαζόμενες *μεταβλητές απόκλισης*, κάθε ανισότητα $a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$ γράφεται:

   $$a_{i1} + \ldots + a_{in}x_n + x_{n+1} = b_i \text{ με } x_{n+1} \geq 0$$

   Αν το σύστημα των περιορισμών περιέχει ταυτόχρονα ανισότητες και ισότητες, εισάγουμε μία μεταβλητή απόκλισης ανά ανισότητα μόνο.

   Κατά την επίλυση του προβλήματος όπως θα δούμε παρακάτω, οι μεταβλητές απόφασης και απόκλισης παίζουν ακριβώς τον ίδιο ρόλο.

2. Αν ο περιορισμός $x_i \geq 0$ είναι της μορφής $x_i \geq a$ (αντίστοιχα $x_i \leq a$), αναγόμαστε στο αρχικό πρόβλημα αντικαθιστώντας τη μεταβλητή x_i από $x'_i = x_1 - a$ (αντίστοιχα $x'_i = a - x_i$) η οποία ικανοποιεί $x'_i \geq 0$.

3. Το πρόβλημα ελαχιστοποίησης της $z = cx$ ανάγεται στη μεγιστοποίηση της $z = -cx$ κάτω από τους ίδιους περιορισμούς.

Με το ενδεχόμενο εισαγωγής μεταβλητών απόκλισης, το πρόβλημά μας για τη συνέχεια θα είναι:

$$\boxed{\begin{array}{c} \max z = cx \\ Ax = b \\ x \geq 0 \end{array}}$$

Θα συμβολίζουμε στη συνέχεια με $p, n \leq p \leq n + m$, το συνολικό αριθμό μεταβλητών (εκ των οποίων οι n είναι μεταβλητές απόφασης και οι $p - n$ είναι μεταβλητές απόκλισης).

Θα λέμε ότι το διάνυσμα $(x_1, \ldots, x_p)$ είναι μία εφικτή λύση του προβλήματος αν οι μεταβλητές x_i ικανοποιούν όλους τους περιορισμούς. Ένα πρόβλημα που δεν επιδέχεται εφικτή λύση είναι προφανώς αδύνατο.

Όταν το πρόβλημα επιδέχεται μία εφικτή λύση, το σύστημα $Ax = b$ έχει λύσεις. Σε αυτή την περίπτωση, αν η τάξη (rank) του πίνακα A είναι μικρότερη από m, μερικές ισότητες του συστήματος είναι γραμμικοί συνδυασμοί άλλων και μπορούν να αγνοηθούν. Υποθέτουμε λοιπόν ότι τάξη $(A) = m$ το οποίο συνεπάγεται ότι $m \leq p$.

Μία **βάση** είναι ένα σύνολο από m μεταβλητές του συστήματος των οποίων οι αντίστοιχες κολόνες στον πίνακα A είναι γραμμικώς ανεξάρτητες. Οι άλλες μεταβλητές θα ονομάζονται μεταβλητές **εκτός βάσης**. Θα συμβολίζουμε στη συνέχεια τις μεταβλητές εντός βάσης και εκτός βάσης αντίστοιχα ως εξής:

$$\text{B} = \{x_{i_1}, \ldots, x_{i_m}\} \text{ και } \text{EB} = \{x_{i_{m+1}}, \ldots, x_{i_p}\}$$

Το σύστημα όπως θα δούμε στην συνέχεια μπορεί να γραφεί υπό τη μορφή:

$$\boxed{\begin{array}{c} x_{i_1} = b'_1 + a'_{11}x_{i_{m+1}} + \ldots + a'_{1p-m}x_{i_p} \\ \vdots \\ x_{i_m} = b'_m + a'_{m1}x_{i_{m+1}} + \ldots + a'_{mp-m}x_{i_p} \end{array}}$$

όπου b'_i και a'_{ij} πραγματικοί αριθμοί.

Ονομάζουμε **βασική λύση** σχετικά με τη θεωρούμενη βάση, τη μοναδική λύση του συστήματος $Ax = b$ για την οποία οι μεταβλητές εκτός βάσης είναι μηδενικές, δηλαδή:

$$\begin{array}{l} x^*_{i_k} = b'_k \quad \text{για} \quad 1 \leq k \leq m \quad \text{και} \\ x^*_{i_k} = 0 \quad \text{για} \quad m+1 \leq k \leq p \end{array}$$

Μία τέτοια βασική λύση είναι **εφικτή** αν ικανοποιεί τους περιορισμούς μη αρνητικότητας $b'_k \geq 0$, για $1 \leq k \leq m$.

Η λύση του προβλήματος γίνεται σε δύο φάσεις. Αρχικά εξετάζουμε αν το πρόβλημα επιδέχεται μία εφικτή λύση, και σε αυτή την περίπτωση καθορίζουμε μία **βασική εφικτή λύση**.

Στη συνέχεια, ξεκινώντας από αυτή την βασική εφικτή λύση, αναζητούμε το μέγιστο της αντικειμενικής συνάρτησης με τη βοήθεια της μεθόδου Simplex (θα δούμε στη

συνέχεια ότι είτε αυτό το μέγιστο υπάρχει, είτε η αντικειμενική συνάρτηση δεν είναι φραγμένη).

Θα δούμε τώρα τη μέθοδο Simplex και αμέσως μετά θα περιγράψουμε τον τρόπο με τον οποίο μπορούμε να βρούμε, αν βέβαια υπάρχει, μία βασική εφικτή λύση. Αυτό, γιατί όπως θα δούμε, θα κάνουμε πάλι χρήση της Simplex, για να βρούμε μία βασική εφικτή λύση.

3.7 Αλγόριθμος Simplex

Η Simplex είναι η πιο διαδεδομένη μέθοδος επίλυσης προβλημάτων γραμμικού προγραμματισμού. Είναι μια επαναληπτική μέθοδος για την επίλυση ενός γραμμικού προβλήματος της μορφής:

$$\text{Min} \quad c^T x$$

με τους περιορισμούς:

$$Ax = b$$
$$x \geq 0$$

Η μέθοδος συνίσταται στο να βελτιώσει προοδευτικά μία δεδομένη βασική εφικτή λύση[1], αυξάνοντας την τιμή της αντικειμενικής συνάρτησης μέχρι να βρεθεί η βέλτιστη τιμή ή να συμπεράνουμε ότι η τιμή της αντικειμενικής συνάρτησης δεν είναι φραγμένη.

Το σύνολο των λύσεων του παραπάνω προβλήματος σχηματίζει ένα πολύτοπο P. Οι εφικτές λύσεις του προβλήματος (οι λύσεις που δεν παραβιάζουν τους περιορισμούς) βρίσκονται είτε στα εσωτερικά σημεία του πολυτόπου είτε στις κορυφές του πολυτόπου. Μία τουλάχιστον βέλτιστη λύση βρίσκεται στις κορυφές του πολυτόπου. Η μέθοδος μετακινείται από μια βασική εφικτή λύση σε κάποια άλλη, δηλαδή κινείται στα ακρότατα σημεία του πολυτόπου P. Σε κάθε επανάληψη εξετάζει αν η τρέχουσα βασική λύση

[1] Ένα σημείο x είναι βασική λύση αν το x ικανοποιεί τους περιορισμούς ισότητας του γραμμικού προγράμματος και αν οι στήλες του πίνακα περιορισμών A που αντιστοιχούν στα μη μηδενικά στοιχεία του x είναι γραμμικά ανεξάρτητες. Το x είναι βασική εφικτή λύση αν ικανοποιεί και τον περιορισμό $x \geq 0$.

είναι βέλτιστη. Αν δεν είναι βέλτιστη, επιλέγεται μια κατεύθυνση μετακίνησης, ώστε η συνάρτηση κόστους να βελτιώνεται και μετακινείται σε μια γειτονική βασική εφικτή λύση. Στη συνέχεια η διαδικασία επαναλαμβάνεται.

Για να αποφύγουμε ένα βαρύ φορμαλισμό, θα εξηγήσουμε τη μέθοδο σε ένα παράδειγμα.

Περιγραφή της μεθόδου σε ένα παράδειγμα

Ας θεωρήσουμε το παράδειγμα της εταιρίας και των συμβολαίων, εισάγοντας τρεις μεταβλητές απόκλισης x_3, x_4, x_5. Έχουμε ότι $\max z = 4x_1 + 3x_2$. Υπάρχει βέβαια και η πολλαπλασιαστική σταθερά 2, που θα τη λάβουμε υπ' όψιν στο τέλος για την τελική τιμή της z. Το πρόβλημα γράφεται:

$$\begin{aligned}
&\max z = 4x_1 + 3x_2 \\
&5x_1 + 3x_2 + x_3 \quad\quad = 30 \\
&2x_1 + 3x_2 \quad + x_4 \quad\quad = 24 \\
&\ x_1 + 3x_2 \quad\quad + x_5 \quad = 18 \\
&x_i \geq 0, i = 1, 2, 3, 4, 5
\end{aligned}$$

Πρώτο βήμα

Μπορούμε να πάρουμε ως βάση το σύνολο των μεταβλητών B=$\{x_3, x_4, x_5\}$ και επομένως έχουμε EB =$\{x_1, x_2\}$. Σε αυτή τη βάση αντιστοιχεί η βασική λύση:

$$\{x_1 = 0,\ x_2 = 0,\ x_3 = 30,\ x_4 = 24,\ x_5 = 18\}$$

όπου θέσαμε μηδενικές τιμές στις μεταβλητές εκτός βάσης. Αυτή η βασική λύση είναι εφικτή αφού ικανοποιεί όλους τους περιορισμούς $x_i \geq 0, \quad \forall \quad i \leq i \leq 5$. Παρατηρούμε ότι δίνει στην αντικειμενική συνάρτηση z την τιμή 0.

Το παρακάτω σύστημα που δίνει τις μεταβλητές της βάσης και την αντικειμενική συνάρτηση σε σχέση με τις μεταβλητές εκτός βάσης θα ονομάζεται το **λεξικό** συνδεδεμένο με τη τρέχουσα βάση.

$$
\begin{aligned}
x_3 &= 30 - 5x_1 - 3x_2 \\
x_4 &= 24 - 2x_1 - 3x_2 \\
x_5 &= 18 - x_1 - 3x_2 \\
z &= 4x_1 + 3x_2 \\
&\text{με } x_i \geq 0, \forall i = 1, \ldots, 5
\end{aligned}
$$

Το λεξικό θα καλείται εφικτό αν η αντίστοιχη βασική λύση είναι εφικτή.

Αν δώσουμε τώρα στο x_2 μία (αυστηρά) θετική τιμή διατηρώντας ταυτόχρονα στο x_1 μηδενική τιμή, η αντικειμενική συνάρτηση αυξάνει διότι ο συντελεστής της x_2 μέσα στη z είναι θετικός. Η μεγαλύτερη δυνατή τιμή για τη x_2, με $x_1 = 0$, λαμβάνοντας υπ' όψιν τους περιορισμούς $x_3 \geq 0, \quad x_4 \geq 0, \quad x_5 \geq 0$, είναι 6, τιμή που επιβάλλεται από τον περιορισμό $x_5 \geq 0$. Επιπλέον έχουμε $x_2 = 6$ όταν $x_5 = 0$.

Ας σημειώσουμε ότι οι μοναδικοί περιορισμοί που μπορούν να περιορίσουν την τιμή του x_2 είναι αυτοί για τους οποίους ο συντελεστής του x_2 στο δεύτερο μέλος είναι αρνητικός.

Δεύτερο βήμα

Αντικαθιστούμε την μεταβλητή x_5 η οποία είναι μέσα στη βάση με την x_2 η οποία είναι εκτός βάσης. Λέμε ότι εισάγουμε την x_2 στη βάση και βγάζουμε την x_5 εκτός βάσης. Έχουμε τώρα τη νέα βάση $\mathrm{B} = \{x_2, x_3, x_4\}$ και τις μεταβλητές εκτός βάσης $\mathrm{EB} = \{x_1, x_5\}$. Το νέο λεξικό εκφράζει τις μεταβλητές x_2, x_3, x_4 και την αντικειμενική συνάρτηση z σε σχέση με τις μεταβλητές x_1 και x_5.

$$
\left.
\begin{aligned}
3x_2 &= 18 - x_1 - x_5 \\
x_3 &= 30 - 5x_1 - (18 - x_1 - x_5) \\
x_4 &= 24 - 2x_1 - (18 - x_1 - x_5) \\
z &= 4x_1 + (18 - x_1 - x_5)
\end{aligned}
\right\| \Rightarrow
\begin{aligned}
x_2 &= 6 - \tfrac{x_1}{3} - \tfrac{x_5}{3} \\
x_3 &= 12 - 4x_1 + x_5 \\
x_4 &= 6 - x_1 + x_5 \\
z &= 18 + 3x_1 - x_5
\end{aligned}
$$

Η αντίστοιχη βασική λύση $(x_1, x_2, x_3, x_4, x_5) = (0, 6, 12, 6, 0)$ είναι εφικτή αφού αυτό ακριβώς καθόρισε την επιλογή της x_5 για να βγει από τη βάση. Η τιμή z γίνεται τώρα 18.

Αν διατηρήσουμε για την μεταβλητή x_5 την τιμή 0 και δώσουμε στην x_1 μία θετική τιμή, η τιμή της αντικειμενικής συνάρτησης z αυξάνει, αφού ο συντελεστής του x_1 είναι

θετικός.

Για να μείνουν οι μεταβλητές x_2, x_3 και x_4 μη αρνητικές (≥ 0), η μεταβλητή x_1 θα πρέπει να είναι το πολύ 3(καθορίζεται από την δεύτερη εξίσωση). Θα έχουμε $x_1 = 3$ όταν $x_3 = 0$.

Τρίτο βήμα

Εισάγουμε τη μεταβλητή x_1 στη βάση και βγάζουμε τη μεταβλητή x_3 εκτός βάσης. Έχουμε τη νέα βάση: $\mathrm{B} = \{x_1, x_2, x_4\}$ και τις μεταβλητές εκτός βάσης $\mathrm{EB} = \{x_3, x_5\}$.

Το νέο λεξικό εκφράζει τις μεταβλητές x_1, x_2, x_4 σε σχέση με τις μεταβλητές x_3 και x_5.

$$\begin{array}{l} 4x_1 = 12 - x_3 + x_5 \\ x_2 = 6 - \frac{12 - x_3 + x_5}{12} - \frac{x_5}{3} \\ x_4 = 6 - \frac{12 - x_3 + x_5}{4} + x_5 \\ z = 18 + \frac{3}{4}(12 - x_3 + x_5) - x_5 \end{array} \Rightarrow \begin{array}{l} x_1 = 3 - \frac{x_3}{4} - \frac{x_5}{4} \\ x_2 = 5 + \frac{x_3}{12} - \frac{5}{12}x_5 \\ x_4 = 3 + \frac{x_3}{4} - \frac{3}{4}x_5 \\ z = 27 - \frac{3}{4}x_3 - \frac{1}{4}x_5 \end{array}$$

Αφού τώρα οι συντελεστές των x_3 και x_5 μέσα στη z είναι αρνητικοί και $x_3 \geq 0, x_5 \geq 0$, η συνάρτηση z δεν μπορεί πλέον να αυξηθεί, αφού οποιαδήποτε θετική τιμή στις μεταβλητές x_3 ή x_5 θα μείωνε την τιμή της. Το μέγιστο της z είναι 27 και επιτυγχάνεται για $x_1 = 3,\ x_2 = 5,\ x_3 = 0,\ x_4 = 3,\ x_5 = 0$.

Το αρχικό πρόβλημα έχει λοιπόν επιλυθεί.

Η επιχείρηση έχει συμφέρον να κλείσει 3 συμβόλαια του πρώτου τύπου και 5 συμβόλαια του δεύτερου τύπου. Θα εκμεταλλευτεί πλήρως τους 30 μηχανικούς και όλο το διαθέσιμο χρόνο μηχανής, ενώ τρεις μηχανικοί θα παραμείνουν διαθέσιμοι και θα κερδίζει $z^* \times 2000 = 54000$ EURO, αν η νομισματική μονάδα στην παρουσίαση του προβλήματος είναι 1000 EURO.

Παρατήρηση 1

Θα μπορούσαμε, στο δεύτερο βήμα, να εισαγάγουμε στη βάση την μεταβλητή x_1 αντί για τη μεταβλητή x_2, και να βγάλουμε τη μεταβλητή x_3, το οποίο θα έδινε το λεξικό:

$$x_1 = 6 - \frac{3}{5}x_2 - \frac{x_3}{5}$$
$$x_4 = 12 + \frac{9}{5}x_2 - \frac{2}{5}x_3$$
$$x_5 = 12 + \frac{12}{5}x_2 - \frac{x_3}{5}$$
$$z = 24 + \frac{3}{5}x_2 - \frac{4}{5}x_3$$

Στη συνέχεια θα εισάγαγαμε την x_2 στη βάση και θα βγάζαμε την x_5, οπότε θα βρίσκαμε το ίδιο λεξικό όπως προηγουμένως στο τρίτο βήμα.

Παρατήρηση 2

Για την ορθότητα του προβλήματος, οι μεταβλητές x_1 και x_2 οι οποίες αντιπροσωπεύουν έναν αριθμό συμβολαίων, θα πρέπει εξ΄ αρχής να θεωρηθούν ακέραιες. Εδώ, η βέλτιστη λύση που βρήκαμε δίνει πράγματι ακέραιες τιμές στις μεταβλητές παρ΄ ότι τις θεωρήσαμε πραγματικές. Αλλά αυτό είναι μία ευχάριστη ΣΥΜΠΤΩΣΗ! Η μέθοδος Simplex επιτρέπει να πάρουμε μία βέλτιστη λύση σε πραγματικούς και όχι αναγκαία σε ακέραιους αριθμούς. Το πρόβλημα βελτιστοποίησης με ακέραιους αριθμούς είναι πολύ πιο δύσκολο και θα το μελετήσουμε αργότερα.

3.7.1 Χαρακτηριστικά της μεθόδου Simplex και δικαιολόγηση

Στο τέλος κάθε βήματος, έχουμε ένα εφικτό λεξικό το οποίο γράφεται, αν συμβολίσουμε $t_1, \ldots, t_m$ τις βασικές μεταβλητές, και $u_1, \ldots, u_{p-m}$ τις μεταβλητές εκτός βάσης.

$$\boxed{\begin{aligned} t_i &= k_i + \sum_{j=1}^{p-m} \mu_{ij} u_j \quad \text{για } 1 \le i \le m \\ Z &= z_0 + \sum_{j=1}^{p-m} \lambda_j u_j \end{aligned}}$$

με

$$\boxed{\begin{aligned} k_i &\ge 0 \quad \text{για } 1 \le i \le m \\ z_0 &\ge 0 \end{aligned}}$$

Τρεις περιπτώσεις μπορούν να παρουσιαστούν:

1. $\lambda_j \leq 0$, για κάθε $j \in \{1, \ldots, p-m\}$: *Κριτήριο βελτιστότητας*

   Σε αυτή την περίπτωση z_0 είναι η μέγιστη τιμή της z. Αυτό το μέγιστο έχει επιτευχθεί όταν οι μεταβλητές έχουν τιμές
   $u_j^* = 0$, για $1 \leq j \leq p-m$ και
   $t_i^* = k_i$, για $1 \leq i \leq m$.

   Το πρόβλημα έχει επιλυθεί.

2. Υπάρχει ένας δείκτης $j \in \{1, \ldots, p-m\}$ τέτοιος ώστε $\lambda_j > 0$, και γι' αυτό το δείκτη $j, \mu_{ij} \geq 0, \forall i \in \{1, \ldots, m\}$. Αυτό σημαίνει ότι οι περιορισμοί της μη αρνητικότητας ικανοποιούνται για οποιαδήποτε τιμή της μεταβλητής u_j. Όταν η u_j τείνει στο άπειρο, μπορούμε να βρούμε μία εφικτή λύση για την οποία η συνάρτηση z παίρνει τιμές αυθαίρετα μεγάλες. *Η αντικειμενική συνάρτηση δεν είναι φραγμένη.*

3. Υπάρχει ένας δείκτης j και ένας δείκτης i για τους οποίους $\lambda_j > 0$ και $\mu_{ij} < 0$. Συνεχίζουμε να εφαρμόζουμε τη μέθοδο εισάγοντας τη μεταβλητή u_j στη βάση και εξάγοντας από τη βάση τη μεταβλητή t_i με το δείκτη i επιλεγμένο ανάμεσα σε αυτούς τους δείκτες για τους οποίους $\mu_{ij} < 0$ με τον εξής τρόπο:

   - Η αύξηση $-\frac{k_i}{\mu_{ij}}$ που επιτρέπεται για το u_i από τον περιορισμό $t_i \geq 0$ θα πρέπει να είναι ελάχιστη, για να μην παραβιάζεται κανένας από τους περιορισμούς i με $\mu_{ij} < 0$.
   - Όταν περνάμε από την τρέχουσα βασική λύση στη νέα, η τιμή της z αυξάνει κατά $-\lambda_j \frac{k_i}{\mu_{ij}}$, το οποίο είναι θετικό αφού $\mu_{ij} < 0$.

   Η μέθοδος τερματίζει όταν φτάσουμε σε μία από τις δύο πρώτες περιπτώσεις.

Αν η z αυξάνει αυστηρά σε κάθε βήμα, δεν ευρίσκουμε ποτέ την ίδια βάση αφού θα έδινε την ίδια τιμή στη z. Δεδομένου ότι ο αριθμός των δυνατών βάσεων είναι το πολύ ίσος με c_p^m και επομένως πεπερασμένος, είμαστε βέβαιοι ότι θα φτάσουμε μετά από έναν πεπερασμένο αριθμό βημάτων στις περιπτώσεις 1 ή 2.

Αλλά αν σε κάποιο βήμα καταλήξουμε στην κατάσταση όπου μερικοί συντελεστές k_i είναι μηδενικοί, δηλαδή αν μέσα στη βασική λύση μερικές βασικές μεταβλητές είναι

μηδενικές (το λεξικό τότε θα λέμε ότι είναι εκφυλισμένο) τότε η z μπορεί να μην αυξάνει σε αυτό το βήμα. Υπάρχει σε αυτή την περίπτωση, ρίσκο να ξαναβρούμε μία ήδη θεωρηθείσα βάση και να οδηγηθούμε έτσι σε ένα φαινόμενο κυκλισμού στο οποίο ο αλγόριθμος θα εκτελείται έπ΄ αόριστον. Αυτό το φαινόμενο είναι σπάνιο στην πράξη αλλά χάρη στο ακόλουθο θεώρημα, το οποίο θα αποδεχθούμε χωρίς απόδειξη, το αρνητικό φαινόμενο της κυκλικότητας μπορεί πάντα να αποφευχθεί.

Θεώρημα (Bland): Αν, σε περίπτωση εκφυλισμού, διαλέγουμε πάντα για εισερχόμενες και εξερχόμενες μεταβλητές ανάμεσα στις υποψήφιες, αυτές με τον πιο μικρό δυνατό δείκτη, ο κυκλισμός είναι αδύνατος.

Με αυτό τον κανόνα, η μέθοδος Simplex ξεκινώντας από μία βασική εφικτή λύση καταλήγει πάντα, σε έναν πεπερασμένο αριθμό βημάτων, είτε στο συμπέρασμα ότι η προς μεγιστοποίηση αντικειμενική συνάρτηση δεν είναι φραγμένη είτε στην εύρεση της μέγιστης τιμής της αντικειμενικής συνάρτησης καθώς και των τιμών των μεταβλητών για τις οποίες αυτό το μέγιστο έχει επιτευχθεί.

Μέσα στη βασική λύση, είπαμε ότι οι μεταβλητές εκτός βάσης είναι μηδενικές. Αυτή η λύση αντιστοιχεί σε μία κορυφή του πολυέδρου. Λέγοντας ότι διαθέτουμε στην εκκίνηση μία βασική εφικτή λύση σημαίνει ότι γνωρίζουμε μία τέτοια κορυφή. Όταν εισάγουμε στη βάση μία μεταβλητή εκτός βάσης, μετατοπιζόμαστε από την τρέχουσα κορυφή σε μία γειτονική της. Η πλευρά που αντιστοιχεί σε αυτή την μετατόπιση είναι τέτοια ώστε η $\sum_{j=1}^{n} c_j x_j$ παίρνει τιμή σε αυτή την κορυφή μεγαλύτερη απ΄ ότι στην προηγούμενη κορυφή. Η μέθοδος Simplex είναι ένας ''περίπατος'' πάνω στις πλευρές του πολυέδρου, από μία κορυφή στην άλλη, ο οποίος βελτιώνει την αντικειμενική συνάρτηση. Όταν το πολύεδρο είναι φραγμένο (δηλαδή ένα πολύτοπο) ο αριθμός των κορυφών είναι πεπερασμένος και ο αλγόριθμος τερματίζει σε ένα πεπερασμένο αριθμό βημάτων, με την προϋπόθεση να ξέρουμε να αποφύγουμε έναν ενδεχόμενο κυκλισμό στην περίπτωση που η προς μεγιστοποίηση συνάρτηση θα έμενε αμετάβλητη.

3.7.2 Η μέθοδος Simplex με πίνακες

Περιγράφουμε τώρα τη μέθοδο Simplex με τη βοήθεια των πινάκων των συντελεστών του προβλήματος, το οποίο θα μας επιτρέψει εύκολα να γράψουμε τη μέθοδο υπό μορφή αλγορίθμου. Μερικά ακόμη σχόλια και κάποιοι συμβολισμοί θα μας χρειαστούν.

Θα λέμε ότι ένας πίνακας είναι διάστασης $q \times r$ αν έχει q γραμμές και r κολόνες.

Αφού ενδεχομένως εισαγάγουμε τις μεταβλητές απόκλισης αναζητούμε να επιλύσουμε το πρόβλημα:

$$\max z = cx \text{ με } \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

όπου $A = (a_{ij})$ είναι ένας $m \times p$ πίνακας με $p \geq m$, τάξης m, και στον οποίο a_j θα προσδιορίζει τη j κολόνα.

$c = (c_j)$ είναι ένας $1 \times p$ πίνακας γραμμή,

$b = (b_i)$ είναι ένας $m \times 1$ πίνακας κολόνα,

$x = (x_j)$ είναι ένα $p \times 1$ διάνυσμα κολόνα που εκφράζει τις μεταβλητές απόφασης.

Η επιλογή μίας βάσης με m μεταβλητές αντιστοιχεί στο να διαλέξει κανείς έναν αντιστρέψιμο υποπίνακα $B_{m\times m}$ του A. Θα συμβολίζουμε A_N τον σχηματιζόμενο από τις υπόλοιπες κολόνες πίνακα.

Το ίδιο C_B και C_N (αντίστοιχα x_B και x_N) θα προσδιορίζουν τις συνιστώσες του c (αντίστοιχα του x) που αντιστοιχούν στους δείκτες των βασικών μεταβλητών και των μη βασικών μεταβλητών (μεταβλητές εκτός βάσης).

Με αυτούς τους συμβολισμούς, μπορούμε να γράψουμε το πρόβλημα ως:

$$\boxed{\begin{array}{c} z = C_B x_B + C_N x_N \\ Ax = Bx_B + A_N x_N = b \end{array}}$$

Υποθέτουμε προς το παρόν ότι η βασική λύση, που αντιστοιχεί στον πίνακα B, και την οποία βρίσκουμε μηδενίζοντας το διάνυσμα x_N των μεταβλητών εκτός βάσης, είναι εφικτή, δηλαδή

$$x_B^* = B^{-1}b \geq 0$$

Ας αναζητήσουμε το σχετικό με τον πίνακα B λεξικό, το οποίο δίνει τις βασικές μεταβλητές x_B και την αντικειμενική συνάρτηση z σε συνάρτηση των μεταβλητών εκτός βάσης x_N.

Έχουμε λοιπόν:

$$x_B{=}B^{-1}(b - A_N x_N) \text{ ή ακόμη } x_B = x_B^* - B^{-1}A_N x_N \text{ με } x_B^* = B^{-1}b$$

Η συνάρτηση z γράφεται:

$$\begin{aligned} z{=}&C_B(x_B^* - B^{-1}A_N x_N) + C_N x_N \\ {=}&C_B x_B^* + (C_N - C_B B^{-1}A_N)x_N \\ {=}&z^* + (C_N - C_B B^{-1}A_N)x_N \text{ με } z^* = C_B x_B^* = C_B B^{-1}b \end{aligned}$$

Αν θεωρήσουμε το διάνυσμα γραμμή $y = C_B B^{-1}$ με m συνιστώσες, η αντικειμενική συνάρτηση γράφεται

$$z = z^* + (C_N - yA_N)x_N \text{ με } z^* = yb$$

Η j-στή συνιστώσα, του πίνακα γραμμή $C_N - yA_N$ είναι $c'_j = c_j - ya_j$ όπου a_j είναι η j-στή κολόνα του A. Ο όρος $(C_N - yA_N)x_N$ είναι η γραμμική μορφή της $z = \sum_{j \in N} c'_j x_j$.

Κατά συνέπεια $z = z^* + \sum_{j \in N}(c_j - ya_j)x_j$.

Αν $c_j - ya_j \leq 0$ για κάθε δείκτη j του N, τότε z^* είναι το μέγιστο της z. Επιτυγχάνεται για τις τιμές των μεταβλητών

$$x_B^* = B^{-1}b \text{ και } x_N^* = 0$$

Διαφορετικά, διαλέγουμε μία κολόνα a_j εισερχόμενη στη βάση για την οποία $c_j > ya_j$. Όταν οι μεταβλητές από x_N κρατούν την τιμή μηδέν εκτός της x_j η οποία παίρνει μία θετική τιμή, ας αναζητήσουμε πώς μετατρέπονται οι περιορισμοί $x_B \geq 0$.

Ο πίνακας $A_N x_n$ περιορίζεται στην κολόνα $x_j a_j$ και ο πίνακας $B^{-1}A_N x_N$ στην κολόνα $x_j B^{-1}a_j = x_j d$, έχοντας συμβολίσει $d = B^{-1}a_j$.

Οι περιορισμοί $x_B \geq 0$ γράφονται λοιπόν $x_B^* \geq x_j d$.

Είναι αυτόματα ικανοποιημένοι για τους δείκτες i (που δεν ανήκουν στο N) με $d_i \leq 0$. Για τους άλλους δείκτες, ο πιο ισχυρός περιορισμός αντιστοιχεί στο ελάχιστο

$\frac{x_i^*}{d_i}$. Διαλέγουμε ένα δείκτη i που δίνει αυτό το ελάχιστο για να καθορίσουμε την κολόνα a_i που εξέρχεται από τη βάση. Ο νέος πίνακας B επιτυγχάνεται αντικαθιστώντας την κολόνα a_i με a_j (σεβόμενοι πάντα τη σειρά των δεικτών). Μετά την επιλογή του i η νέα βάση αντιστοιχεί στην βασική εφικτή λύση για την οποία $x_j^* = \frac{x_i^*}{d_i}$. Οι παλιές μεταβλητές της βάσης x_B^* γίνονται $x_B^* - \left(\frac{x_i^*}{d_i}\right) d$ (μπορούμε επίσης να υπολογίσουμε τη νέα βασική λύση από την $x_B^* = B^{-1}b$) και η νέα τιμή της αντικειμενικής συνάρτησης είναι $C_B x_B^*$.

Δε μένει πλέον παρά να επαναλάβουμε μέχρι να επιτύχουμε μία βάση B για την οποία $c_j \leq ya_j$ για κάθε δείκτη j του N.

Μπορούμε λοιπόν να δώσουμε τον αλγόριθμο Simplex υπό την ακόλουθη μορφή για ένα πρόβλημα μεγιστοποίησης:

Αλγόριθμος Simplex

Αρχικοποίηση: Διάλεξε ένα υποπίνακα $B_{m\times m}$ του A, αντιστρέψιμο, έτσι ώστε $x_B^* = B^{-1}b \geq 0$ (αν δεν γνωρίζουμε έναν τέτοιο πίνακα, εφαρμόζουμε τη μέθοδο των δύο σταδίων που θα δούμε αργότερα).

Θέσε $N = \{j \mid \text{κολόνα } a_j \notin B\}$

1. Καθόρισε το διάνυσμα γραμμή $y = C_B B^{-1}$.

2. Αν $c_j \leq ya_j,\ \forall j \in N$ τότε $z_{max} = yb$ (ή $C_B x_B^*$). Μία βέλτιστη λύση δίδεται από το διάνυσμα x_B^* και $x_N^* = 0$.

   διαφορετικά επίλεξε ένα δείκτη $j \in N$ τέτοιο ώστε $c_j > ya_j$.

3. Καθόρισε το διάνυσμα κολόνα $d = B^{-1}a_j$.

4. Αν $d_i \leq 0, \forall i \notin N$ τότε η συνάρτηση z δεν είναι φραγμένη.

   διαφορετικά επίλεξε ένα δείκτη $i \notin N$ τέτοιο ώστε $d_i > 0$ και $\frac{x_i^*}{d_i}$ να είναι ελάχιστο.

5. Ενημέρωση:

   Παλαιές μεταβλητές εκτός βάσης:

$$x_k^* = 0 \text{ για κάθε } k \in N - \{j\}; x_j^* = \frac{x_i^*}{d_i}$$

Παλαιές μεταβλητές της βάσης:

$$x_B^* = x_B^* - \frac{x_i^*}{d_i} d \text{ δηλαδή } x_k^* = x_k^* - \frac{x_i^*}{d_i} d_k \text{ για κάθε } k \notin N$$

$$B = (B - a_i) \bigcup a_j; \quad N = (N - j) \bigcup i$$

$$z^* = C_B x_B^* \text{ με τη νέα βάση } B.$$

Επέστρεψε στο βήμα 1.

Ας παρατηρήσουμε ότι όταν εφαρμόζουμε τον αλγόριθμο με το χέρι, είναι μερικές φορές πιο απλό για να υπολογίσουμε τα διανύσματα y και d να επιλύσουμε τα συστήματα $yB = C_B$ και $Bd = a_j$ μάλλον απ΄ ότι να υπολογίσουμε B^{-1} και να κάνουμε τα γινόμενα πινάκων.

Ας εφαρμόσουμε τη μέθοδο στο συνηθισμένο μας παράδειγμα της εταιρίας και των συμβολαίων.

Αρχικοποίηση

Τα δεδομένα του προβλήματος είναι:

$$A = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{pmatrix} = \begin{pmatrix} 5 & 3 & 1 & 0 & 0 \\ 2 & 3 & 0 & 1 & 0 \\ 1 & 3 & 0 & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 30 \\ 24 \\ 18 \end{pmatrix} \quad c = \begin{pmatrix} 4 & 3 & 0 & 0 & 0 \end{pmatrix}$$

Έστω $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ αντιστρέψιμος και τέτοιος ώστε $x_B^* = \begin{pmatrix} x_3^* = 30 \\ x_4^* = 24 \\ x_5^* = 18 \end{pmatrix}$.
Από όπου $N = \{1, 2\}, C_B = (0, 0, 0)$ και $z^* = C_B x_B^* = 0$.

Πρώτο βήμα:

1. $y = C_B B^{-1} = (0, 0, 0)$, $B = (a_3 \quad a_4 \quad a_5)$
2. $ya_j = 0, \forall j = 1, 2$. Έχουμε $c_1 = 4 > ya_1$.

Ας επιλέξουμε $j = 1$. Το διάνυσμα a_1 εισέρχεται στη βάση.

3. $d = \begin{pmatrix} d_3 \\ d_4 \\ d_5 \end{pmatrix} = B^{-1}a_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}$.

4. $\frac{x_3^*}{d_3} = 6, \frac{x_4^*}{d_4} = 12, \frac{x_5^*}{d_5} = 18$ από όπου $i = 3$. Το διάνυσμα a_3 εξέρχεται από τη βάση.

5. Οι τιμές των παλιών μεταβλητών εκτός βάσης γίνονται:

   $x_1^* = \frac{x_3^*}{d_3} = 6, x_2^* = 0$

   Οι τιμές των παλιών μεταβλητών εντός βάσης γίνονται

   $x_B^* = x_B^* - \left(\frac{x_3^*}{d_3}\right) d = \begin{pmatrix} 30 \\ 24 \\ 18 \end{pmatrix} - 6 \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}$ δηλαδή $x_3^* = 0, x_4^* = 12, x_5^* = 12$.

   Νέα Βάση: $B = (a_1 \quad a_4 \quad a_5) = \begin{pmatrix} 5 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$,

   $N = \{2, 3\}, C_B = (4, 0, 0), x_B^* = \begin{pmatrix} x_1^* = 6 \\ x_4^* = 12 \\ x_5^* = 12 \end{pmatrix}$ με $z^* = C_B x_B^* = 24$.

Δεύτερο βήμα

1. $yB = C_B$ δίνει $\begin{cases} 5y_1 + 2y_2 + y_3 = 4 \\ y_2 = 0 \\ y_3 = 0 \end{cases}$ δηλ. $y = (4/5 \quad 0 \quad 0)$.

2. $c_2 = 3 > ya_2 = \frac{12}{5}$. Ας πάρουμε $j = 2$, και το διάνυσμα a_2 εισέρχεται στη βάση.

3. $Bd = a_2$, με $d = \begin{pmatrix} d_1 \\ d_4 \\ d_5 \end{pmatrix}$ δίνει $\begin{cases} 5d_1 = 3 \\ 2d_1 + d_4 = 3 \\ d_1 + d_5 = 3 \end{cases}$ δηλαδή $\begin{pmatrix} d_1 = \frac{3}{5} \\ d_4 = \frac{9}{5} \\ d_5 = \frac{12}{5} \end{pmatrix}$.

4. $\frac{x_1^*}{d_1} = 10, \frac{x_4^*}{d_4} = \frac{20}{3}, \frac{x_5^*}{d_5} = 5$ από όπου $i = 5$. Το διάνυσμα a_5 εξέρχεται από τη βάση.

5. $x_2^* = \frac{x_5^*}{d_5} = 5, x_3^* = 0 \qquad \begin{pmatrix} x_1^* \\ x_4^* \\ x_5^* \end{pmatrix} = \begin{pmatrix} 6 \\ 12 \\ 12 \end{pmatrix} - 5 \begin{pmatrix} 3/5 \\ 9/5 \\ 12/5 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 0 \end{pmatrix}$

$$B = (a_1 \quad a_2 \quad a_4) = \begin{pmatrix} 5 & 3 & 0 \\ 2 & 3 & 1 \\ 1 & 3 & 0 \end{pmatrix}, \quad N = \{3, 5\}$$

$$C_B = (4, 3, 0), x_B^* = \begin{pmatrix} x_1^* = 3 \\ x_2^* = 5 \\ x_4^* = 3 \end{pmatrix}, \quad z^* = C_B x_B^* = 27.$$

Τρίτο βήμα:

1. $yB = C_B$ δίνει $\begin{cases} 5y_1 + 2y_2 + y_3 = 4 \\ 3y_1 + 3y_2 + 3y_3 = 3 \\ y_2 = 0 \end{cases}$ δηλ. $y = (3/4 \quad 0 \quad 1/4)$.

2. $c_3 = 0 < ya_3 = (3/4 \quad 0 \quad 1/4)(1 \quad 0 \quad 0)^t = \frac{3}{4}$

   $c_5 = 0 < ya_5 = (3/4 \quad 0 \quad 1/4)(0 \quad 0 \quad 1)^t = \frac{1}{4}$

Ο αλγόριθμος τελειώνει και έχουμε $z^* = 27$ το βέλτιστο της αντικειμενικής συνάρτησης το οποίο επιτυγχάνεται για τις τιμές 3 και 5 των μεταβλητών x_1 και x_2 αντίστοιχα.

Καθορισμός μίας εφικτής βασικής λύσης

Είδαμε ότι η μέθοδος Simplex επιτρέπει, ξεκινώντας από μία βασική εφικτή λύση, να συμπεράνουμε αν το πρόβλημα μεγιστοποίησης επιδέχεται μία λύση και να βρει αυτή τη λύση.

Αλλά μπορεί να συμβεί να διαθέτουμε μία βάση στην εκκίνηση για την οποία η βασική λύση να μην είναι εφικτή. Το πρόβλημα που τίθεται τότε είναι να γνωρίζουμε αν υπάρχουν εφικτές λύσεις, δηλαδή αν οι περιορισμοί είναι συμβατοί και αν ναι, να κατασκευάσουμε μία βασική εφικτή λύση.

Όπως κάναμε προηγουμένως, περιγράφουμε τη μέθοδο σε ένα παράδειγμα.

Έστω το γραμμικό πρόγραμμα:

$$\max z = x_1 - x_2 + x_3$$
$$\begin{cases} 2x_1 - x_2 + 2x_3 & \leq 4 \\ 2x_1 - 3x_2 + x_3 & \leq -5 \\ -x_1 + x_2 - 2x_3 & \leq -1 \\ x_i,\ i & = 1,2,3 \geq 0 \end{cases}$$

Μπορούμε, όπως στο προηγούμενο παράδειγμα, να θεωρήσουμε τη βάση συνιστούμενη από τις μεταβλητές απόκλισης x_4, x_5, x_6 που εισαγάγουμε για να μετατρέψουμε τους περιορισμούς σε ισότητες, το οποίο θέτει το πρόβλημα στη μορφή:

$$P: \left\| \begin{array}{l} \max z = x_1 - x_2 + x_3 \\ \text{με} \begin{cases} x_4 = 4 - 2x_1 + x_2 - 2x_3 \\ x_5 = -5 - 2x_1 + 3x_2 - x_3 \\ x_6 = -1 + x_1 - x_2 + 2x_3 \\ x_i \geq 0, \quad 1 \leq i \leq 6 \end{cases} \end{array} \right.$$

Αυτό το λεξικό δεν είναι δυστυχώς εφικτό αφού η βασική λύση (0,0,0,4,-5,-1) δεν ικανοποιεί τους περιορισμούς $x_5 \geq 0$ και $x_6 \geq 0$.

Πρώτο βήμα

Ας εισάγουμε στις ισότητες των περιορισμών που αντιστοιχούν σε αρνητικές βασικές μεταβλητές μία νέα θετική μεταβλητή x_0, και ας θεωρήσουμε το νέο πρόβλημα:

$$P': \left\| \begin{array}{l} \max W = -x_0 \\ \text{με} \begin{cases} x_4 = 4 \qquad - 2x_1 + x_2 - 2x_3 \\ x_5 = -5 + x_0 - 2x_1 + 3x_2 - x_3 \\ x_6 = -1 + x_0 + x_1 - x_2 + 2x_3 \\ x_i \geq 0, \quad i = 0, 1, \ldots, 6 \end{cases} \end{array} \right.$$

Πρόταση: P επιδέχεται μία εφικτή λύση αν και μόνο αν η βέλτιστη λύση του P' είναι $W_{max} = 0$. Επιπλέον, αν P είναι εφικτό, τότε επιδέχεται μία βασική εφικτή λύση.

Απόδειξη

1. Αν $(x_1^*, \ldots, x_6^*)$ είναι εφικτή λύση του P, τότε το διάνυσμα $(0, x_1^*, \ldots, x_6^*)$ είναι μία εφικτή λύση του P' για την οποία η συνάρτηση W παίρνει την τιμή 0. Αφού $W \leq 0$, έχουμε ότι $W_{max} = 0$.

2. Αντίστροφα, αν το μέγιστο της W είναι μηδέν και επιτυγχάνεται για $(0, x_1^*, \ldots, x_6^*)$ τότε $(x_1^*, \ldots, x_6^*)$ είναι μία εφικτή λύση του P. Επί πλέον, σε αυτή την περίπτωση $(0, x_1^*, \ldots, x_6^*)$ επιτυγχάνεται από τη μέθοδο Simplex σαν βασική λύση ενός λεξικού εφικτού του P' και κατά συνέπεια $(x_1^*, \ldots, x_6^*)$ είναι βασική λύση του αντίστοιχου (μηδενίζοντας x_0) εφικτού λεξικού του P.

Για να επιλύσουμε το πρόβλημα P', *εισάγουμε τη μεταβλητή* x_0 *στη βάση* $\{x_4, x_5, x_6\}$ *παρά το ότι ο συντελεστής του* x_0 *μέσα στη* W *είναι αρνητικός, και εξάγουμε τη μεταβλητή η οποία επιβάλλει στη* x_0 *τον περιορισμό τον πιο απαιτητικό.* Εδώ ο περιορισμός $x_5 \geq 0$ επιβάλλει $x_0 \geq 5$, και ο περιορισμός $x_6 \geq 0$ επιβάλλει $x_0 \geq 1$. Εξάγουμε λοιπόν την x_5.

Το νέο λεξικό με τη βάση $\{x_0, x_4, x_6\}$ γράφεται:

$$\begin{cases} x_0 = 5 + 2x_1 - 3x_2 + x_3 + x_5 \\ x_4 = 4 - 2x_1 + x_2 - 2x_3 \\ x_6 = 4 + 3x_1 - 4x_2 + 3x_3 + x_5 \\ W = -5 - 2x_1 + 3x_2 - x_3 - x_5 \\ \text{με } x_i \geq 0, 0 \leq i \leq 6 \end{cases}$$

Το λεξικό είναι εφικτό και μπορούμε να χρησιμοποιήσουμε κανονικά πλέον με τη μέθοδο Simplex για να επιλύσουμε το P'. Κατά συνέπεια, αφού το W είναι φραγμένο από 0, θα έχουμε ή $W_{max} = 0$ και έτσι θα διαθέτουμε μία βασική εφικτή λύση για το P, η οποία επιτρέπει να ολοκληρώσουμε την επίλυση του προβλήματος με τη μέθοδο Simplex, ή $W_{max} < 0$ και το πρόβλημα P δεν είναι εφικτό.

Για το παράδειγμά μας, έχουμε τα ακόλουθα διαδοχικά λεξικά:

$$\text{B} = \{x_0, x_2, x_4\} = \begin{cases} x_0 = 2 - \frac{x_1}{4} - \frac{5}{4}x_3 + \frac{x_5}{4} + \frac{3}{4}x_6 \\ x_2 = 1 + \frac{3}{4}x_1 + \frac{3}{4}x_3 + \frac{x_5}{4} - \frac{x_6}{4} \\ x_4 = 5 - \frac{5}{4}x_1 - \frac{5}{4}x_3 + \frac{x_5}{4} - \frac{x_6}{4} \\ W = -2 + \frac{x_1}{4} + \frac{5}{4}x_3 - \frac{x_5}{4} - \frac{3}{4}x_6 \end{cases}$$

με $x_1 \geq 0$, για $0 \leq i \leq 6$

$$\mathrm{B} = \{x_2, x_3, x_4\} = \begin{cases} x_2 = \frac{11}{5} - \frac{3}{5}x_0 + \frac{3}{5}x_1 + \frac{2}{5}x_5 + \frac{x_6}{5} \\ x_3 = \frac{8}{5} - \frac{4}{4}x_0 - \frac{x_1}{5} + \frac{x_5}{5} + \frac{3}{5}x_6 \\ x_4 = 3 + x_0 - x_1 - x_6 \\ W = -x_0 \end{cases}$$

με $x_i \geq 0$, για $0 \leq i \leq 6$

Το τελευταίο εφικτό λεξικό δείχνει ότι $W_{max} = 0$ και ότι αυτή η τιμή επιτυγχάνεται για

$$\left(x_0^* = 0, x_1^* = 0, x_2^* = \frac{11}{5}, x_3^* = \frac{8}{5}, x_4^* = 3, x_5^* = 0, x_6^* = 0\right).$$

Κατά συνέπεια $(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*) = \left(0, \frac{11}{5}, \frac{8}{5}, 3, 0, 0\right)$ είναι μία βασική εφικτή λύση του P που αντιστοιχεί στη βάση $\{x_2, x_3, x_4\}$.

Δεύτερο στάδιο

Το πρόβλημα P χρησιμοποιώντας τις εκφράσεις των x_2, x_3 και x_4 στην αντικειμενική συνάρτηση, γράφεται τώρα υπό τη μορφή:

$$\max z(= x_1 - x_2 + x_3) = -\frac{3}{5} + \frac{x_1}{5} - \frac{x_5}{5} + \frac{2}{5}x_6$$

με

$$\begin{cases} x_2 = \frac{11}{5} + \frac{3}{5}x_1 + \frac{2}{5}x_5 + \frac{x_6}{5} \\ x_3 = \frac{8}{5} - \frac{x_1}{5} + \frac{x_5}{5} + \frac{3}{5}x_6 \\ x_4 = 3 - x_1 - x_6 \end{cases} \quad \text{και } x_i \geq 0 \text{ για } 1 \leq i \leq 6$$

Τελειώνουμε την επίλυσή του με τη μέθοδο Simplex εισάγοντας τη μεταβλητή x_6 στη βάση στη θέση της μεταβλητής x_4. Έχουμε λοιπόν:

$$\mathrm{B} = \{x_2, x_3, x_6\} = \begin{cases} x_2 = \frac{14}{5} - \frac{2}{5}x_1 - \frac{x_4}{5} + \frac{2}{5}x_5 \\ x_3 = \frac{17}{5} - \frac{4}{5}x_1 - \frac{3}{5}x_4 + \frac{x_5}{5} \\ x_6 = 3 - x_1 - x_4 \\ z = \frac{3}{5} - \frac{x_1}{5} - \frac{3}{5}x_4 - \frac{x_5}{5} \end{cases}$$

Καταλήγουμε λοιπόν ότι $z_{max} = \frac{3}{5}$, τιμή που έχει επιτευχθεί από $x_1 = 0,\ x_2 = \frac{14}{5},\ x_3 = \frac{17}{5},\ x_4 = 0,\ x_5 = 0,\ x_6 = 3$, το οποίο τελειώνει την επίλυση του P.

Παρατήρηση

Θα δούμε παρακάτω ότι μερικά προβλήματα για τα οποία η γνωστή στην αρχή βασική λύση δεν είναι εφικτή μπορούν να επιλυθούν μόνο σε ένα στάδιο ξεκινώντας από το δυϊκό πρόβλημα, χωρίς να χρησιμοποιήσουμε την τεχνική του βοηθητικού προβλήματος P' που μόλις περιγράψαμε.

Ανακεφαλαιώνοντας, κάθε πρόβλημα γραμμικού προγραμματισμού ανήκει σε μία από τις παρακάτω τρεις κατηγορίες:

- Το πρόβλημα δεν είναι εφικτό (περιορισμοί ασυμβίβαστοι).
- Το πρόβλημα είναι εφικτό αλλά όχι φραγμένο.
- Το πρόβλημα είναι εφικτό και η αντικειμενική συνάρτηση επιδέχεται ένα βέλτιστο.

Γνωρίζουμε λοιπόν τώρα, τον τρόπο κατάταξης ένα δεδομένου προβλήματος σε μία από τις τρεις κατηγορίες, και στο τρίτο ενδεχόμενο να το επιλύσουμε. Σημειώνουμε ότι στη περιγραφή της μεθόδου Simplex θεωρήσαμε ένα πρόβλημα μεγιστοποίησης. Αν το πρόβλημα μας ήταν ελαχιστοποίησης μπορούμε να θεωρήσουμε την συνάρτηση $-z$ και να εφαρμόζουμε τον αλγόριθμο όπως τον περιγράψαμε. Εν τούτοις όμως μικρές προφανείς αλλαγές στον αλγόριθμο επιτρέπουν να τον χρησιμοποιήσουμε απ΄ ευθείας για προβλήματα ελαχιστοποίησης. Οι αλλαγές αυτές αφορούν το βήμα 2 και μπορείτε πολύ εύκολα να τις επισημάνετε.

3.8 Δυϊκή θεωρία

Σε αυτή την ενότητα επιστρέφουμε στην περίπτωση που το πρόβλημά μας στην αρχή είναι του πρώτου τύπου, δηλαδή όπου οι m περιορισμοί έχουν δοθεί υπό τη μορφή ανισοτήτων.

Ορισμός

Θεωρούμε το πρόβλημα P:

$$\max z = \sum_{j=1}^{n} c_j x_j$$

με τους περιορισμούς:

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j \leq b_i & \text{για } 1 \leq i \leq m \\ x_j \geq 0 & \text{για } 1 \leq j \leq n \end{cases}$$

Αντιστοιχούμε στο πρόβλημα P το ακόλουθο πρόβλημα $D(P)$ ή απλά D(αν δεν υπάρχει πρόβλημα σύγχυσης).

$$\min W = \sum_{i=1}^{m} b_i y_i$$

με τους περιορισμούς:

$$\begin{cases} \sum_{i=1}^{m} a_{ij}y_i \geq c_j & \text{για } 1 \leq j \leq n \\ y_i \geq 0 & \text{για } 1 \leq i \leq m \end{cases}$$

Το πρόβλημα D καλείται Δυϊκό του P (Dual).

Αν συμβολίσουμε Q^t τον ανάστροφο ενός πίνακα Q τότε τα δύο προβλήματα P και D (Primal / Dual) γράφονται υπό τη μορφή πινάκων:

$$P : \left\| \begin{array}{l} \max z = cx \\ \text{με} \\ Ax \leq b \\ x \geq 0 \end{array} \right.$$

$$D : \left\| \begin{array}{l} \min W = b^t y \\ \text{με} \\ A^t y \geq c^t \\ y \geq 0 \end{array} \right.$$

όπου x είναι το διάνυσμα κολόνα διάστασης n, A ένας $m \times n$ πίνακας, c ένα διάνυσμα γραμμή διάστασης n, b ένα διάνυσμα κολόνα διάστασης m, y ένα διάνυσμα κολόνα διάστασης m, A^t ο ανάστροφος πίνακας του A διαστάσεων $n \times m$, c^t ο ανάστροφος του διανύσματος c δηλαδή διάνυσμα κολόνα και b^t ο ανάστροφος του διανύσματος b δηλαδή διάνυσμα γραμμή διάστασης m.

Παράδειγμα

Ας πάρουμε το πρόβλημα P που μελετήσαμε στις προηγούμενες ενότητες:

$$P: \left\| \begin{array}{l} \max z = 4x_1 + 3x_2, n = 2, m = 3 \\ \text{με} \\ 5x_1 + 3x_2 \leq 30 \\ 2x_1 + 3x_2 \leq 24 \\ x_1 + 3x_2 \leq 18 \\ x_1, x_2 \geq 0 \end{array} \right.$$

Το Δυϊκό πρόβλημα είναι:

$$D: \left\| \begin{array}{l} \min W = 30y_1 + 24y_2 + 18y_3 \\ \text{με} \\ 5y_1 + 2y_2 + y_3 \geq 4 \\ 3y_1 + 3y_2 + 3y_3 \geq 3 \\ y_1, y_2, y_3 \geq 0 \end{array} \right.$$

Για να επιλύσουμε το P όπως γνωρίζουμε μέχρι τώρα, εισάγουμε τις m μεταβλητές απόκλισης $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij}x_j$ και για να επιλύσουμε το D, τις n μεταβλητές απόκλισης $y_{m+j} = \sum_{i=1}^{m} a_{ij}y_j - c_j$ οι οποίες είναι όλες θετικές ή μηδενικές. Μπορούμε να παρατηρήσουμε ότι στον i-στο περιορισμό του P, στον οποίο υπεισέρχεται η μεταβλητή x_{n+i}, αντιστοιχεί η μεταβλητή y_i. Επίσης στον j-στο περιορισμό του D, στον οποίο υπεισέρχεται η μεταβλητή y_{m+j}, αντιστοιχεί η μεταβλητή x_j. Αυτή η συμμετρία, που θα συναντήσουμε καθ' όλη τη διάρκεια της μελέτης της Δυϊκότητας, περιγράφει το ακόλουθο θεώρημα.

Θεώρημα: Το Δυϊκό πρόβλημα του Δυϊκού του P είναι το ίδιο το πρόβλημα P

$$\mathbf{D(D(P)) = P}$$

Απόδειξη: Το πρόβλημα $D(P)$ (δυϊκό του P), μπορεί να γραφεί:

$$\max \sum_{i=1}^{m} (-b_i)y_i$$

με τους περιορισμούς:

$$\begin{cases} \sum_{i=1}^{m}(-a_{ij})y_i \leq -c_j & \text{για } 1 \leq j \leq n \\ y_i \geq 0 & \text{για } 1 \leq i \leq m \end{cases}$$

Το πρόβλημα λοιπόν $D(D(P))$ γράφεται:

$$\min \sum_{j=1}^{n}(-c_j)t_j$$

με τους περιορισμούς:

$$\begin{cases} \sum_{j=1}^{n}(-a_{ij})t_j \geq -b_i & \text{για } 1 \leq i \leq m \\ t_j \geq 0 & \text{για } 1 \leq j \leq n \end{cases}$$

ή ακόμη:

$$\max \sum_{j=1}^{n} c_j t_j$$

με τους περιορισμούς:

$$\begin{cases} \sum_{j=1}^{n} a_{ij}t_j \leq b_i & \text{για } 1 \leq i \leq m \\ t_j \geq 0 & \text{για } 1 \leq j \leq n \end{cases}$$

όπου αναγνωρίζουμε το πρόβλημα P.

Θεωρήματα Δυϊκότητας

Τα παρακάτω θεωρήματα δείχνουν πως η επίλυση ενός προβλήματος και η επίλυση του δυϊκού του, είναι ισοδύναμες. Η ισοδυναμία των επιλύσεων απορρέει από την συμμετρία των προβλημάτων.

Θεώρημα 3.8.1 *(1ο θεώρημα δυϊκότητας)*

Αν $(x_1^*, x_2^*, \ldots, x_n^*)$ *και* $(y_1^*, y_2^*, \ldots, y_m^*)$ *είναι εφικτές λύσεις για το αρχικό πρόβλημα*

(Primal) P και για το δυϊκό του (Dual) D αντίστοιχα, τότε

$$\sum_{j=1}^{n} c_j x_j^* \leq \sum_{i=1}^{m} b_i y_i^*$$

Επί πλέον, όταν έχουμε ισότητα, αυτές οι λύσεις είναι βέλτιστες για P και D αντίστοιχα.

Απόδειξη

$$\sum_{j=1}^{n} c_j x_j^* \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i^* \right) x_j^* = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j^* \right) y_i^* \leq \sum_{i=1}^{m} b_i y_i^*$$

Κατά συνέπεια, $\sum_{i=1}^{m} b_i y_i^*$ είναι ένα άνω φράγμα για τη συνάρτηση z του P και $\sum_{j=1}^{n} c_j x_j^*$ ένα κάτω φράγμα για την συνάρτηση W του D.

Αν η ισότητα ισχύει, το άνω φράγμα της z που επιτυγχάνεται για την τιμή z^* είναι ένα μέγιστο. Έχουμε λοιπόν $z_{\max} = z^*$ και $\{x_j^*\}_{1 \leq j \leq n}$ είναι μία βέλτιστη λύση του P. Το ίδιο, το κάτω φράγμα της W, που επιτυγχάνεται από W^*, είναι ένα ελάχιστο και έχουμε λοιπόν $W_{\min} = W^*$ και $\{y_i^*\}_{1 \leq i \leq m}$ είναι μία βέλτιστη λύση για το πρόβλημα D.

Πόρισμα

Αν το πρόβλημα P (Primal) είναι εφικτό αλλά όχι φραγμένο, το δυϊκό του πρόβλημα, (Dual) δεν είναι εφικτό και αντίστροφα.

Απόδειξη

Από την απόδειξη του προηγούμενου θεωρήματος έχουμε ότι $\sum_{j=1}^{n} c_j x_j^* \leq \sum_{i=1}^{m} b_i y_i^*$ και επομένως αν το πρόβλημα D ήταν εφικτό τότε το P θα ήταν φραγμένο.

Ας σημειώσουμε ότι μπορεί και τα δύο προβλήματα P και D να μην είναι εφικτά. Επίσης αν το P είναι εφικτό και φραγμένο τότε έχει βέλτιστη λύση.

Το παρακάτω θεώρημα δίδει ένα αντίστροφο αποτέλεσμα του προηγούμενου θεωρήματος.

Θεώρημα 3.8.2 *(2ο θεώρημα δυϊκότητας)*

Αν το αρχικό πρόβλημα P (Primal) έχει μία βέλτιστη λύση, το δυϊκό πρόβλημα D (Dual) έχει επίσης μία βέλτιστη λύση και έχουμε $z_{\max} = W_{\min}$.

Απόδειξη

Ας υποθέσουμε ότι το πρόβλημα P επιδέχεται βέλτιστες λύσεις, δηλαδή ότι είναι εφικτό και φραγμένο. Ας εφαρμόσουμε σε αυτό τη μέθοδο Simplex, μετά από την εισαγωγή των m μεταβλητών απόκλισης $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij}x_j$.

Στο τελευταίο λεξικό, η αντικειμενική συνάρτηση γράφεται $z = z^* - \sum_{k=1}^{n+m} \eta_k x_k$ όπου οι συντελεστές η_k που αντιστοιχούν στις μεταβλητές της βάσης είναι μηδενικοί. Οι άλλοι συντελεστές είναι θετικοί ή μηδενικοί, διαφορετικά θα μπορούσαμε να βελτιώσουμε την αντικειμενική συνάρτηση z. Αν διαχωρίσουμε τις μεταβλητές απόφασης και απόκλισης έχουμε:

$$z = z^* - \sum_{j=1}^{n} \eta_j x_j - \sum_{i=1}^{m} \eta_{n+i} \left(b_i - \sum_{j=1}^{n} a_{ij}x_j \right) = z^* - \sum_{i=1}^{m} b_i \eta_{n+i} + \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij}\eta_{n+i} - n_j \right) x_j$$

Αφού $z = \sum_{j=1}^{n} c_j x_j$ και από την ανεξαρτησία των μεταβλητών x_j έχουμε:

$$\begin{cases} z^* = \sum_{i=1}^{m} b_i \eta_{n+i} \\ \sum_{i=1}^{m} a_{ij}\eta_{n+i} = c_j + \eta_j \quad \text{για } 1 \le j \le n \end{cases}$$

ή ακόμη, αφού $\eta_j \ge 0$

$$\begin{cases} z^* = \sum_{i=1}^{m} b_i \eta_{n+i} \\ \sum_{i=1}^{n} a_{ij}\eta_{n+i} \ge c_j \quad \text{για } 1 \le j \le n \end{cases}$$

Άρα οι m αριθμοί η_{n+i} συνιστούν μία εφικτή λύση του Δυϊκού για την οποία W παίρνει την τιμή z^*. Μετά το θεώρημα 3.8.1 αυτή η λύση είναι βέλτιστη και $z_{\max} = W_{\min} = z^*$.

Παρατηρήσεις

1. Η απόδειξη δείχνει ότι η επίλυση του αρχικού προβλήματος (Primal) επιτρέπει να γνωρίσουμε όχι μόνο το ελάχιστο της W, αλλά επίσης και μία βέλτιστη λύση του D, χάριν των σχέσεων $y_i^* = \eta_{n+i}$, όπου οι η_{n+i} είναι δεδομένες από το τελευταίο λεξικό του P.

2. Όταν χρησιμοποιούμε τη μέθοδο Simplex υπό τη μορφή πινάκων για την επίλυση του P, οι συνιστώσες y_i του τελευταίου διανύσματος γραμμή y, που υπολογίσαμε, συνιστούν μία βέλτιστη λύση του προβλήματος D.

Παράδειγμα

Ας πάρουμε πάντα το ίδιο αριθμητικό παράδειγμα με $n = 2, m = 3$.

$$\max z = 4x_1 + 3x_2$$

με τους περιορισμούς:

$$\begin{cases} 5x_1 + 3x_2 & \leq 30 \\ 2x_1 + 3x_2 & \leq 24 \\ x_1 + 3x_2 & \leq 18 \\ x_1, x_2 & \leq 0 \end{cases}$$

Το τελευταίο λεξικό κατά την επίλυσή του έδινε $z = 27 - \frac{3}{4}x_3 - \frac{1}{4}x_5$, δηλαδή $\eta_{n+1} = \frac{3}{4}, \eta_{n+2} = 0, \eta_{n+3} = \frac{1}{4}$. Το δυϊκό πρόβλημα λοιπόν είναι λυμένο με $W_{\min} = 27$ και έχει βέλτιστη λύση το διάνυσμα $(y_1^* = \frac{3}{4}, y_2^* = 0, y_3^* = \frac{1}{4})$.

Παρατήρηση

Σαν συνέπεια του προηγούμενου θεωρήματος και της συμμετρίας μεταξύ P και D έχουμε ότι:

Το αρχικό πρόβλημα P επιδέχεται μια βέλτιστη λύση αν και μόνο αν το δυϊκό πρόβλημα επιδέχεται μια βέλτιστη λύση.

Τα εκτεθέντα προηγουμένως δείχνουν ότι υπάρχουν τέσσερις δυνατότητες για τα προβλήματα P και D:

- P είναι μη εφικτό και D είναι μη εφικτό.
- P είναι μη εφικτό και D είναι εφικτό μη φραγμένο.
- P είναι εφικτό μη φραγμένο και D είναι μη εφικτό.
- P επιδέχεται μια βέλτιστη λύση και D επιδέχεται μια βέλτιστη λύση.

Στο τελευταίο ενδεχόμενο, η βέλτιστη επίλυση του ενός από τα προβλήματα δίδει επίσης τη βέλτιστη λύση για το άλλο.

Τα δύο τελευταία θεωρήματα αυτής της ενότητας που θα δούμε δίνουν ελέγχους (tests) βέλτιστου (optimality), δηλαδή επιτρέπουν να αναγνωρίσουμε αν δεδομένες εφικτές λύσεις $\{x_j^*\}$ και $\{y_i^*\}$ των P και D αντίστοιχα είναι βέλτιστες, ή απλά αν μια λύση $\{x_j^*\}$ του P είναι βέλτιστη.

Θεώρημα 3.8.3

Έστω $(x_1^*, x_2^*, \ldots, x_n^*)$ και $(y_1^*, y_2^*, \ldots, y_m^*)$ εφικτές λύσεις για το αρχικό πρόβλημα (Primal) P και για το Δυϊκό του (Dual) D αντίστοιχα. Οι δύο λύσεις είναι βέλτιστες για P και D αντίστοιχα αν και μόνο αν οι δύο παρακάτω συνθήκες ικανοποιούνται:

$$\begin{aligned} &\text{α) για } j = 1, \ldots, n: \quad x_j^* = 0 \text{ ή } y_{m+j}^* = 0 \ (\text{δηλ. } \textstyle\sum_{i=1}^m a_{ij} y_i^* = c_j) \\ &\text{β) για } i = 1, \ldots, m: \quad y_i^* = 0 \text{ ή } x_{n+i}^* = 0 \ (\text{δηλ. } \textstyle\sum_{j=1}^n a_{ij} x_j^* = b_i) \end{aligned}$$

Απόδειξη

Ας σημειώσουμε ότι σε αυτές τις συνθήκες (α) και (β), το ''ή'' δεν είναι αποκλειστικό.

1. Αν οι συνθήκες (α) και (β) ικανοποιούνται τότε οι δύο ανισότητες στην απόδειξη του θεωρήματος της δυϊκότητας (Θεώρημα 3.1) γίνονται ισότητες και από το ίδιο θεώρημα οι λύσεις είναι βέλτιστες.

2. Αντίστροφα, αν οι λύσεις $\{x_j^*\}_{1 \le j \le n}$ και $\{y_i^*\}_{1 \le i \le m}$ είναι βέλτιστες για τα προβλήματα P και D, τότε $\max z = \min W$ από το δεύτερο θεώρημα της δυϊκότητας (Θεώρημα 3.2) με $\max z = \sum_{j=1}^n c_j x_j^*$ και $\min W = \sum_{i=1}^m b_i y_i^*$.

Οι ανισότητες στην απόδειξη του πρώτου θεωρήματος της δυϊκότητας, είναι λοιπόν ισότητες και έχουμε από τα δύο αριστερότερα μέλη:

$$\sum_{j=1}^{n}\left(\sum_{i=1}^{m}a_{ij}y_i^*-c_j\right)x_j^*=0 \text{ με } \left(\sum_{i=1}^{m}a_{ij}y_i^*-c_j\right)x_j^*\geq 0$$

για κάθε $j=1,\ldots,n$, απ΄ όπου

$$x_j^*=0 \text{ ή } \sum_{i=1}^{m}a_{ij}y_i^*=c_j \text{ για κάθε } j.$$

Επίσης έχουμε από την ισότητα των δύο δεξιοτέρων μελών στη σχέση στην απόδειξη του θεωρήματος 3.8.1 ότι

$$\sum_{i=1}^{m}\left(b_i-\sum_{j=1}^{n}a_{ij}x_j^*\right)y_i^*=0 \text{ με } \left(b_i-\sum_{j=1}^{n}a_{ij}x_j^*\right)y_i^*\geq 0$$

για κάθε $i=1,\ldots,m$ απ΄ όπου

$$y_i^*=0 \text{ ή } \sum_{j=1}^{n}a_{ij}x_j^*=b_i \text{ για κάθε } i.$$

Το θεώρημα δίδει αυτόματα σαν πόρισμα τον έλεγχο βέλτιστου (optimality test) μιας λύσης του P ως εξής:

Θεώρημα 3.8.4

Μια εφικτή λύση $(x_1^*,\ldots,x_n^*)$ του προβλήματος P είναι βέλτιστη αν και μόνο αν υπάρχουν m αριθμοί $(y_1^*,\ldots,y_m^*)$ τέτοιοι ώστε:

$$\begin{cases} y_i^*\geq 0 \text{ και } y_i^*=0 & \text{αν } \sum_{j=1}^{n}a_{ij}x_j^*<b_i \\ \sum_{i=1}^{m}a_{ij}y_i^*\geq c_j \text{ και } \sum_{i=1}^{m}a_{ij}y_i^*=c_j & \text{αν } x_j^*>0 \end{cases}$$

Θα τελειώσουμε με μία πρακτική εφαρμογή της δυϊκότητας.

Προβλήματα δυϊκά - εφικτά

Τα προηγούμενα θεωρήματα δείχνουν ότι είναι ισοδύναμο να λύσει κανείς ένα πρόβλημα ή το δυϊκό του. Αυτό μπορεί να είναι ενδιαφέρον για να αποφύγουμε τη μέθοδο

των δύο φάσεων, που είδαμε στα προηγούμενα, για την επίλυση μερικών προβλημάτων για τα οποία δεν γνωρίζουμε βασική εφικτή λύση.

Ας θεωρήσουμε παραδείγματος χάριν ένα πρόβλημα P για το οποίο όλοι οι συντελεστές c_j είναι αρνητικοί ή μηδέν και μερικοί συντελεστές b_i αρνητικοί. Η βασική λύση που αντιστοιχεί στη βάση που σχηματίζεται από τις μεταβλητές απόκλισης δεν είναι εφικτή. Αντίθετα για το δυϊκό πρόβλημα ελαχιστοποίησης έχουμε τη διατύπωση:

$$\min \sum_{i=1}^{m} b_i y_i = \max \sum_{i=1}^{m} (-b_i) y_i$$

με τους περιορισμούς:

$$y_i \geq 0 \text{ για } 1 \leq i \leq m$$

$$\sum_{i=1}^{m} (-a_{ij}) y_i \leq -c_j \text{ για} 1 \leq j \leq n$$

Η βασική λύση τώρα είναι εφικτή γιατί όλοι οι συντελεστές $(-c_j)$ είναι θετικοί ή μηδέν. Μπορούμε λοιπόν να επιλύσουμε το δυϊκό πρόβλημα σε μία φάση.

Παράδειγμα

$$\max -x_0$$

με τους περιορισμούς:

$$\begin{aligned} -x_1 + x_0 &\leq 2 \\ x_1 + 2x_0 &\leq -1 \\ x_0, x_1 &\geq 0 \end{aligned}$$

Εισάγοντας τις μεταβλητές απόκλισης οι περιορισμοί γίνονται

$$\begin{aligned} -x_1 + x_0 + x_2 &= 2 \\ x_1 + 2x_0 + x_3 &= -1 \\ x_0, x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Η προφανής βασική λύση $x_2 = 2$ και $x_3 = -1$ δεν είναι εφικτή. Για το δυικό όμως

έχουμε:

$$\left.\begin{array}{rl} \min 2y_1 - y_2 & \\ -y_1 + y_2 & \geq 0 \\ y_1 + 2y_2 & \geq -1 \end{array}\right\} \Leftrightarrow \left.\begin{array}{rl} \max -2y_1 + y_2 & \\ y_1 - y_2 & \leq 0 \\ -y_1 - 2y_2 & \leq 1 \end{array}\right\} \Leftrightarrow \begin{array}{rl} \max -2y_1 + y_2 & \\ y_1 - y_2 + y_3 & = 0 \\ -y_1 - 2y_2 + y_4 & = 1 \end{array}$$

για το οποίο έχουμε τη βασική εφικτή λύση $y_3 = 0$, $y_4 = 1$ και μπορούμε να συνεχίσουμε με τη μέθοδο Simplex. Θα υπενθυμίσουμε πάλι εδώ ότι ο αλγόριθμος της Simplex είναι εκθετικός αλγόριθμος στη χείριστη περίπτωση, αλλά στην πράξη συμπεριφέρεται πολύ ικανοποιητικά. Τα προβλήματα που μελετήσαμε όμως είναι πολυωνυμικά και μπορούν να επιλυθούν με την ελλειψοειδή μέθοδο, η οποία είναι πολυωνυμικός αλγόριθμος, αλλά στην πράξη έχει αρκετά μειονεκτήματα.

3.9 Το πρόβλημα Μεταφοράς

Θα μελετήσουμε τώρα ένα άλλο πρόβλημα με πολλές πρακτικές εφαρμογές. Επίσης θα δούμε πως η δυϊκότητα επιτρέπει να τροποποιήσουμε την Simplex και να σχεδιάσουμε έναν αποδοτικό αλγόριθμο. Ας δούμε πρώτα ένα παράδειγμα.

Παράδειγμα

Μία εταιρεία διαθέτει τρεις αποθήκες και πέντε καταστήματα. Επιθυμεί να τροφοδοτήσει με εμπορεύματα τα καταστήματα από τις αποθήκες έτσι ώστε το κόστος να είναι το ελάχιστο δυνατόν. Οι διαθέσιμες ποσότητες στις αποθήκες είναι αντίστοιχα 11, 12, 7 και οι ανάγκες των καταστημάτων είναι 6, 6, 3, 2, 13.

Ας παρατηρήσουμε ότι οι διαθέσιμες ποσότητες επαρκούν για να ικανοποιήσουν τις ανάγκες των καταστημάτων, αφού: $11 + 12 + 7 = 6 + 6 + 3 + 2 + 13 = 30$.

Γνωρίζουμε ότι το κόστος μεταφοράς μιας μονάδας εμπορεύματος από την αποθήκη i $(i = 1, 2, 3)$ στο κατάστημα j $(j = 1, \ldots, 5)$ είναι c_{ij} που δίδεται στον παρακάτω πίνακα κόστους ανά μεταφερόμενη μονάδα εμπορεύματος:

A \ K	1	2	3	4	5
1	1	1	2	6	3
2	4	3	4	8	8
3	5	6	7	12	10

Το πρόβλημα μπορεί να γραφεί ως ακολούθως.

Έστω x_{ij} η μεταφερόμενη ποσότητα από την αποθήκη i (για $1 \leq i \leq m$, όπου m ο αριθμός των αποθηκών) στο κατάστημα $j(j = 1, \ldots, n)$, όπου n ο αριθμός των καταστημάτων. Η προς ελαχιστοποίηση αντικειμενική συνάρτηση είναι:

$$x_{11}+x_{12}+2x_{13}+6x_{14}+3x_{15}+4x_{21}+3x_{22}+4x_{23}+8x_{24}+8x_{25}+5x_{31}+6x_{32}+7x_{33}+12x_{34}+10x_{35}$$

ή ακόμη $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$, με $m = 3$, $n = 5$ και c_{ij}, όπως περιγράφεται στον πίνακα.

Οι περιορισμοί είναι:

1. Η ποσότητα που φεύγει από την i αποθήκη πρέπει να είναι ίση με τη διαθέσιμη ποσότητα στην αποθήκη αυτή, έστω d_i.

$$\begin{aligned} i=1 \quad & x_{11}+x_{12}+x_{13}+x_{14}+x_{15}=11(=d_1) \\ i=2 \quad & x_{21}+x_{22}+x_{23}+x_{24}+x_{25}=12(=d_2) \\ i=3 \quad & x_{31}+x_{32}+x_{33}+x_{34}+x_{35}=7(=d_3) \end{aligned}$$

2. Η ποσότητα που φθάνει στο κατάστημα j πρέπει να είναι ίση με τις ανάγκες b_j.

$$\begin{aligned} j=1 \quad & x_{11}+x_{21}+x_{31}=6(=b_1) \\ j=2 \quad & x_{12}+x_{22}+x_{32}=6(=b_2) \\ j=3 \quad & x_{13}+x_{23}+x_{33}=3(=b_3) \\ j=4 \quad & x_{14}+x_{24}+x_{34}=2(=b_4) \\ j=5 \quad & x_{15}+x_{25}+x_{35}=13(=b_5) \end{aligned}$$

3. Οι μεταφερόμενες ποσότητες είναι θετικές ή μηδέν.

$$\forall i \, \forall j \;\; x_{ij} \geq 0$$

Αυτό το πρόβλημα είναι ένα πρόβλημα γραμμικού προγραμματισμού μιας ειδικής μορφής, ονομαζόμενο πρόβλημα μεταφοράς.

Ορισμοί και RANK ενός προβλήματος μεταφοράς

Ονομάζουμε πρόβλημα μεταφοράς ένα πρόβλημα γραμμικού προγραμματισμού της παρακάτω μορφής:

$$\begin{aligned} & \min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ & \sum_{j=1}^{n} x_{ij} = d_i \text{ για } i=1,\ldots,m \\ & \sum_{i=1}^{m} x_{ij} = b_j \text{ για } j=1,\ldots,n \\ & \forall i,j, x_{ij} \geq 0 \end{aligned}$$

Ένα πρόβλημα μεταφοράς είναι λοιπόν ένα πρόβλημα με $n \cdot m$ μεταβλητές και με $n+m$ περιορισμούς, αλλά αυτοί οι περιορισμοί δεν είναι ανεξάρτητοι. Πράγματι:

- στο παράδειγμα που δώσαμε, το άθροισμα των τριών πρώτων ισοτήτων είναι ίσο με το άθροισμα των πέντε τελευταίων.

- γενικά, το άθροισμα των m πρώτων ισοτήτων υποδηλώνει ότι το άθροισμα των μεταφερόμενων ποσοτήτων $\sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}$ ισούται με το άθροισμα των διαθέσιμων ποσοτήτων $\sum_{i=1}^{m}d_i$, στις αποθήκες.

  Επίσης το άθροισμα των n τελευταίων ισοτήτων υποδηλώνει ότι το άθροισμα των μεταφερόμενων ποσοτήτων $\sum_{i=1}^{m}\sum_{j=1}^{n}x_{ij}$ ισούται με το άθροισμα των αναγκών, στα εργοστάσια.

- Το σύστημα δεν επιδέχεται λύσεις παρά μόνο αν: $\sum_{i=1}^{m}d_i = \sum_{j=1}^{n}\sum_{i=1}^{m}x_{ij}$ το οποίο ισούται με το άθροισμα των αναγκών $\sum_{j=1}^{n}b_j$ και σε αυτή την περίπτωση υπάρχει μια σχέση ανάμεσα στις $m+n$ ισότητες. Μια ισότητα μπορεί να προκύψει σαν γραμμικός συνδυασμός των υπολοίπων.

  Για να επιλύσουμε το πρόβλημα θα απαλείψουμε λοιπόν μια εξίσωση (π.χ. την τελευταία). Τότε έχουμε ένα πρόβλημα με $m+n-1$ περιορισμούς οι οποίοι είναι ανεξάρτητοι.

- Αν $\sum_{i=1}^{m}d_i \neq \sum_{j=1}^{n}b_j$, το σύστημα είναι αδύνατο.

  Μπορούμε παρ΄ όλα αυτά να αναζητήσουμε μία βέλτιστη λύση προσθέτοντας μια αποθήκη $m+1$ ή ένα φανταστικό κατάστημα $n+1$ έτσι ώστε να αποκαταστήσουμε την ισότητα διαθέσιμων ποσοτήτων και αναγκών. Επιπλέον, τα αντίστοιχα κόστη θα προβλέψουμε να είναι $c_{m+1j} = +\infty, \forall j = 1, 2, \ldots, n$ ή $c_{in+1} = +\infty, \forall i = 1, 2, \ldots, m$.

  Αλλά σε αυτή την περίπτωση, θα εξασφαλίσουμε μια μεταφορά με ελάχιστο κόστος η οποία θα μας οδηγήσει να αφήσουμε αποθέματα στις αποθήκες ή να μην ικανοποιήσουμε πλήρως τα καταστήματα.

Μορφή του πίνακα ενός προβλήματος μεταφοράς

Ο πίνακας A των συντελεστών των αγνώστων ενός προβλήματος μεταφοράς είναι μιας πολύ ειδικής μορφής. Όλα τα στοιχεία του είναι είτε 1 είτε 0. Τα μόνα μη μηδενικά στοιχεία μιας κολόνας που αντιστοιχεί στη μεταβλητή $x_{k\ell}$ (κολόνα $(k-1)n+\ell$) είναι αυτά που ευρίσκονται(βλέπε πίνακα παρακάτω):

- στη γραμμή k (περιορισμός $\sum_{j=1}^{n} x_{ij} = d_k$)
- στη γραμμή $m+\ell$ (περιορισμός $\sum_{i=1}^{m} x_{i\ell} = b_\ell$)

Κάθε κολόνα περιέχει λοιπόν δύο στοιχεία ίσα με 1, εκτός από τις κολόνες που αντιστοιχούν στις μεταβλητές x_{in} οι οποίες δεν περιέχουν παρά μόνο ένα 1, δεδομένου της κατάργησης της τελευταίας γραμμής.

Μπορούμε να γράψουμε κάθε κολόνα $x_{kl} = e_k + e_{m+l}$, όπου e_i είναι τα διανύσματα της βάσης.

Παράδειγμα

Ο πίνακας A του παραδείγματος αναφοράς είναι:

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
1	1	1	1	1										
					1	1	1	1	1					
										1	1	1	1	1
1					1					1				
	1					1					1			
		1					1					1		
			1					1					1	

Δεδομένου ότι το πρόβλημα μεταφοράς είναι ένα πρόβλημα γραμμικού προγραμματισμού, θα χρησιμοποιήσουμε τη μέθοδο Simplex για να το επιλύσουμε, αλλά σε μια απλοποιημένη μορφή λαμβάνοντας υπ' όψιν την ειδική μορφή του πίνακα.

Θα μελετήσουμε στη συνέχεια το χαρακτηρισμό μιας βάσης, την αναζήτηση μιας βασικής λύσης εκκίνησης, τον υπολογισμό των κριτηρίων υποψηφιότητας εισαγωγής και την αλλαγή της βάσης.

Βασική λύση - Δέντρο

Ο πίνακας A είναι τάξης $m+n-1$ αφού διαγράψουμε μια ισότητα η οποία προκύπτει ως γραμμικός συνδυασμός των υπολοίπων[2]. Μια βάση αποτελείται λοιπόν από $m+n-1$ δείκτες (μεταβλητές). Αυτό σημαίνει ότι, από τις $m \cdot n$ δυνατές συνδέσεις ανάμεσα στις m αποθήκες και τα n καταστήματα, μόνο $m+n-1$ θα χρησιμοποιηθούν πραγματικά για να επιτευχθεί μία λύση ελαχίστου κόστους μεταφοράς.

Σε κάθε λύση του προβλήματος μεταφοράς, μπορούμε να αντιστοιχήσουμε έναν γράφο που ορίζεται ως εξής:

Οι κόμβοι του γράφου είναι οι m αποθήκες $A_1, A_2, \ldots, A_m$ και τα n καταστήματα $K_1, K_2, \ldots, K_n$ και αν $x_{ij} \neq 0$ υπάρχει μία πλευρά από την αποθήκη i στο κατάστημα j.

Μια βασική λύση αντιστοιχεί λοιπόν σε ένα γράφο με $n+m-1$ πλευρές.

Θεώρημα 3.9.1

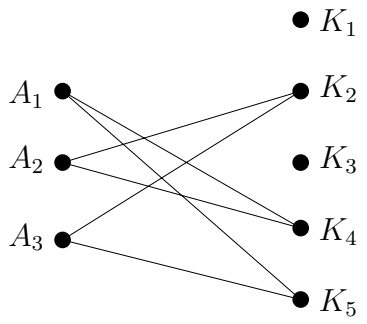
Για να σχηματίζει ένα σύνολο από $m+n-1$ μεταβλητές, βάση σ΄ ένα πρόβλημα μεταφοράς, πρέπει και αρκεί ο αντίστοιχος γράφος να είναι δέντρο δηλαδή ένας γράφος με $m+n$ κόμβους, $m+n-1$ πλευρές και χωρίς κύκλο.

Απόδειξη

Ας αποδείξουμε ότι η συνθήκη είναι αναγκαία. Ας θεωρήσουμε το παράδειγμά μας που ήδη περιγράψαμε, τον πίνακα A και τον κύκλο $(A_1, K_4, A_2, K_2, A_3, K_5, A_1)$

Αυτό σημαίνει ότι οι μεταβλητές $x_{14}, x_{24}, x_{22}, x_{32}, x_{35}, x_{15}$ είναι μέσα στη βάση. Οι αντίστοιχες κολώνες στον πίνακα των συντελεστών Α είναι οι εξής:

[2]Παραδείγματος χάριν πρόσθεσε τις 3 πρώτες εξισώσεις του παραδείγματος και αφαίρεσε το άθροισμα των τεσσάρων υπολοίπων οπότε έχουμε την πέμπτη εξίσωση



Σχήμα 3.11: Μιά λύση που αντιστοιχεί σε κύκλο δεν μπορεί να είναι βασική εφικτή λύση

x_{14}	x_{24}	x_{22}	x_{32}	x_{35}	x_{15}
1	0	0	0	0	1
0	1	1	0	0	0
0	0	0	1	1	0
0	0	0	0	0	0
0	0	1	1	0	0
0	0	0	0	0	0
1	1	0	0	0	0

Αν συμβολίσουμε με $k(x_{ij})$ την κολώνα του πίνακα Α που αντιστοιχεί στην μεταβλητή x_{ij} τότε ο γραμμικός συνδυασμός:

$$k(x_{14}) - k(x_{24}) + k(x_{22}) - k(x_{32}) + k(x_{35}) - k(x_{15})$$

δίνει το μηδενικό διάνυσμα. Δηλαδή οι μεταβλητές $x_{14}, x_{24}, x_{22}, x_{32}, x_{35}, x_{15}$ δεν μπορούν να ανήκουν σε μια βάση (ο αντιστοιχός πίνακας B θα πρέπει να είναι αντιστρέψιμος). Στη γενική περίπτωση η επιχειρηματολογία είναι ακριβώς η ίδια (άσκηση).

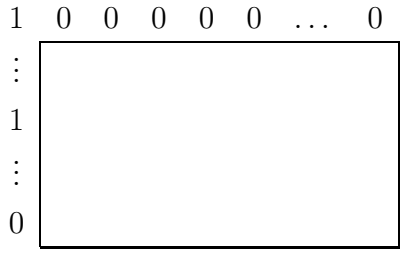
Αντίστροφα: Ας δείξουμε ότι η συνθήκη είναι ικανή.

Ας υποθέσουμε ότι έχουμε ένα δέντρο με $n + m$ κόμβους και ας δείξουμε ότι οι αντίστοιχες μεταβλητές σχηματίζουν μία βάση. Ας θυμηθούμε κατ΄ αρχάς ότι ένα δέντρο περιέχει το λιγότερο ένα κόμβο βαθμού 1.

Επομένως ο γράφος περιέχει το λιγότερο ένα κόμβο απ΄ όπου δεν φεύγει παρά μόνο μία πλευρά. Αντιμεταθέτοντας τα νούμερα των αποθηκών μεταξύ τους, και ενδεχομένως

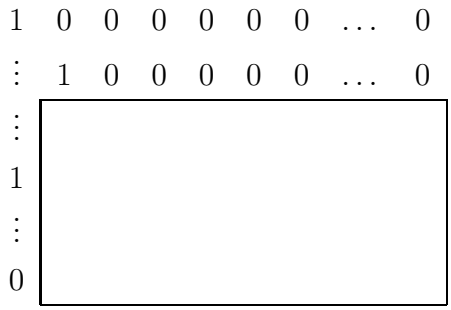
τα καταστήματα με τις αποθήκες, μπορούμε να υποθέσουμε ότι ο κόμβος βαθμού ένα είναι η αποθήκη με το νούμερο ένα.

Ο Πίνακας B μπορεί να γραφεί τότε ως εξής:



Τα μηδενικά της πρώτης γραμμής δηλώνουν το γεγονός ότι δεν υπάρχει καμία άλλη μεταβλητή x_{ij} παρά μόνο αυτή της πρώτης κολόνας μέσα στη βάση (αφού από την αποθήκη 1 δεν φεύγει παρά μόνο μία πλευρά).

Διαγράφοντας την αποθήκη 1 και την αντίστοιχη πλευρά, μένει ένας γράφος με $n+m-2$ πλευρές και χωρίς κύκλο. Μπορούμε λοιπόν να εφαρμόσουμε την ίδια λογική και με τη βοήθεια αλλαγών στην απαρίθμηση, να επιτύχουμε τη μορφή:

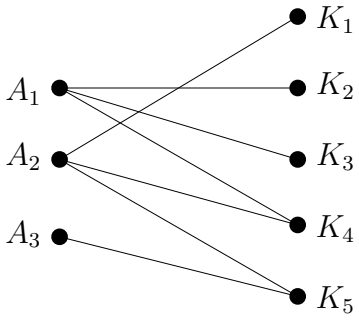


Με διαδοχικές επαναλήψεις μπορούμε να δώσουμε στον πίνακα B την τριγωνική μορφή. Η ορίζουσα του B, ισούται με το γινόμενο των στοιχείων της διαγωνίου, και επομένως είναι ίση με ένα. Ο πίνακας B είναι λοιπόν κανονικός και συνιστά μία βάση.

Πλέγμα μεταφορών

Είδαμε ότι σε κάθε βασική λύση, μπορούμε να αντιστοιχίσουμε ένα δέντρο. Μπορούμε επίσης να παρουσιάσουμε μία βάση με τη βοήθεια ενός πλέγματος μεταφοράς: το πλέγμα είναι ένας πίνακας με m γραμμές και n κολόνες όπου παρουσιάζονται οι τιμές των x_{ij}.

Παράδειγμα



	1	2	3	4	5	
1		6	3	2		11
2	6			0	6	12
3					7	7
	6	6	3	2	13	

Σχήμα 3.12: Μια βασική λύση υπό μορφή δένδρου πάνω και το αντίστοιχο πλέγμα μεταφοράς κάτω. Οι γραμμές αντιστοιχούν στις αποθήκες και οι κολόνες στα καταστήματα. Οι τιμές εκτός του πλέγματος δεξιά είναι οι διαθέσιμες στις αποθήκες ποσότητες ενώ κάτω φαίνονται οι αιτούμενες από τα καταστήματα ποσότητες.

Το άθροισμα των τιμών κάθε κολόνας στο πλέγμα του σχήματος 3.12 είναι ίσο με τις ανάγκες του αντίστοιχου καταστήματος και το άθροισμα των τιμών κάθε γραμμής είναι ίσο με τη διαθέσιμη ποσότητα της αντίστοιχης αποθήκης. Ας παρατηρήσουμε όπως συμβαίνει στο παράδειγμα μας στο πλέγμα, μια μεταβλητή (εδώ η x_{24}) μπορεί να υπάρχει μέσα στη βάση με μηδενική τιμή. Λέμε τότε ότι έχουμε το φαινόμενο του εκφυλισμού, το οποίο θα συζητήσουμε αργότερα.

Αρχική λύση: Μέθοδος της Βόρειο-Δυτικής γωνίας

Όπως κάναμε με τη μέθοδο Simplex χρειαζόμαστε σαν αρχική λύση μία εφικτή βάση. Η μέθοδος της ΒΔ γωνίας μας επιτρέπει να βρούμε εύκολα μια εφικτή λύση ως εξής:

- Αρχίζουμε από την πάνω αριστερά γωνία του πλέγματος δηλαδή την κυψέλη ($i = 1$, $j = 1$) στην οποία δίνουμε την τιμή:

$$x_{11} = \min(d_1, b_1)$$

όπου d_1 είναι η διαθέσιμη ποσότητα στην αποθήκη 1 και b_1 η αιτούμενη ποσότητα από το κατάστημα 1. Αυτό οδηγεί στον κορεσμό, είτε της γραμμής 1, είτε της κολόνας 1.

- Διαγράφουμε τότε είτε την κορεσμένη γραμμή, είτε την κορεσμένη κολόνα.

- Επαναλαμβάνουμε τη διαδικασία στο μειωμένο πλέγμα.

Ας δούμε τη μέθοδο στο προηγούμενο παράδειγμα.

Παράδειγμα

1ο βήμα

6	5				11
					12
					7
6	6	3	2	13	

$x_{11} = 6$ (κορεσμός και διαγραφή κολόνας 1), $b_1 = 0$, $d_1 = 5$

$x_{12} = 5$ (κορεσμός και διαγραφή γραμμής 1), $b_1 = 1$, $d_1 = 0$

2ο βήμα

6	5				11
	1	3	2	6	12
					7
6	6	3	2	13	

$x_{22} = 1$ (κορεσμός κολόνας 2), $d_2 = 11$, $b_2 = 0$

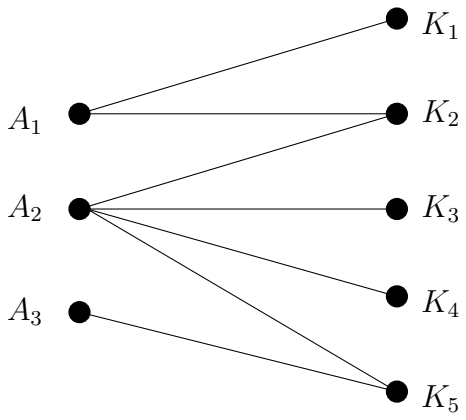
$x_{23} = 3$ (κορεσμός κολόνας 3), $d_2 = 8$, $b_3 = 0$

$x_{24} = 2$ (κορεσμός κολόνας 4), $d_2 = 6$, $b_4 = 0$

$x_{25} = 6$ (κορεσμός γραμμής 2), $d_2 = 0$, $b_5 = 7$

6	5				11
	1	3	2	6	12
				7	7
6	6	3	2	13	

$x_{35} = 7$ (κορεσμός γραμμής 3 και κολόνας 5), $d_3 = 0$, $b_5 = 0$



Σχήμα 3.13: Μια βασική εφικτή λύση που προκύπτει από τη μέθοδο της Βορειοδυτικής Γωνίας και το αντίστοιχο δέντρο.

Θεώρημα 3.9.2

Η μέθοδος της ΒΔ-γωνίας δίνει:

1. *μία εφικτή λύση αφού σε κάθε βήμα λαμβάνουμε υπ' όψιν τους περιορισμούς.*
2. *μία βασική λύση (αντιστοιχεί σε ένα γράφο με $m + n - 1$ πλευρές και χωρίς κύκλο).*

δηλαδή έχουμε μία βασική εφικτή λύση.

Απόδειξη

Για το (1) η απόδειξη είναι προφανής. Για το (2) ας παρατηρήσουμε ότι φεύγοντας από τη μεταβλητή x_{11}, μετατοπιζόμαστε σε κάθε βήμα είτε κατά μια κολόνα, είτε κατά μία γραμμή. Για να φτάσουμε στην κυψέλη (m, n), πραγματοποιούμε λοιπόν $m - 1$ κάθετες μετατοπίσεις και $n - 1$ οριζόντιες. Δηλαδή συνολικά $m + n - 2$ μεταβλητές εκτός από την x_{11}. Επομένως συνολικά $m + n - 1$ μεταβλητές, οι οποίες αντιστοιχούν σε $m + n - 1$ πλευρές του αντίστοιχου γράφου.

Επίσης σε κάθε καταχώρηση τιμής μιας μεταβλητής, ο ένας από τους δύο δείκτες i και j της μεταβλητής x_{ij} που επιλέγουμε αυξάνει. Δεν μπορεί λοιπόν να υπάρχει κύκλος. Επομένως η μέθοδος κατασκευάζει μία βασική λύση η οποία ικανοποιεί τους περιορισμούς και άρα είναι μία βασική εφικτή λύση.

Υπολογισμός των ''reduced costs''

Μόλις είδαμε ότι είναι εύκολο να βρούμε μια αρχική βασική εφικτή λύση. Το επόμενο βήμα, για να επιλύσουμε το πρόβλημά μας με τη μέθοδο Simplex είναι η επιλογή ενός δείκτη r ο οποίος θα επιτρέψει να περάσουμε από μια βάση σε μια νέα γειτονική βάση.

Στην περίπτωση μεγιστοποίησης επιλέγουμε ένα δείκτη r έτσι ώστε $c_r > ya_r$ (δες μέθοδο Simplex).

Στο πρόβλημα μεταφοράς, έχουμε την ελαχιστοποίηση της αντικειμενικής συνάρτησης. Στην περίπτωση αυτή επιλέγουμε το δείκτη r έτσι ώστε:

$$c_r < ya_r$$

Ο δείκτης r θα είναι παρακάτω για το πρόβλημά μας ένα ζευγάρι δεικτών ij.

Ορισμός

Στην περίπτωση του προβλήματος μεταφοράς, η αντικειμενική συνάρτηση αντιπροσωπεύει κόστος και οι συνιστώσες των διανυσμάτων C_B και $C_N - yA_N$ που συνιστούν το κριτήριο υποψηφιότητας εισαγωγής (βλ. αλγόριθμο Simplex) ονομάζονται reduced costs. Για τις μεταβλητές εντός βάσης τα reduced costs είναι μηδέν.

Τα reduced costs στο πρόβλημα μεταφοράς μπορούν να γραφούν:

$$d_{ij} = c_{ij} - y_i - y_{m+j} \text{ με } y_{m+n} = 0$$

δεδομένου ότι:

- Τα κόστη στην αντικειμενική συνάρτηση εδώ είναι c_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$.
- Οι μεταβλητές είναι x_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$.
- Για $j < n$, μόνο δύο στοιχεία στην κολόνα της μεταβλητής x_{ij} στον πίνακα A είναι μη μηδενικά:
  - το στοιχείο της γραμμής i
  - το στοιχείο της γραμμής $m + j$

- Για $j = n$, μόνο το στοιχείο της γραμμής i είναι μη μηδενικό (δες τον πίνακα του παραδείγματος).
- Για το διάνυσμα y έχουμε: $y = C_B B^{-1}$ (δες Simplex). Άρα $yB = C_B$.
- B^{-1} είναι ένας πίνακας $(m + n - 1) \times (m + n - 1)$.
- C_B είναι ένα διάνυσμα γραμμή με $m + n - 1$ συνιστώσες.
- y είναι ένα διάνυσμα γραμμή με $m + n - 1$ συνιστώσες.

Υπολογισμός του διανύσματος y

Ας δούμε τώρα πως μπορούμε εύκολα να υπολογίσουμε το διάνυσμα $y = C_B B^{-1}$.

Για τις μεταβλητές που ανήκουν στη βάση, γνωρίζουμε ότι οι αντίστοιχες συνιστώσες, στο διάνυσμα κριτηρίου υποψηφιότητας είναι μηδέν. Επομένως

$$\boxed{\forall (i,j) \in B \text{ έχουμε: } y_i + y_{m+j} = c_{ij}}$$

Έχουμε έτσι $m + n - 1$ εξισώσεις ανεξάρτητες με $m + n$ αγνώστους. Θέτοντας $y_{m+n} = 0$, έχουμε τη δυνατότητα να υπολογίσουμε τις συνιστώσες του διανύσματος y με μοναδικό τρόπο. Στη συνέχεια μπορούμε να υπολογίσουμε, για τις μεταβλητές εκτός βάσης τα reduced costs d_{ij} ως εξής:

$$\boxed{\forall (i,j) \notin B \text{ έχουμε: } d_{ij} = c_{ij} - y_i - y_{m+j}}$$

Τα reduced costs d_{ij} είναι στο πλήθος $m \cdot n$. Μπορούμε να χρησιμοποιήσουμε για τον υπολογισμό τους ένα πλέγμα με m γραμμές και n κολόνες. Τα d_{ij} που αντιστοιχούν στις μεταβλητές της βάσης είναι μηδέν και είναι άσκοπο να τις εμφανίσουμε στο πλέγμα. Στις αντίστοιχες θέσεις θα έχουμε μόνο τις τιμές των c_{ij} που αντιστοιχούν στις βασικές μεταβλητές. Το διάνυσμα y θα γράφεται υπό τη μορφή μίας κολόνας $(v_i), i = 1, \ldots, m$ και μιας γραμμής $(v_{m+j}), j = 1, \ldots, n$. Θα δούμε ένα παράδειγμα στη βασική λύση της ΒΔ γωνίας.

Παράδειγμα

Η βασική λύση που προέκυψε από την ΒΔ γωνία είναι x_{11}, x_{12}, x_{22}, x_{23}, x_{24}, x_{25}, x_{35}. Στο πλέγμα στις θέσεις των μεταβλητών αυτών φαίνονται τα αντίστοιχα κόστη c_{11},c_{12},c_{22},c_{23},c_{24},c_{25} και c_{35}. Στις μεταβλητές εκτός βάσης έχουμε στο πάνω μέρος το κόστος c_{ij} και στο κάτω μέρος το reduced cost d_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$, $m = 3$ και $n = 5$.

c_{ij} / d_{ij}					
	1	1	2 / 0	6 / 0	3 / -3	6
	4 / 1	3	4	8	8	8
	5 / 0	6 / 1	7 / 1	12 / 2	10	10
	-5	-5	-4	0	0	

Δεξιά του πλέγματος έχουμε το διάνυσμα $(v_i)_{1\leq i\leq m}$ και κάτω από το πλέγμα έχουμε το διάνυσμα $(v_{m+j})_{1\leq i\leq n}$ τα οποία υπολογίζονται ως εξής:

$$
\begin{aligned}
&c_{35} = 10 = v_{m+n} + v_3 \Rightarrow v_3 = 10, \text{ θέτοντας } (u_{m+n} = 0)\\
&c_{25} = 8 = v_{m+n} + v_2 \Rightarrow v_2 = 8\\
&c_{24} = 8 = v_{m+4} + v_2 \text{ με } v_2 = 8 \Rightarrow v_{m+4} = 0 \text{ κλπ.}\\
&\ldots\ldots\ldots
\end{aligned}
$$

Μετά υπολογίζουμε τα d_{ij}. Π.χ. $d_{15} = 3 - v_1 - v_8 = 3 - 6 - 0 = -3$, $d_{14} = 0$, $d_{13} = 0$,...

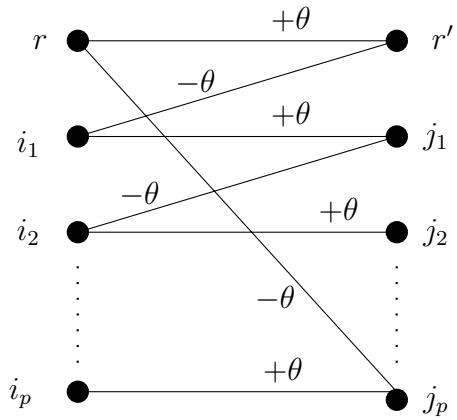
Παρατηρούμε ότι η μεταβλητή x_{15} έχει reduced cost αρνητικό. Δεν είμαστε επομένως στο βέλτιστο και αυτή η μεταβλητή θα πρέπει να εισαχθεί στη βάση.

Αλλαγή βάσης

Το επόμενο βήμα στον αλγόριθμο της Simplex είναι τώρα να καθορίσουμε τη μεταβλητή η οποία θα εξέλθει από τη βάση όταν κάποια μεταβλητή $x_{rr'}$ εισέρχεται στη βάση. Θα πρέπει να κάνουμε την επιλογή ώστε αφ΄ ενός μεν η λύση μας να είναι εφικτή, αφ΄ ετέρου δε να είναι βασική λύση.

Ας θεωρήσουμε το γράφο που αντιστοιχεί στη βάση B. Ο γράφος αυτός βέβαια είναι όπως ήδη είπαμε ένα δέντρο. Αν του προσθέσουμε την πλευρά rr' δημιουργούμε ένα και μόνο κύκλο.

Έστω $(r, r', i_1, j_1, i_2, j_2, \ldots, j_p, r)$ αυτός ο κύκλος. Ας παρατηρήσουμε ότι έχουμε ένα ζυγό αριθμό πλευρών.



Όταν $x_{rr'}$ εισέρχεται στη βάση και παίρνει την τιμή $\theta > 0$, το κατάστημα r' δέχεται ένα πλεόνασμα θ σε σχέση με την ανάγκη του. Πρέπει λοιπόν να ελαττώσουμε την $x_{i_1 r'}$ κατά θ. Αυτή η ελάττωση του $x_{i_1 r'}$ οδηγεί την αποθήκη i_1 σε ένα πλεόνασμα κατά θ μονάδες.

Για να διατηρήσουμε την ισορροπία, οφείλουμε να αυξήσουμε κατά θ τη μεταβλητή $x_{i_1 j_1}$ το οποίο δημιουργεί ένα πλεόνασμα κατά θ μονάδες στο j_1 και οδηγεί στην ελάττωση του $x_{i_2 j_1}$ κατά θ κ.λπ.

Για να είναι η λύση που βρίσκουμε εφικτή, θα πρέπει όλες οι νέες τιμές των μεταβλητών να μένουν θετικές ή μηδέν.

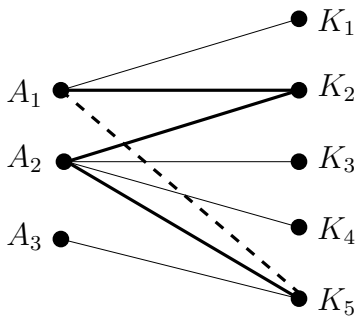
Το πρόβλημα βέβαια τίθεται μόνο για τις μεταβλητές απ' όπου αφαιρούμε θ. Αν διαλέξουμε θ έτσι ώστε:

$$\theta = \min\{x_{i_1 r'}, x_{i_2 j_1}, x_{i_k j_{k-1}}, \ldots, x_{r j_p}\}$$

μία το λιγότερο από τις μεταβλητές που εμπλέκεται στον κύκλο, μηδενίζεται. Διαλέγοντας αυτήν την μεταβλητή για να εξέλθει από τη βάση, διασπάμε τον κύκλο που υπάρχει και έχουμε μια νέα εφικτή βάση.

Παράδειγμα 1:

6	5 - θ			$+\theta$
	$1+\theta$	3	2	$6-\theta$
				7



Η είσοδος της x_{15} προσθέτει στο γράφο την πλευρά που είναι διακεκομμένη. Δημιουργείται έτσι ο κύκλος x_{15}, x_{12}, x_{22}, x_{25}, x_{15}. Ο κύκλος αυτός φαίνεται επίσης στο πλέγμα, όπου στη θέση $(1,5)$ έχει εισαχθεί η ποσότητα $+\theta$. Στη θέση $(1,2)$ έχει αφαιρεθεί θ, στη θέση $(2,2)$ έχει προστεθεί θ και στη θέση $(2,5)$ έχει αφαιρεθεί θ. Το θ επιλέγεται έτσι ώστε $\theta = \min\{x_{12}, x_{25}\} = 5$. Ο κύκλος διασπάται εξάγοντας από τη βάση τη μεταβλητή x_{12}.

Συνοπτικά ο αλγόριθμος έχει ως εξής:

1. Εύρεση μιας βασικής εφικτής λύσης με τη μέθοδο της ΒΔ γωνίας.
2. Υπολόγισε το διάνυσμα $y = (y_1, \ldots, y_m, y_{m+1}, \ldots, y_{m+n})$ χρησιμοποιώντας τα κόστη της βασικής εφικτής λύσης $y_i + y_{m+j} = c_{ij}$, $\forall\,(i,j) \in B$
3. Υπολόγισε τα reduced costs για τις μεταβλητές εκτός βάσης $d_{ij} = c_{ij} - y_i - y_{m+j}$, $\forall\,(i,j) \notin B$
4. Αν $d_{ij} \geq 0$, $\forall\,(i,j) \notin B$ τότε B είναι βέλτιστη λύση. Διαφορετικά επίλεξε $d_{ij} < 0$ και εισήγαγε x_{ij} στη βάση με τιμή $+\theta$.
5. Διέσπασε τον κύκλο θέτοντας θ ίσο με την τιμή της μικρότερης μεταβλητής απ' όπου αφαιρείται το θ (η μεταβλητή εξέρχεται από τη βάση) και επανέλαβε από το βήμα 2.

Ας δούμε τώρα συνολικά το παράδειγμά μας:

Αρχικά φαίνεται ο πίνακας με τα μοναδιαία κόστη. Στο δεύτερο πίνακα βλέπουμε την πρώτη βασική εφικτή λύση και την πρώτη επανάληψη του αλγορίθμου. Ακολουθούν δύο ακόμη επαναλήψεις. Στην τρίτη επανάληψη τα reduced costs είναι όλα μη αρνητικά.

1	1	2	6	3
4	3	4	8	8
5	6	7	12	10

1	1	2 / 0	6 / 0	3 / -3	6	6	5-θ			+θ
4 / 1	3	4	8	8	8		1+θ	3	2	6-θ
5 / 0	6 / 1	7 / 1	12 / 2	10	10					7

$\theta = 5$

-5	-5	-4	0	0

1	1 / 3	2 / 3	6 / 3	3	3	6-θ				5+θ
4 / -2	3	4	8	8	8		6	3	2	1
5 / -3	6 / 1	7 / 1	12 / 2	10	10	+θ				7-θ

$\theta = 6$

-2	-5	-4	0	0

1	1 / 3	2 / 3	6 / 3	3	3					11
4	3	4	8	8	8		6	3	2	1
5	6 / 1	7 / 1	12 / 2	10	10	6				1

-5	-5	-4	0	0

Βέλτιστη λύση: $x^*_{15} = 11, x^*_{22} = 6, x^*_{23} = 3, x^*_{24} = 2, x^*_{25} = 1, x^*_{31} = 6, x^*_{35} = 1$

Παράδειγμα 2

Δίδεται το πρόβλημα μεταφοράς που περιγράφεται από τον ακόλουθο πίνακα (κόστη c_{ij}, διαθέσιμες ποσότητες a_i και ζητούμενες ποσότητες b_j). Να γραφεί το αντίστοιχο γραμμικό πρόβλημα και να επιλυθεί.

		Πελάτες					Διαθέσιμες
		1	2	3	4	5	Ποσότητες
Αποθήκες	I	5	6	4	8	10	80
	II	7	9	10	5	6	50
	III	8	3	6	2	4	70
Ζητούμενες Ποσότητες		40	20	60	30	50	200

Το Γραμμικό Πρόγραμμα έχει ως εξής:

$$\min 5x_{11}+6x_{12}+4x_{13}+8x_{14}+10x_{15}+7x_{21}+9x_{22}+10x_{23}+5x_{24}+6x_{25}+8x_{31}+3x_{32}+6x_{33}+2x_{34}+4x_{35}$$

$$\begin{aligned}
x_{11} + x_{12} + x_{13} + x_{14} + x_{15} &= 80\\
x_{21} + x_{22} + x_{23} + x_{24} + x_{25} &= 50\\
x_{31} + x_{32} + x_{33} + x_{34} + x_{35} &= 70\\
x_{11} + x_{21} + x_{31} &= 40\\
x_{12} + x_{22} + x_{32} &= 20\\
x_{13} + x_{23} + x_{33} &= 60\\
x_{14} + x_{24} + x_{34} &= 30\\
x_{15} + x_{25} + x_{35} &= 50
\end{aligned}$$

Παρακάτω (σχήμα 3.14) φαίνεται η λύση που προκύπτει από τη μέθοδο της ΒΔ γωνίας. Η αντικειμενική συνάρτηση έχει την τιμή 1090.

	1	2	3	4	5					
I	40	20	20			~~80~~	~~40~~	~~20~~	0	
II			40	10		~~50~~	~~10~~	0		$z = 1090$
III				20	50	~~70~~	~~50~~	0		
			~~60~~	~~30~~						
	~~40~~	~~20~~								
			~~40~~	~~20~~	~~50~~					
	0	0	0	0	0					

Σχήμα 3.14: Μία λύση με τη μέθοδο της ΒΔ γωνίας

3.10 Μέθοδος Vogel (ή μέγιστης διαφοράς)

Θα δούμε τώρα μία άλλη μία μέθοδο η οποία επιτρέπει να βρούμε μια εφικτή λύση η οποία αρκετές φορές είναι πολύ καλή και πολλές φορές μάλιστα βέλτιστη.

Αλγόριθμος

1. Θεώρησε τον πίνακα με τα κόστη, συμπεριλαμβάνοντας τις διαθέσιμες και ζητούμενες ποσότητες.

2. Υπολόγισε τη διαφορά ανάμεσα στα δύο πιο μικρά κόστη για κάθε γραμμή και κάθε κολόνα. Βρίσκουμε έτσι m διαφορές για τις γραμμές και n διαφορές για τις κολόνες.

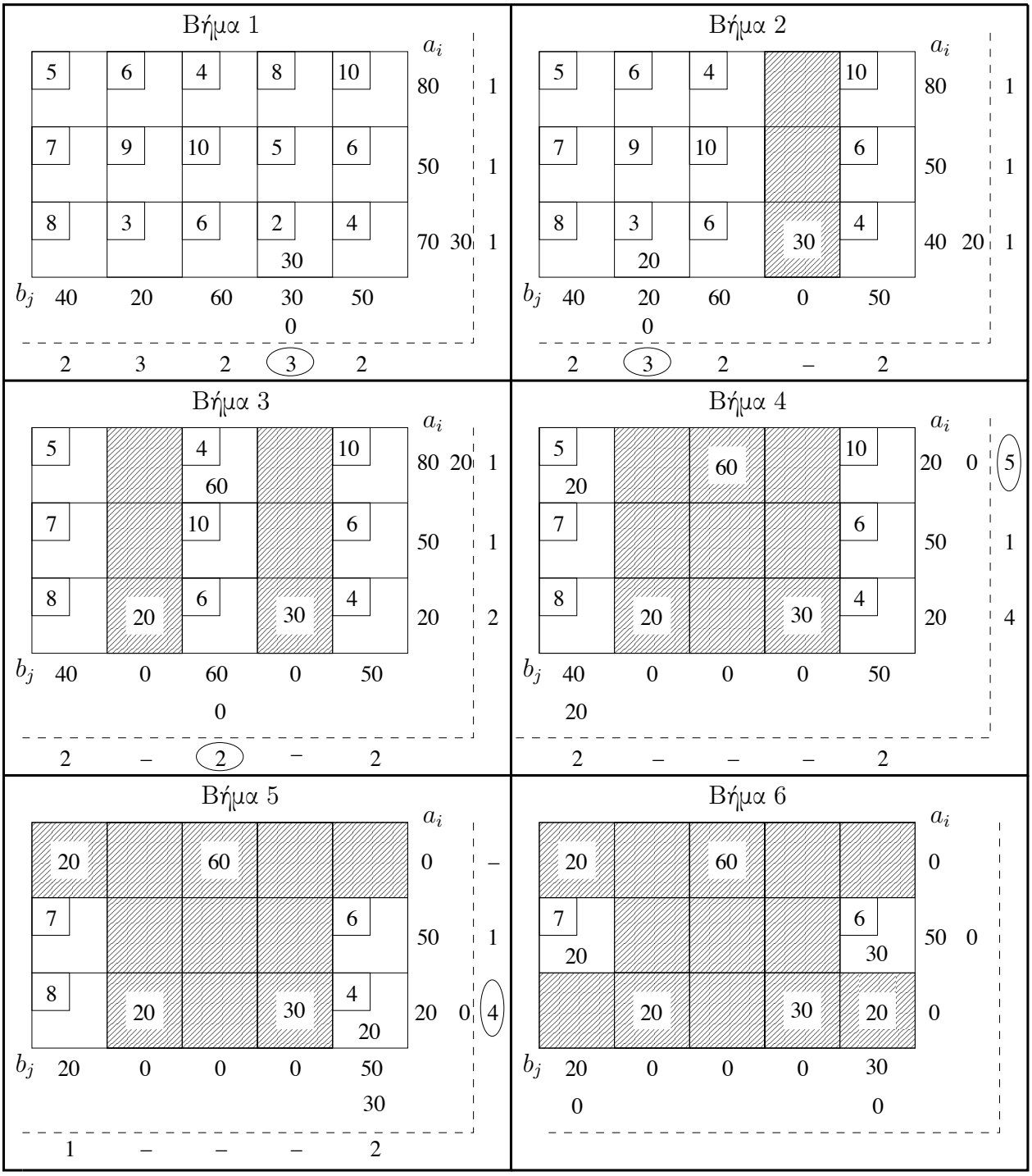
3. Διάλεξε τη γραμμή ή την κολόνα η οποία έχει τη μεγαλύτερη διαφορά. Αν υπάρχουν ισοδύναμες επιλογές, τότε η επιλογή είναι αυθαίρετη.

4. Καταχώρησε την μεγαλύτερη δυνατή ποσότητα (σεβόμενος τους περιορισμούς) στην κυψέλη που έχει το πιο μικρό κόστος της γραμμής ή της κολόνας που βρήκαμε στο βήμα 3.

5. Διέγραψε την γραμμή ή την κολόνα που είναι κορεσμένη.

6. Επέστρεψε στο βήμα 2, με τον εναπομείναντα πίνακα. Η διαδικασία τελειώνει όταν όλες οι γραμμές και οι κολόνες είναι κορεσμένες.

Παράδειγμα 2 (μέθοδος Vogel)



Σχήμα 3.15: Η εκτέλεση της μεθόδου Vogel στο Παράδειγμα 2

Η λύση με τη μέθοδο Vogel είναι (δες τον πίνακα εκτέλεσης)

$$x_{11} = 20, x_{13} = 60, x_{21} = 20, x_{25} = 30, x_{32} = 20, x_{34} = 30, x_{35} = 20$$

η οποία έχει κόστος:

$$z^* = 20 \times 5 \ + 60 \times 4 + 20 \times 7 + 30 \times 6 + 20 \times 3 + 30 \times 2 + 20 \times 4$$

δηλ. $z^* = 860$.

Η λύση αυτή είναι κατά πολύ καλύτερη από αυτήν της ΒΔ-γωνίας. Μπορούμε εύκολα να διαπιστώσουμε επίσης ότι είναι και βέλτιστη λύση.

Ήδη στην αρχή αυτού του κεφαλαίου είδαμε ένα παράδειγμα όπου η μέθοδος της ΒΔ γωνίας, έδινε μία αρχική λύση εκφυλισμένη, δηλαδή η λύση περιέχει τουλάχιστον μια βασική μεταβλητή ίση με μηδέν.

Παρατηρούμε ότι η λύση περιέχει μόνο 6 μεταβλητές θετικές. Επίσης, διαπιστώνουμε, ότι ο υπολογισμός των $v_i, i = 1, \ldots, m$ και $v_{m+j}, j = 1, \ldots, n$ δεν είναι προφανής. Για να ξεπεράσουμε αυτή τη δυσκολία, μπορούμε να εισάγουμε στην κυψέλη (3,4) με τη μηδενική μεταβλητή, μία απειροελάχιστη διαταραχή $\epsilon > 0$. Έτσι μπορούμε να υπολογίσουμε τα v_i, v_{m+j} συναρτήσει του ϵ και να συνεχίσουμε κανονικά τη μέθοδο. Δεν έχουμε παρά να θέσουμε στο τέλος $\epsilon = 0$ στη βέλτιστη λύση.

Παράδειγμα 3

Ας θεωρήσουμε το πρόβλημα μεταφοράς:

	K_1	K_2	K_3	d_i
A_1	5	1	1	4
A_2	2	6	9	6
b_j	3	5	2	

Το πρόβλημα γράφεται σαν ένα πρόβλημα γραμμικού προγραμματισμού ως εξής:

$$x_{11} + x_{21} \geq 3 \tag{3.1}$$
$$x_{12} + x_{22} \geq 5 \tag{3.2}$$
$$x_{13} + x_{23} \geq 2 \tag{3.3}$$
$$x_{11} + x_{12} + x_{13} \leq 4 \tag{3.4}$$
$$x_{21} + x_{22} + x_{23} \leq 6 \tag{3.5}$$

$\min z = 5x_{11} + x_{12} + x_{13} + 2x_{21} + 6x_{22} + 9x_{23}$
$x_{ij} \geq 0, 1 \leq i \leq 2, 1 \leq j \leq 3$

το οποίο είναι ένα πρόβλημα με διαφορετική φορά ανισοτήτων, αλλά μετατρέπεται απλά στο εξής:

$x_{11} + x_{21} \geq 3$
$x_{12} + x_{22} \geq 5$
$x_{13} + x_{23} \geq 2$
$-x_{11} - x_{12} - x_{13} \geq -4$
$-x_{21} - x_{22} - x_{23} \geq -6$
$\min z$ και $x_{ij} \geq 0$

Με πίνακες το πρόβλημα γράφεται:

$x \geq 0$
$Ax \geq B$
$\min z = Cx$

όπου x, A, B και C έχουν ως εξής:

$$x = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \end{pmatrix} \quad A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 \end{pmatrix}$$

$$B = \begin{pmatrix} 3 \\ 5 \\ 2 \\ -4 \\ -6 \end{pmatrix} \qquad C = \begin{pmatrix} 5 & 1 & 1 & 2 & 6 & 9 \end{pmatrix}$$

Το πρόβλημα μπορούμε βέβαια να το επιλύσουμε εισάγοντας τις 5 μεταβλητές απόκλισης και να συνεχίσουμε κανονικά με τη Simplex. Αλλά η διάσταση του προβλήματος αυξάνει με αυτόν τον τρόπο, οπότε όπως είδαμε στα προηγούμενα μπορούμε να κάνουμε καλύτερα, χάριν του παρακάτω θεωρήματος, κερδίζοντας χρόνο εκτέλεσης χωρίς την εισαγωγή των μεταβλητών απόκλισης. Κατ' αρχάς ας παρατηρήσουμε ότι η αρχική διατύπωση του γραμμικού προβλήματος απαιτεί $\sum_{i=1}^{m} d_i = \sum_{j=1}^{n} b_j$. Διαφορετικά αθροίζοντας τις τρεις πρώτες ανισώσεις και τις δύο επόμενες ξεχωριστά, θα είχαμε:

$$\left.\begin{array}{r} (1)+(2)+(3) \Rightarrow \sum_i \sum_j x_{ij} \geq 10 \\ (4)+(5) \Rightarrow \sum_j \sum_i x_{ij} \leq 10 \end{array}\right\} \Rightarrow \sum_j \sum_i x_{ij} = 10$$

Το οποίο υποδηλώνει ότι και με αυτή τη γραφή του προβλήματος απαιτείται

$$\sum_{i=1}^{m} d_i = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij} = \sum_{j=1}^{n} b_j$$

Μπορούμε τελικά σύμφωνα με το παρακάτω θεώρημα να θεωρούμε τους περιορισμούς ως ισότητες και να έχουμε την ισοδύναμη μορφή:

$x_{ij} \geq 0, \min z$, με τους περιορισμούς:

$$\begin{array}{cccccccc}
x_{11} & & & +x_{21} & & & = 3 \\
& x_{12} & & & +x_{22} & & = 5 \\
& & x_{13} & & & +x_{23} & = 2 \\
x_{11} & +x_{12} & +x_{13} & & & & = 4 \\
& & & x_{21} & +x_{22} & +x_{23} & = 6
\end{array}$$

Θεώρημα:
Αν $\sum_{i=1}^{m} d_i = \sum_{j=1}^{n} b_j$ τότε το πρόβλημα μεταφοράς γράφεται ισοδύναμα (P) ή (P'):

$$\boxed{\begin{array}{l} x \geq 0 \\ Ax = B \\ \min z = \sum_{i,j} c_{ij} x_{ij} \end{array}} \quad (P) \qquad \boxed{\begin{array}{l} x \geq 0 \\ Ax \geq B \\ \min z = \sum_{ij} c_{ij} x_{ij} \end{array}} \quad (P')$$

Απόδειξη:

$(P) \Rightarrow (P')$: προφανές, δηλαδή η βέλτιστη λύση του (P) είναι εφικτή για το (P'), αφού όλοι οι περιορισμοί στο (P') ικανοποιούνται.

$(P') \Rightarrow (P)$: Η βέλτιστη λύση του (P') είναι εφικτή λύση για το (P). Πράγματι, αν είχαμε κάποια παραβίαση για κάποιο περιορισμό j: $\sum_{i=1}^{m} x_{ij} > b_j$ (ή με τον ίδιο τρόπο $\sum_{j=1}^{n} x_{ij} > d_i$) τότε θα είχαμε

$$0 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} x_{ij} \right) + \sum_{i=1}^{m} \left(\sum_{j=1}^{n} - x_{ij} \right) > \sum_{j=1}^{n} b_j + \sum_{i=1}^{m} (-d_i) = 0$$

το οποίο είναι αντίφαση. Επομένως όλοι οι περιορισμοί στο (P) ικανοποιούνται. Άρα έχουμε για τις δύο βέλτιστες λύσεις z_p και $z_{p'}$ αντίστοιχα για τα προβλήματα (P) και (P') ότι $z_p \geq z_{p'}$ και $z'_p \geq z_p$ δηλαδή $z_p = z_{p'}$.

Κεφάλαιο 4

Ακέραιος προγραμματισμός

Ας θεωρήσουμε το γραμμικό πρόγραμμα:

$$(PL)\begin{cases} \min z = cx \\ Ax = b \\ x \geq 0 \end{cases}$$

Όπου όλοι οι συντελεστές $c_i, i = 1, \ldots, n$ (διάνυσμα c), $a_{ij}, 1 \leq i \leq m, 1 \leq j \leq n$ (πίνακας A) και $b_i, 1 \leq i \leq m$ (διάνυσμα b) υποθέτουμε ότι είναι ακέραιοι. Επίσης υποθέτουμε ότι το πολύτοπο

$$\mathcal{P} = \{x | x \in R^n, Ax = b, x \geq 0\}$$

είναι φραγμένο και μη κενό.

Εκτός από ειδικές περιπτώσεις (π.χ. ο πίνακας περιορισμών A να είναι totally unimodular)[1] μια βέλτιστη λύση του (PL) θα εμπεριέχει μεταβλητές x_i με πραγματικές τιμές.

Ας υποθέσουμε ότι οι μεταβλητές x_i αντιπροσωπεύουν πλήθος αντικειμένων μη διαιρουμένων (πλοία, αεροπλάνα, κ.λ.π.). Για παράδειγμα μια αεροπορική εταιρεία αναζητά να πραγματοποιήσει ένα ετήσιο προγραμματισμό πτήσεων, ελαχιστοποιώντας το συνολικό κόστος των αεροπλάνων που θα χρησιμοποιήσει. Οι μεταβλητές αντιπροσωπεύουν

[1] Ένας τετραγωνικός πίνακας ακεραίων B καλείται unimodular αν η ορίζουσά του $det(B)$ παίρνει τιμές ± 1 ή 0. Ένας ακέραιος πίνακας A καλείται total unimodular αν κάθε τετραγωνικός μη ιδιάζων υποπίνακας του A είναι unimodular.

σε αυτή την περίπτωση τον αριθμό των αεροπλάνων από κάθε τύπο για να αγοράσει ή να ενοικιάσει. Δεν είναι επομένως αποδεκτό να έχει μια βέλτιστη λύση η οποία δεν είναι ακέραια.

Θα πρέπει σε αυτή την περίπτωση να επιβάλλουμε στις μεταβλητές συμπληρωματικούς περιορισμούς του τύπου: x_j ακέραιος, $\forall j = 1, \ldots, n$. Το πρόβλημα θα γραφεί λοιπόν:

$$(PI) \begin{cases} \min z = cx \\ Ax = b \\ x \geq 0, x_j \text{ ακέραιος }, 1 \leq j \leq n \end{cases}$$

Ένα τέτοιο πρόβλημα καλείται γραμμικό ακέραιο πρόγραμμα.

Το πρόβλημα (PL) που προέρχεται από το (PI) «ξεχνώντας» τους ακέραιους περιορισμούς καλείται συνεχές γραμμικό πρόγραμμα ή γραμμική χαλάρωση που αντιστοιχεί στο ακέραιο πρόγραμμα (PI).

Η πρώτη ιδέα που έρχεται, όταν αντιμετωπίζουμε ένα πρόβλημα σε ακέραιους αριθμούς, είναι να χρησιμοποιήσουμε μια μέθοδο στρογγυλέματος, π.χ. αντικαθιστώντας, στη βέλτιστη συνεχή λύση, κάθε κλασματική συνιστώσα από τον πλησιέστερο ακέραιο. Το παρακάτω παράδειγμα δείχνει καθαρά την ανεπάρκεια αυτών των μεθόδων και επιτρέπει να καταλάβουμε καλύτερα την ενυπάρχουσα δυσκολία στα προβλήματα ακέραιου προγραμματισμού.

Παράδειγμα

Ας θεωρήσουμε το πρόβλημα του σακιδίου "knapsack" με δύο μεταβλητές και έναν περιορισμό:

$\min Z = -10x_1 - 11x_2$

$10x_1 + 12x_2 \leq 59$

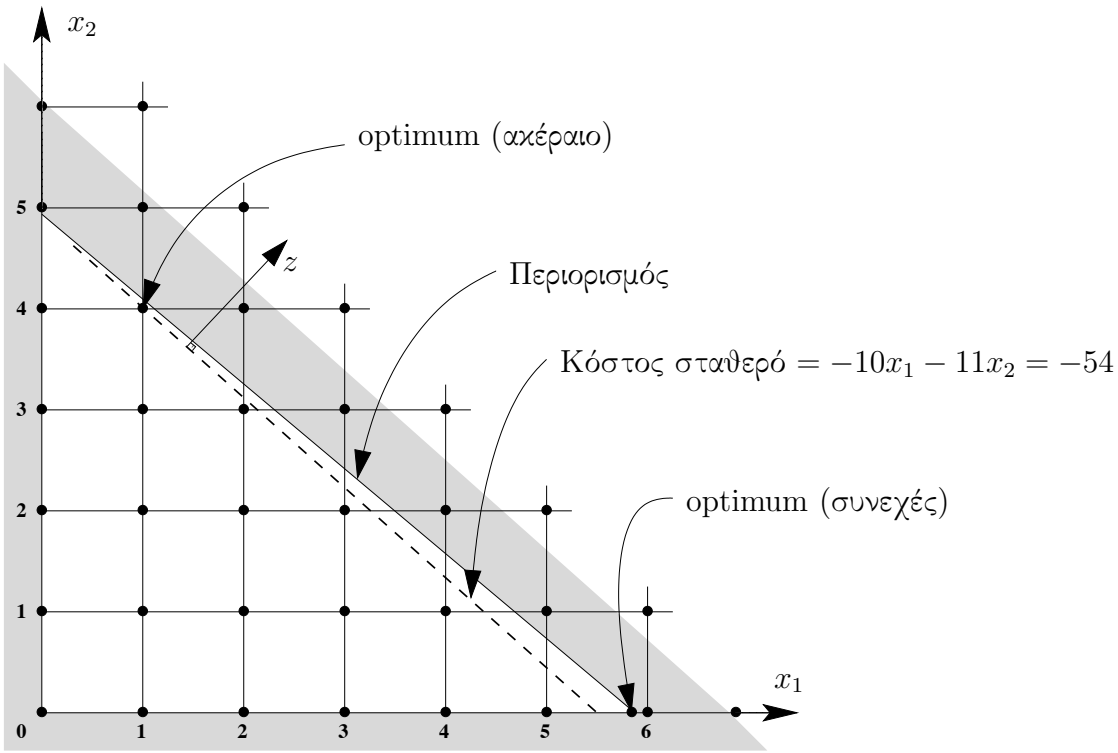
$x_1, x_2 \in \mathbb{N}$

Μπορούμε να παρουσιάσουμε εύκολα το σύνολο των συνεχών λύσεων και των ακεραίων λύσεων στο επίπεδο (βλ. σχήμα 4.1).

Το συνεχές βέλτιστο είναι το σημείο με συντεταγμένες $x_1 = 5,9$ και $x_2 = 0$ για το οποίο $z = -59$. Μια απλή μέθοδος στρογγύλευσης θα οδηγούσε στη λύση $x_1 = 6, x_2 = 0$, η οποία δυστυχώς δεν ικανοποιεί τους περιορισμούς.

Εξετάζοντας τώρα τα σημεία με ακέραιες συντεταγμένες στο εσωτερικό του πολυέδρου, διαπιστώνουμε ότι η βέλτιστη ακέραια λύση είναι το σημείο: $x_1 = 1, x_2 = 4$ για το οποίο η αντικειμενική συνάρτηση έχει την τιμή $z = -54$.

Βλέπουμε ότι αυτό το σημείο δεν έχει καμία σχέση με το συνεχές βέλτιστο, μάλιστα είναι αρκετά μακριά.



Σχήμα 4.1: Το σημείο (1,4) είναι η βέλτιστη λύση του ακέραιου προγράμματος ενώ η βέλτιστη λύση του συνεχούς προγράμματος είναι το σημείο (5.9,0).

Στην πραγματικότητα μπορούμε να κατασκευάσουμε παραδείγματα, για τα οποία η ακέραια βέλτιστη λύση είναι όσο θέλουμε απομεμακρυσμένη από τη συνεχή βέλτιστη λύση.

Αυτό εξηγεί γιατί οι μέθοδοι στρογγύλευσης είναι γενικά αναποτελεσματικές. Ωστόσο σε μερικές περιπτώσεις μπορούν να δώσουν καλές προσεγγιστικές λύσεις. Στο προηγούμενο παράδειγμα, στρογγυλεύοντας το συνεχές βέλτιστο στον κατώτερο ακέραιο, θα βρίσκαμε τη λύση $x_1 = 5, x_2 = 0$ με κόστος $z = -50$ η οποία απέχει λιγότερο από 10% του βέλτιστου).

4.1 Μέθοδοι Διαχωρισμού και Αποτίμησης (Branch and Bound) ή μέθοδος νοερής απαρίθμησης (implicit enumeration)

Η βασική αρχή αυτών των μεθόδων οφείλεται κυρίως στους Land και Doig (1960), Bertier και Roy (1964). Πρακτικά με τη Branch and bound μέθοδο μπορούμε να επιτύχουμε:

1. Μία βέλτιστη λύση
2. Όλες τις βέλτιστες λύσεις
3. Μία ϵ-προσεγγιστική λύση
4. Όλες τις ϵ-προσεγγιστικές λύσεις
5. Μία καλή λύση

Εδώ θα περιοριστούμε όμως στην πρώτη περίπτωση. Ο αναγνώστης μπορεί να προσαρμόσει την παρουσίαση για τις άλλες περιπτώσεις.

Η βασική αρχή της Branch and Bound μεθόδου είναι να διαχωρίσει το σύνολο των εφικτών λύσεων σε μικρότερα υποσύνολα έτσι ώστε να απομονώσει σε κάποιο από αυτά τα υποσύνολα μια βέλτιστη (ή ϵ-προσεγγιστική) λύση. Για την παρουσίαση της αναζήτησης κατασκευάζουμε ένα κατευθυνόμενο δέντρο του οποίου η ρίζα είναι το σύνολο όλων των λύσεων και οι άλλοι κόμβοι του αντιστοιχούν στα υποσύνολα λύσεων.

Ας θεωρήσουμε ένα ακέραιο γραμμικό πρόγραμμα του τύπου:

$$\min z = cx$$
$$Ax = b$$
$$x_j \geq 0, x_j \text{ ακέραιος}, 1 \leq j \leq n$$

όπου A είναι ένας $m \times n$ πίνακας ακεραίων, $b \in \mathbb{Z}^m, x \in \mathbb{Z}^n$.

Το πολύτοπο:

$$\mathcal{P} = \{x \in R^n | Ax = b, x \geq 0\}$$

το οποίο υποθέτουμε φραγμένο, είναι ένα κυρτό πολύεδρο. Έτσι είναι πάντα δυνατό να αντιστοιχίσουμε σε κάθε μεταβλητή x_j φράγματα μεταβολής: $a_j \leq x_j \leq b_j$

Για να καθορίσουμε b_j, π.χ., θα μπορούσαμε να επιλύσουμε το γραμμικό συνεχές πρόγραμμα:

$$\begin{aligned} &\max x_j \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

Έτσι μπορούμε πάντα να πάμε στην περίπτωση όπου κάθε μεταβλητή x_j δεν μπορεί να πάρει παρά μόνο έναν πεπερασμένο αριθμό τιμών.

4.2 Αναγωγή ακεραίου προγράμματος σε ένα πρόγραμμα με δυαδικές μεταβλητές

Μπορούμε να πάμε πιο μακριά ακόμη, παρατηρώντας ότι κάθε μεταβλητή x_j η οποία παίρνει $k+1$ ακέραιες τιμές $0, 1, 2, \ldots, k$ μπορεί να γραφεί σαν ένας γραμμικός συνδυασμός:

$$x_j = y_0 + 2y_1 + 2^2 y_2 + \cdots + 2^p y_p$$

από $p+1$ μεταβλητές $y_0, y_1, \ldots, y_p$ με $y_i = 0$ ή 1.

Ο ακέραιος p είναι ο πιο μικρός ακέραιος έτσι ώστε $k \leq 2^{p+1} - 1$. Έτσι μπορούμε πάντα να αναχθούμε στην περίπτωση όπου το ακέραιο γραμμικό πρόγραμμα (PI) είναι ένα γραμμικό πρόγραμμα με δυαδικές μεταβλητές. Το προφανές μειονέκτημα είναι ότι το πλήθος των μεταβλητών αυξάνει σημαντικά.

Παράδειγμα

Έστω το ακέραιο πρόγραμμα με φραγμένες μεταβλητές:

$$\begin{cases} \min z = 3x_1 - 4x_2 \\ x_1 + 2x_2 \leq 6 \\ -2 \leq x_1 \leq +4 \\ 1 \leq x_2 \leq 3 \\ x_1, x_2 \in \mathbb{N} \end{cases}$$

Χρησιμοποιώντας 5 δυαδικές μεταβλητές (3 για τη x_1 και 2 για τη x_2) θα έχουμε:

$$\begin{aligned} x_1 &= -2 + y_1 + 2y_2 + 4y_3 \\ x_2 &= 1 + y_4 + 2y_5 \end{aligned}$$

Η αντικατάσταση δίνει:

$$\begin{aligned} &\min z = -10 + 3y_1 + 6y_2 + 12y_3 - 4y_4 - 8y_5 \\ &y_1 + 2y_2 + 4y_3 + 2y_4 + 4y_5 \leq 6 \\ &y_1 + 2y_2 + 4y_3 \leq 6 \\ &y_4 + 2y_5 \leq 2 \\ &y_i = 0 \text{ ή } 1,\ 1 \leq i \leq 5 \end{aligned}$$

Γι΄ αυτό το λόγο, θα περιοριστούμε στη συνέχεια στη μελέτη γραμμικών προγραμμάτων με δυαδικές μεταβλητές της μορφής:

$$(P) \begin{cases} \min z &= cx \\ Ax &= b \\ x_j &\in \{0,1\}, \quad \forall x_j = 1, \ldots, n \end{cases}$$

ή ακόμη, συμβολίζοντας S το σύνολο των n-διανυσμάτων με συνιστώσες 0 ή 1:

$$(P) \begin{cases} \min z &= cx \\ Ax &= b \\ x &\in S \end{cases}$$

Αφού $|S| = 2^n$, το σύνολο των λύσεων του (P) περιέχει ένα πεπερασμένο αριθμό στοιχείων.

4.3 Ρητή Απαρίθμηση

Η πιο απλή ιδέα που μπορούμε να φανταστούμε για να επιλύσουμε το πρόβλημα (P) θα ήταν να απαριθμήσουμε τα 2^n στοιχεία του S, και να κρατήσουμε, ανάμεσα στα διανύσματα $x \in S$ που ικανοποιούν τους περιορισμούς $Ax = b$, το διάνυσμα $\bar{x}$ που οδηγεί στην πιο μικρή τιμή για την αντικειμενική συνάρτηση $z = c \cdot x$.

Είναι προφανές ότι ένας τέτοιος αλγόριθμος θα ήταν πεπερασμένος. Αλλά όταν ο αριθμός n των μεταβλητών γίνεται μεγάλος (π.χ. ίσος με 50 για να σχηματίσουμε μια ιδέα) είναι εύκολο να διαπιστώσουμε ότι χρειάζονται αιώνες στον πιο ισχυρό υπολογιστή για να απαριθμήσουμε έστω ένα μέρος από τα 2^n στοιχεία του S.

Για να επιλύσουμε με τον υπολογιστή πραγματικά προβλήματα πρακτικού ενδιαφέροντος (100 δυαδικές μεταβλητές ή παραπάνω) πρέπει λοιπόν να αναζητήσουμε μία αλγοριθμική αρχή, που μας επιτρέπει να καθορίσουμε μία βέλτιστη λύση χωρίς να χρειάζεται να απαριθμήσουμε πλήρως το σύνολο των στοιχείων του S.

Ακριβώς αυτή η ιδέα της νοερής απαρίθμησης (παρά ρητής απαρίθμησης) των λύσεων είναι στη βάση των μεθόδων διαχωρισμού και αποτίμησης (Branch and Bound), ονομαζόμενες έτσι επειδή χρησιμοποιούν μια αναπαράσταση του συνόλου S υπό τη μορφή ενός κατευθυνόμενου δέντρου.

4.4 Η έννοια του διαχωρισμού (branching rule)

Οι κόμβοι του (κατευθυνόμενου) δέντρου αντιστοιχούν σε υποσύνολα του S (δηλ. υποσύνολα από διανύσματα 0-1) και είναι κατανεμημένα σε επίπεδα με τον εξής τρόπο: Υπάρχει μόνο ένας κόμβος στο επίπεδο ''0'' (ονομαζόμενο ρίζα του δένδρου) ο οποίος αντιστοιχεί σε ολόκληρο το σύνολο S.

Για να κατασκευάσουμε το επίπεδο ''1'' του δέντρου πρέπει κατ΄ αρχήν να επιλέξουμε (αυτή η επιλογή είναι αυθαίρετη) μια μεταβλητή, π.χ. x_1. Το επίπεδο 1 εμπεριέχει δύο κόμβους συμβολιζόμενους $S_{\bar{1}}$ και S_1 οι οποίοι αντιστοιχούν: ο πρώτος στο υποσύνολο διανυσμάτων 0-1 για τα οποία η μεταβλητή x_1 έχει την τιμή 0. Ο δεύτερος στο υποσύνολο διανυσμάτων 0-1 για τα οποία η μεταβλητή x_1 έχει την τιμή 1. Προφανώς έχουμε: $S_{\bar{1}} \subset S$ και $S_1 \subset S$. Επιπλέον $S_{\bar{1}}$ και S_1 σχηματίζουν μια διαμέριση του S, δηλαδή $S_{\bar{1}} \cap S_1 = \emptyset$ και $S_{\bar{1}} \cup S_1 = S$.

Οι κόμβοι S και $S_{\bar{1}}$ (αντίστ. S και S_1) συνδέονται με ένα τόξο το οποίο αντιπροσωπεύει τη σχέση της έγκλισης. Λέμε ότι ''διαχωρίσαμε'' S σχετικά με τη μεταβλητή x_1. Με τον ίδιο τρόπο, για να κατασκευάσουμε το επίπεδο ''2'' θα επιλέξουμε (αυθαίρετα) μια δεύτερη μεταβλητή, π.χ. x_2. Βρίσκουμε έτσι 2^2 κόμβους επιπέδου 2:

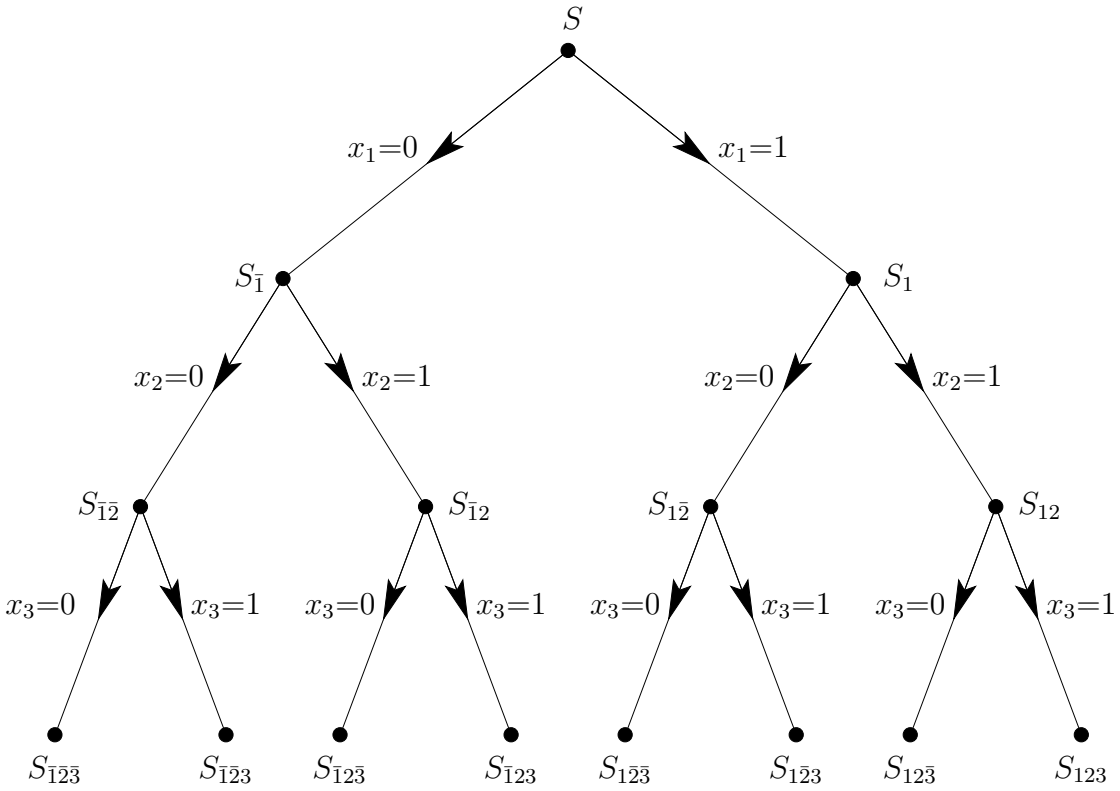
$S_{\bar{1}\bar{2}}$ σύνολο διανυσμάτων 0-1 για τα οποία $x_1 = 0$ και $x_2 = 0$,
$S_{\bar{1}2}$ σύνολο διανυσμάτων 0-1 για τα οποία $x_1 = 0$ και $x_2 = 1$,
$S_{1\bar{2}}$ σύνολο διανυσμάτων 0-1 για τα οποία $x_1 = 1$ και $x_2 = 0$,
S_{12} σύνολο διανυσμάτων 0-1 για τα οποία $x_1 = 1$ και $x_2 = 1$.

Οι σχέσεις έγκλισης αναπαρίστανται με τόξα που συνδέουν τον κόμβο $S_{\bar{1}}$ με τους κόμβους $S_{\bar{1}\bar{2}}$ και $S_{\bar{1}2}$ αφ΄ ενός και τον κόμβο S_1 με τους κόμβους $S_{1\bar{2}}$ και S_{12} αφ΄ ετέρου.

Λέμε ότι διαχωρίσαμε $S_{\bar{1}}$ (αντίστ. S_1) σχετικά με τη μεταβλητή x_2.

Στο σχήμα 4.2 βλέπουμε το (κατευθυνόμενο) δέντρο, που αντιστοιχεί σ΄ ένα πρόβλημα του τύπου (P) και περιέχει 3 δυαδικές μεταβλητές x_1, x_2, x_3.

Ας παρατηρήσουμε ότι τα οκτώ στοιχεία του S αντιστοιχούν στους κόμβους χωρίς διαδόχους, ονομαζόμενοι τελικοί κόμβοι του δέντρου.



Σχήμα 4.2: Το δένδρο που αντιστοιχεί σε ένα ακέραιο γραμμικό πρόγραμμα με 3 δυαδικές μεταβλητές. Οι τελικοί κόμβοι (φύλλα) αντιστοιχούν σε όλες τις δυνατές λύσεις του προβλήματος, δηλαδή 2^3 λύσεις.

4.5 Αποτίμηση (evaluation function)

Ας υποθέσουμε ότι, για κάθε κόμβο S_i του προηγούμενου δέντρου, μπορούμε να καθορίσουμε με υπολογισμό, μια πρόχειρη εκτίμηση, δηλαδή ένα κάτω φράγμα $f(S_i)$ του κόστους $c \cdot x$ της καλύτερης λύσης $x \in S_i$:

$$f(S_i) \leq \min\{c \cdot x\}$$

Η συνάρτηση f καλείται συνάρτηση εκτίμησης. Ας κατασκευάσουμε προοδευτικά, ξεκινώντας από το επίπεδο ''0'' (κόμβος S), το δέντρο των υποσυνόλων του S, κάνοντας τη διαχώριση και εξετάζοντας τις μεταβλητές με μια σειρά (a priori αυθαίρετη). Υποθέτουμε ότι έχουμε ένα πρόβλημα ελαχιστοποίησης.

Έστω $\bar{x}$ μια λύση του προβλήματος (P), με κόστος $\bar{z} = c \cdot \bar{x}$, καθορισμένη είτε κατά τη διάρκεια των προηγουμένων βημάτων της κατασκευής του δέντρου, είτε με οποιαδήποτε προσεγγιστική μέθοδο (ευριστική). Αν δεν έχουμε καμία λύση μπορούμε να θέσουμε $\bar{z} = +\infty$.

Ας υποθέσουμε τότε ότι ο κόμβος S_i του δέντρου είναι έτσι ώστε:

$$f(S_i) \geq \bar{z}$$

Εξ΄ ορισμού της συνάρτησης f, καμμία λύση που περιέχεται μέσα στο S_i δεν μπορεί να είναι καλύτερη από τη $\bar{z}$, και επομένως είμαστε σίγουροι ότι S_i δεν περιέχει καμία βέλτιστη λύση.

Αυτή η διαπίστωση επιτρέπει να αποφύγουμε την εξερεύνηση (την απαρίθμηση) όλων των αμέσων και εμμέσων διαδόχων του κόμβου S_i μέσα στο δέντρο.

Περιορίζοντας, σε κάθε κόμβο, την απαρίθμηση στους διαδόχους των συγκεκριμένων κόμβων S_i για τους οποίους:

$$f(S_i) < \bar{z}$$

μπορούμε να επιτύχουμε μια σημαντική μείωση του αριθμού των πραγματικά εξεταζόμενων κόμβων, ικανή για να επεξεργαστούμε προβλήματα ενδιαφέροντος μεγέθους. Αυτή είναι η γενική αρχή της μεθόδου. Να απορρίπτει χώρους λύσεων μέσα στους οποίους σίγουρα δεν εμπεριέχεται η βέλτιστη λύση.

4.6 Πρακτική υλοποίηση

Στην προηγούμενη περιγραφή της μεθόδου διαπιστώνουμε μεγάλους βαθμούς ελευθερίας, και μπορούμε να πούμε ότι υπάρχουν τόσοι αλγόριθμοι όσοι και οι τρόποι να επιλέξει κανείς:

(α) τη φύση της συνάρτησης εκτίμησης (evaluation function),

(β) τον προς διαχώριση κόμβο σε κάθε βήμα (search strategy),

(γ) τη μεταβλητή πάνω στην οποία θα πραγματοποιήσουμε τη διαχώριση του επιλεγμένου κόμβου (branching rule)

Ας εξετάσουμε διαδοχικά αυτά τα τρία σημεία: Η επιλογή του (α) δεν είναι πάντα προφανής. Μπορούμε πάντα να χρησιμοποιήσουμε για ''πρόχειρη'' εκτίμηση τη συνεχή βέλτιστη λύση του γραμμικού προγράμματος περιορισμένου στις ελεύθερες μεταβλητές. Εν τούτοις, μια τέτοια εκτίμηση είναι γενικά ''ακριβή'' σε χρόνο υπολογισμού. Γι΄ αυτό το λόγο θα προτιμούσαμε μια συνάρτηση εκτίμησης λιγότερο καλή (δηλαδή απέχουσα περισσότερο από την ακέραια βέλτιστη λύση) αλλά που μπορούμε να βρούμε ταχύτερα.

Για την επιλογή του (β) πολυάριθμες στρατηγικές έχουν προταθεί, εκ των οποίων καμμία δεν αποβαίνει συστηματικά η καλύτερη. Στη μέθοδο ονομαζόμενη ''αρχικά κατά βάθος'', επιλέγουμε τον κόμβο με το υψηλότερο επίπεδο ανάμεσα στους κόμβους οι οποίοι δεν είναι ακόμη διαχωρισμένοι. Αυτή η μέθοδος στοχεύει να προσδιορίσει το ταχύτερο δυνατόν μια λύση του προβλήματος.

Στη μέθοδο ονομαζόμενη ''αρχικά κατά πλάτος'' εξαντλούμε συστηματικά τους κόμβους κάθε επιπέδου. Στη μέθοδο ονομαζόμενη ''αρχικά η καλύτερη'' επιλέγουμε συστηματικά τον κόμβο με την καλύτερη αποτίμηση, παρατηρώντας ότι είναι αυτός, διαισθητικά, ο πιθανότερος για να περιέχει ανάμεσα στους διαδόχους του μια βέλτιστη λύση. Ο κίνδυνος, με αυτή την τεχνική, είναι να υποχρεωθούμε να εξερευνήσουμε ένα σημαντικό μέρος του δέντρου μέχρι να ανακαλύψουμε μια αποδεκτή λύση. Αντίθετα, η πρώτη λύση που βρίσκουμε είναι γενικά καλύτερη ποιοτικά (δηλ. μικρό κόστος για ένα πρόβλημα ελαχιστοποίησης) από αυτήν που βρίσκουμε με την τεχνική ''αρχικά κατά βάθος''.

Η επιλογή του (γ), οδηγεί επίσης, σε πολλές διαφοροποιήσεις. Συχνά, η σειρά των μεταβλητών είναι καθορισμένη μια για πάντα, είτε με τρόπο εντελώς αυθαίρετο, είτε ακολουθώντας ευριστικά κριτήρια δικαιολογημένα διαισθητικά. Αλλά πρέπει να σημειώσουμε ότι αυτή η επιλογή μπορεί να είναι καθοριστική (εξαρτωμένης της υιοθετημένης

σειράς ο χρόνος εκτέλεσης μπορεί να επιδεινώνεται σημαντικά από 1 μέχρι 10 φορές ή ακόμη από 1 μέχρι 100 φορές). Αυτό οδήγησε στην έμπνευση στρατηγικών πολύ πιο ευέλικτων (δυναμικών) στις οποίες επιλέγει κανείς σε κάθε βήμα τη μεταβλητή η οποία φαίνεται η καλύτερη σύμφωνα με κάποιο κριτήριο. Παραδείγματος χάριν, στη μέθοδο ονομαζόμενη των ''penalties'' αντιστοιχούμε σε κάθε μεταβλητή x_i, που πρόκειται να χρησιμοποιηθεί για τη διαχώριση ένα αριθμό p_i (penalty), τιμή ακριβή ή προσεγγιστική της διαφοράς ανάμεσα στις νέες αποτιμήσεις που επιτυγχάνονται θέτοντας $x_i = 0$ από τη μια, $x_i = 1$ από την άλλη. Αν πραγματοποιήσουμε το διαχωρισμό ακολουθώντας τη μεταβλητή x_i που αντιστοιχεί στο μέγιστο penalty p_i, θα βρούμε δύο νέα υποσύνολα εκ των οποίων το ένα θα έχει μεγαλύτερη πιθανότητα απ΄ ότι το άλλο να περιέχει μια βέλτιστη λύση. Πρόκειται, κατά κάποιο τρόπο, για την επιλογή που δίνει τη ''μεγαλύτερη πληροφορία''. Αυτή η τεχνική, οδηγεί στην ελαχιστοποίηση των κινδύνων να εξερευνήσουμε αδίκως έναν κλάδο του δέντρου ενώ η βέλτιστη λύση περιέχεται στον άλλο.

4.7 Αλγόριθμος Διάσχισης και Διαχώρισης (Branch and Bound)

Παρακάτω παρουσιάζεται συνολικά ο αλγόριθμος για ένα πρόβλημα ελαχιστοποίησης. Οι αλλαγές που απαιτούνται για ένα πρόβλημα μεγιστοποίησης είναι προφανείς. Ο αλγόριθμος χρησιμοποιεί τη συνάρτηση EVAL, που υλοποιεί την αποτίμηση ενός υποπροβλήματος (P_i). Εκ των προτέρων θα πρέπει να έχουν οριστεί ο τρόπος αποτίμησης (συνάρτηση f), η στρατηγική διαχώρισης (επιλογή μεταβλητής x_i) και η στρατηγική διάσχισης. Η στρατηγική διάσχισης υλοποιείται από τη δομή Q. Για την κατά πλάτος στρατηγική η Q είναι ουρά (Queue), για την κατά βάθος στρατηγική η Q είναι στοίβα (stack) και γι ατις στρατηγικές η καλύτερη κατ΄ αρχήν (The Best First) ή η χειρότερη κατ΄ αρχήν (The Worst First) η Q μπορεί να είναι ένας σωρός (heap). Ας παρατηρήσουμε ότι όταν στην αποτίμηση επιτυγχάνουμε ένα κάτω φράγμα το οποίο είναι και εφικτή λυση του προβλήματος και με μικρότερη τιμή (βήμα 4 στη συνάρτηση EVAL) τότε επωφελούμαστε να ενημερώσουμε την τρέχουσα καλύτερη λύση $\bar{z}$ και δε συνεχίζουμε τη διαχώριση σε αυτό τον κόμβο του δέντρου. Ουσιαστικά η αποκοπή ενός κλάδου στο δέντρο πραγματοποιείται με τρεις τρόπους:

1. *Ο χώρος των λύσεων στο δεδομένο κόμβο είναι κενός* (by infeasibility)

2. *Η αποτίμηση δίνει μια εφικτή λύση με καλύτερη τιμή από την τρέχουσα λύση* (by optimality)

3. *Η αποτίμηση είναι χειρότερη από την τιμή της τρέχουσας λύσης* (by bound).

Στο αλγοριθμικό σχήμα που περιγράφουμε η διαχώριση είναι δυαδική και το αναπτυσσόμενο δέντρο δυαδικό. Εύκολα όμως το σχήμα αυτό γενικεύεται για διαχωρίσεις μη δυαδικές οδηγώντας στην ανάπτυξη ενός μη δυαδικού δέντρου.

Branch and Bound(S σύνολο λύσεων, $\bar{z}$ βέλτιστη λύση)
1. Αρχικοποίηση
   $\bar{z} = z_0$ /* αρχική λύση αν υπάρχει, διαφορετικά $z_0 = +\infty$ */
   $Q = \emptyset$, EVAL(S, Q)
2. **while** $Q \neq \emptyset$
3. $\quad s = \text{top}(Q)$
4. $\quad$ Επέλεξε μεταβλητή x_i από τις μη φιξαρισμένες μεταβλητές του S
5. $\quad P_{\bar{i}}$: υποπρόβλημα με χώρο λύσεων $s_{\bar{i}}$ όπου η μεταβλητή $x_{\bar{i}}$ είναι 0.
6. $\quad P_i$: υποπρόβλημα με χώρο λύσεων s_i όπου η μεταβλητή x_i είναι 0.
7. EVAL$(S_{\bar{i}}, Q)$, EVAL(S_i, Q)
8. Επέστρεψε την καλύτερη λύση $\bar{z}$ που βρέθηκε σε όλη την αναζήτηση

Σχήμα 4.3: Αλγόριθμος Διαχωρισμού και Διάσχισης για ένα πρόβλημα ελαχιστοποίησης. Το βήμα 4 καθορίζει τη στρατηγική Διαχώρισης ενώ η υλοποίηση της δομής Q καθορίζει τη στρατηγική Διάσχισης.

EVAL(χώρος λύσεων s, structure q)
1. **if** $s = \emptyset$ το πρόβλημα P διαγράφεται από την q, exit
2. Εύρεση κάτω φράγματος $f(s)$ για το πρόβλημα p
3. **if** $f(s) \geq \bar{z}$ το πρόβλημα p διαγράφεται από την q
4. **else if** $f(s)$ είναι εφικτή λύση για το πρόβλημα p $\bar{z} = f(s)$ και το p διαγράφεται από την q
5. **else** το πρόβλημα p προστίθεται στην q

Σχήμα 4.4: Συνάρτηση αποτίμησης του χώρου των λύσεων ενός υποπροβλήματος p. $\bar{z}$ είναι η τιμή της τρέχουσας καλύτερης λύσης.

4.8 Παράδειγμα: το πρόβλημα του σακιδίου

Για να δούμε τη λειτουργία των μεθόδων διαχωρισμού και αποτίμησης, ας θεωρήσουμε το ακόλουθο πρόβλημα:

$$(P) \begin{cases} \min z = -20x_1 - 16x_2 - 11x_3 - 9x_4 - 7x_5 - x_6 \\ 9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 12 \\ x_j \geq 0, x_j \in \{0,1\}, 1 \leq j \leq 6 \end{cases}$$

το οποίο είναι της μορφής:

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_j x_j \leq b \\ x_j = 0 \text{ ή } 1 \end{cases}$$

και είναι το περίφημο πρόβλημα «knapsack» ή πρόβλημα του σακιδίου.

Παρατηρούμε στο παράδειγμα ότι οι μεταβλητές είναι διατεταγμένες, σε αύξουσα σειρά, σύμφωνα με τα πηλίκα $\frac{c_j}{a_j}$ και ότι η συνεχής βέλτιστη λύση, που ευρίσκουμε αντικαθιστώντας τον ακέραιο περιορισμό $x_j = 0$ ή 1 με $0 \leq x_j \leq 1$ και εφαρμόζοντας ένα γνωστό άπληστο (greedy) αλγόριθμο[2] είναι:

[2]Όσο απομένει αρκετός $b \geq a_j$ χώρος θέσε $x_j = 1, b = b - a_j, j = 1, \ldots, m$. Διαφορετικά όταν $a_{m+1} > b$ θέσε $x_{m+1} = b/a_{m+1}$ και $x_j = 0, \forall j = m+2, \ldots, n$. Αποδεικνύεται εύκολα ότι ο

$$x_1 = 1, x_2 = \frac{3}{8}, \text{ με τιμή } z = -20 - 6 = -26$$

Αυτή η τιμή συνιστά λοιπόν, πάνω στο σύνολο όλων των δυνατών ακεραίων λύσεων, μια πρώτη αποτίμηση (πράγματι, θεωρώντας τους ακέραιους περιορισμούς, δεν μπορεί παρά να αυξήσουμε την τιμή του βέλτιστου).

Ας παρατηρήσουμε τώρα ότι μπορούμε να βρούμε εύκολα μια ακέραια προσεγγιστική λύση για το πρόβλημα χρησιμοποιώντας τον ευριστικό αλγόριθμο του Σχήματος 4.5.

(α) Στην αρχή όλες οι μεταβλητές είναι ''0''.
(β) Αναζήτησε τη μεταβλητή με το πιο μικρό πηλίκο $\frac{c_j}{a_j}$ στην οποία μπορούμε να καταχωρήσουμε την τιμή 1, σεβόμενοι τον περιορισμό, και σταθεροποίησε αυτή τη μεταβλητή στην τιμή 1. Επανέλαβε το (β) για όλες τις μεταβλητές.

Σχήμα 4.5: Ευριστικός αλγόριθμος για το διακριτό 0-1 Knapsack

Εδώ αυτός ο αλγόριθμος θα έδινε: $x_1 = 1$ μετά $x_6 = 1$. Η ακέραια λύση που επιτυγχάνουμε έχει τιμή $z = -21$.

Ας μελετήσουμε τώρα τα διαδοχικά βήματα της μεθόδου διαχωρισμού και αποτίμησης.

Στο πρώτο επίπεδο ας διαχωρίσουμε το σύνολο S όλων των λύσεων σε δύο υποσύνολα: το σύνολο S_1 με τα διανύσματα 0-1 για τα οποία $x_1 = 1$ και το σύνολο $S_{\bar{1}}$ των διανυσμάτων 0-1 για τα οποία $x_1 = 0$.

Παρατήρηση

Η επιλογή της μεταβλητής x_1 για να πραγματοποιήσουμε το διαχωρισμό είναι αυθαίρετη. Ας υπενθυμίσουμε ότι η επιλογή δεν είναι αδιάφορης σημασίας και η αποτελεσματικότητα της μεθόδου μπορεί να εξαρτάται σημαντικά από αυτή την επιλογή. Στη συνέχεια του παραδείγματος, οι διαδοχικές διαχωρίσεις θα γίνουν σύμφωνα με την αύξουσα διάταξη των $\frac{c_j}{a_j}$. Αυτή η επιλογή μπορεί να δικαιολογηθεί διαισθητικά. Μπορούμε επίσης να εξακριβώσουμε ότι οδηγεί στη λύση ταχύτερα απ΄ ότι υιοθετώντας π.χ. την αντίστροφη διάταξη.

αλγόριθμος αυτός επιστρέφει τη βέλτιστη λύση.

Σε αυτό το επίπεδο, μπορούμε εύκολα να βρούμε μια αποτίμηση της καλύτερης ακέραιας λύσης που περιέχεται σε κάθε ένα από τα υποσύνολα S_1 και $S_{\bar{1}}$.

Ας πάρουμε το υποσύνολο S_1. Αφού γνωρίζουμε ότι $x_1 = 1$, το πρόβλημα περιοριζόμενο στις μεταβλητές $x_2, \ldots, x_6$ (οι οποίες καλούνται 'ελεύθερες'') είναι:

$$(P_1) \begin{cases} \min -20 - 16x_2 - 11x_3 - 9x_4 - 7x_5 - x_6 \\ 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 3 \\ x_j \in \{0, 1\} \end{cases}$$

και επιδέχεται τη συνεχή λύση:

$$x_2 = \frac{3}{8}, \text{ με τιμή } z = -20 - 6 = -26.$$

Η τιμή -26 είναι λοιπόν μια αποτίμηση της καλύτερης ακέραιας λύσης μέσα στο S_1. Θα τη συμβολίσουμε με $f(S_1)$.

Ας πάρουμε τώρα το υποσύνολο $S_{\bar{1}}$. Αφού ξέρουμε ότι $x_1 = 0$, το πρόβλημα περιορισμένο στις ελεύθερες μεταβλητές είναι:

$$(P_{\bar{1}}) \begin{cases} \min -16x_2 - 11x_3 - 9x_4 - 7x_5 - x_6 \\ 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 12 \\ x_j \in \{0, 1\} \end{cases}$$

και επιδέχεται τη συνεχή βέλτιστη λύση:

$$x_2 = 1, x_3 = \frac{2}{3}, \text{ με τιμή } f(S_{\bar{1}}) = -16 - \frac{22}{3} = -\frac{70}{3} (\approxeq -23, 3)$$

Αφού η μέχρι τώρα ευρεθείσα καλύτερη ακέραια λύση ($x_1 = 1, x_6 = 1$) έχει τιμή -21, είναι καθαρό ότι κάθε ένα από τα υποσύνολα S_1 ή $S_{\bar{1}}$ μπορεί να εμπεριέχει μια καλύτερη λύση, και ειδικότερα μια βέλτιστη λύση. Θα πρέπει λοιπόν να διαχωρίσουμε κάθε ένα από τα υποσύνολα σε δύο υποσύνολα τα οποία, με τη σειρά τους, θα οδηγήσουν σε μια αποτίμηση, και ούτω καθ' εξής μέχρι τον εντοπισμό της βέλτιστης λύσης.

Εντούτοις στο σημείο όπου είμαστε, υπάρχει ένα σημαντικό πρόβλημα: ποιο από τα δύο S_1 ή $S_{\bar{1}}$ θα διαχωρίσουμε πρώτο;

Όπως είπαμε ήδη, διάφοροι κανόνες μπορούν να υιοθετηθούν. Ας διαλέξουμε εδώ, τη μέθοδο της εξερεύνησης ''αρχικά την καλύτερη'' (the Best First), η οποία βασίζεται στην εξής παρατήρηση:

Αν η αποτίμηση ενός υποσυνόλου είναι μία καλή προσέγγιση (δηλ. διαφέρει λίγο) από την καλύτερη ακέραια λύση που περιέχεται σε αυτό το υποσύνολο, είναι λογικό να αναζητήσουμε κατ΄ αρχήν τη βέλτιστη λύση μέσα στο υποσύνολο με την πιο μικρή αποτίμηση. Στο παράδειγμά μας είναι το υποσύνολο S_1 που έχει την πιο μικρή αποτίμηση (δηλαδή $\hat{z}_1 = -26$).

Ας διαχωρίσουμε λοιπόν S_1 σε δύο υποσύνολα:

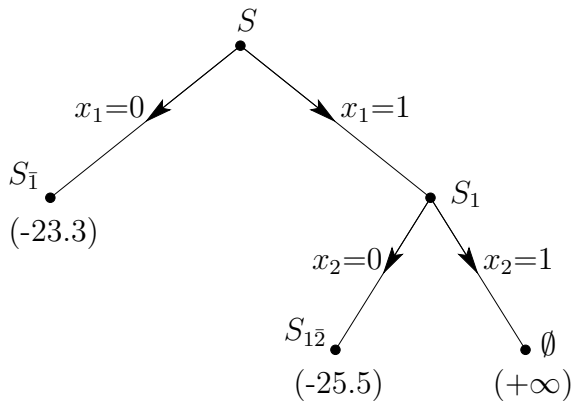
- το υποσύνολο $S_{1\bar{2}}$ με $x_1 = 1$ και $x_2 = 0$
- το υποσύνολο S_{12} με $x_1 = 1$ και $x_2 = 1$

Παρατηρούμε αμέσως ότι το σύνολο S_{12} είναι κενό γιατί $x_1 = 1$ και $x_2 = 1$ δεν επιτρέπει σε καμία περίπτωση να ικανοποιήσουμε τον περιορισμό. Δεν θα ληφθεί λοιπόν καθόλου υπ΄ όψιν στη συνέχεια, και ένας καλός τρόπος για να πραγματοποιήσουμε αυτό είναι να του καταχωρήσουμε μια αποτίμηση $+\infty$.

Για το σύνολο $S_{1\bar{2}}$ έχουμε την αποτίμηση:

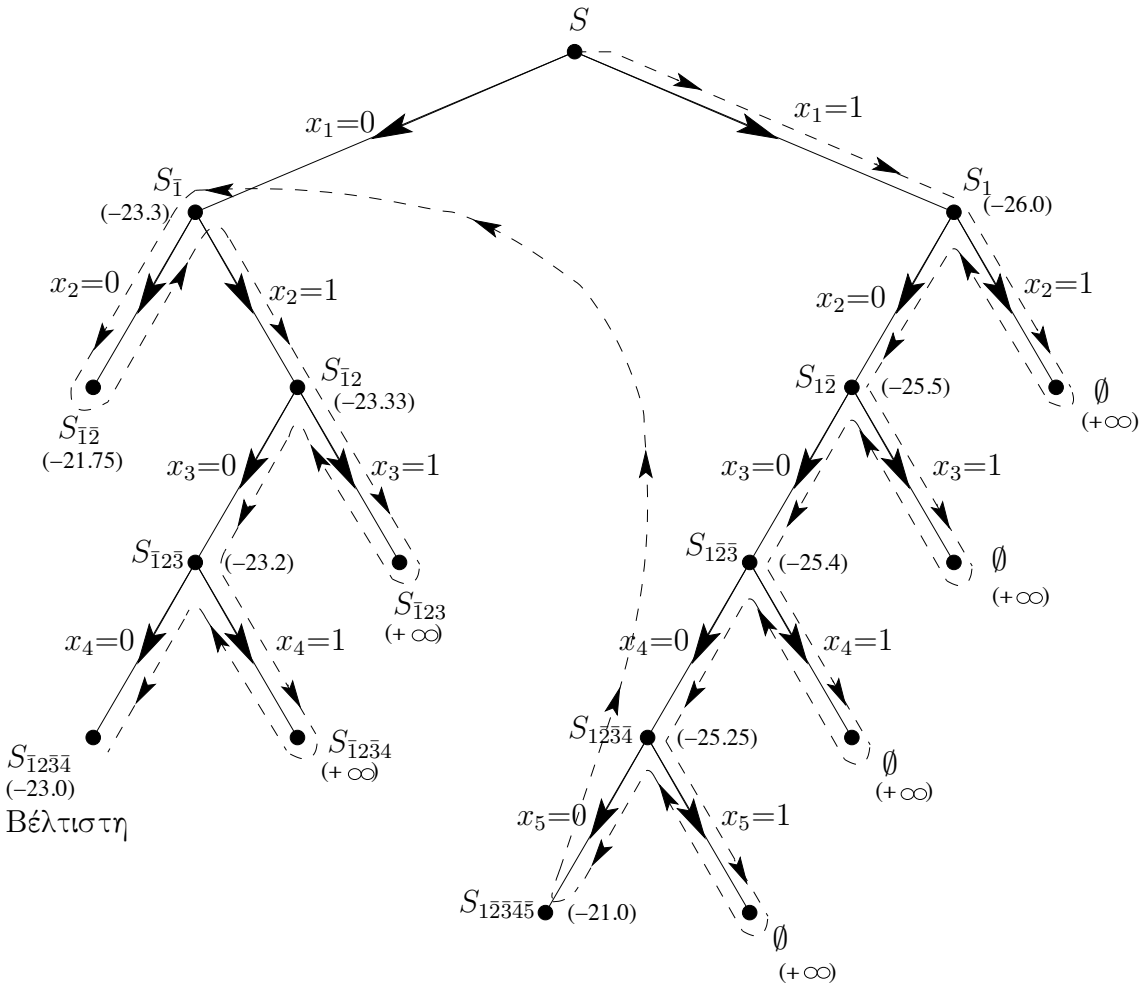
$$-20 - \frac{11}{2} = -25,5$$

Η διαδικασία που αναπτύξαμε μέχρι εδώ φαίνεται στο δέντρο του σχήματος 4.6.



Σχήμα 4.6: Ένα τμήμα του δένδρου του παραδείγματος. Το υποσύνολο S_{12} δεν περιέχει εφικτές λύσεις και άρα η αναζήτηση διακόπτεται σε αυτό τον κλάδο.

Τα σημεία (κόμβοι) αντιστοιχούν στα διαδοχικά υποσύνολα. Οι αποτιμήσεις φαίνονται στις παρενθέσεις για κάθε κόμβο (υποσύνολο). Τα τόξα αντιστοιχούν στις σχέσεις έγκλισης των υποσυνόλων.



Σχήμα 4.7: Το δένδρο αναζήτησης με τη στρατηγική διάσχισης ''αρχικά η καλύτερη'' για το παράδειγμα του σακιδίου. Στον κόμβο $S_{\bar{1}2\bar{3}\bar{4}}$ έχει βρεθεί η βέλτιστη λύση.

Στο σημείο αυτό, το υποσύνολο $S_{1\bar{2}}$ έχει την πιο μικρή αποτίμηση $-25,5$. Αν το διαχωρίσουμε (σχετικά με τη μεταβλητή x_3) ευρίσκουμε τα δύο υποσύνολα:

$S_{1\bar{2}\bar{3}}$ με αποτίμηση $-20 - \frac{27}{5} = -25,4$

$S_{1\bar{2}3}$ με αποτίμηση $+\infty (S_{1\bar{2}3} = \emptyset)$

Συνεχίζοντας έτσι ευρίσκουμε διαδοχικά $S_{1\bar{2}\bar{3}\bar{4}}$ (με αποτίμηση -25,25) και $S_{1\bar{2}\bar{3}\bar{4}\bar{5}}$ (με αποτίμηση -21) (βλέπε Σχήμα 4.7).

Επειδή η αποτίμηση του $S_{\bar{1}}$ είναι καλύτερη (-23,3) ξαναφεύγουμε από το $S_{\bar{1}}$ όπου το διαχωρίζουμε στα $S_{\bar{1}\bar{2}}$(με -21,75) και $S_{\bar{1}2}$(με -23,33). Ξαναφεύγουμε από το $S_{\bar{1}2}$

που διαχωρίζουμε σε $S_{\bar{1}2\bar{3}}$ (-23,2) και $S_{\bar{1}23}(+\infty)$. Μετά ξαναφεύγουμε από $S_{\bar{1}2\bar{3}}$ που διαχωρίζουμε σε $S_{\bar{1}2\bar{3}\bar{4}}$ (-23) και $S_{\bar{1}2\bar{3}4}(+\infty)$.

Σε αυτό το στάδιο, η συνεχής λύση του προβλήματος $P_{\bar{1}2\bar{3}\bar{4}}$ περιορισμένο στις ελεύθερες μεταβλητές δίνει μια ακέραιη λύση:

$$x_2 = 1, x_5 = 1 \text{ με τιμή } -23$$

Αφού κανένα από τα υποσύνολα που απομένουν για εξέταση δεν έχει αποτίμηση κατώτερη από -23, μπορούμε να συμπεράνουμε ότι αυτή η λύση είναι *μια βέλτιστη λύση του προβλήματος*.

Αυτό το παράδειγμα επιτρέπει επίσης να αποτιμήσουμε την αποτελεσματικότητα της μεθόδου. Πράγματι, στους $2^7 - 1 = 127$ κόμβους που περιέχει θεωρητικά το πλήρες δέντρο, διαπιστώνουμε ότι μόνο 17 εξετάστηκαν για την εύρεση της βέλτιστης λύσης, το οποίο σημαίνει μια σημαντική μείωση σε χρόνο αλλά και μνήμη.

Συμβιβασμός: Κόστος - αποτελεσματικότητα της αποτίμησης

Κατά ένα γενικό τρόπο, ο αριθμός των προς εξέταση κόμβων είναι τόσο πιο μικρός όσο η αποτίμηση ενός κόμβου συνιστά μια καλή προσέγγιση του ακέραιου βέλτιστου, μέσα στο υποσύνολο των αντίστοιχων λύσεων, δηλαδή η απόκλιση

$$\min_{x \in s_i}\{c \cdot x\} - f(S_i)$$

είναι μικρή.

Στην πράξη, θα πρέπει γενικά να βρούμε ένα καλό συμβιβασμό ανάμεσα σε: συναρτήσεις αποτίμησης που συνιστούν προσεγγίσεις αρκετά χοντρικές, αλλά πολύ γρήγορες στον υπολογισμό, και συναρτήσεις αποτίμησης καλύτερης προσέγγισης που απαιτούν μεγάλο χρόνο υπολογισμού.

Πρέπει να γνωρίζουμε ότι δεν υπάρχει γενική ικανοποιητική μέθοδος εφαρμόσιμη σε οποιοδήποτε πρόβλημα. Αυτός ο συμβιβασμός πρέπει να ξαναεξετάζεται για κάθε τύπο προβλήματος που έχουμε για επεξεργασία, αν θέλουμε να επιταχύνουμε την επίλυσή του με τρόπο αποτελεσματικό.

4.9 Ένα παράδειγμα μεγιστοποίησης

Ας θεωρήσουμε πάλι το πρόβλημα του σακιδίου με δυαδικές μεταβλητές 0-1. Το πρόβλημα ως ένα γραμμικό πρόγραμμα στο $\{0,1\}$ με ένα περιορισμό έχει ως εξής:

$$\max z = \sum_{i=1}^{n} c_i x_i$$

$$\sum_{i=1}^{n} a_i x_i \leq b$$

$$x \in \{0,1\}^n$$

Έστω το εξής στιγμιότυπο:

$$\begin{aligned} n &= 5 \\ a &= (18, 12, 15, 16, 13) \\ c &= (27, 9, 30, 16, 6.5) \\ b &= 45 \end{aligned}$$

4.9.1 Κανόνας Διαχώρισης

Σε ένα κόμβο s του δέντρου μερικά αντικείμενα θα είναι επιλεγμένα, άλλα θα έχουν απαγορευθεί και άλλα θα έχουν μια κατάσταση ακόμη μη καθορισμένη και θα λέμε ότι είναι ελεύθερα. Θα διαχωρίσουμε ένα κόμβο διαλέγοντας μια μεταβλητή x_i ελεύθερη και θέτοντας $x_i = 1$ (το αντικείμενο i επιλέγεται) ή $x_i = 0$ (i δεν επιλέγεται). Η μεταβλητή x_i είναι αυτή με τη μέγιστη σχετική αξία c_i/a_i.

4.9.2 Αποτίμηση σε ένα κόμβο

Δεδομένου ότι έχουμε ένα πρόβλημα μεγιστοποίησης , η αποτίμηση σε ένα κόμβο s θα είναι ένα πάνω φράγμα (upper bound) της μέγιστης τιμής των λύσεων μέσα σε αυτό τον κόμβο. Η αποτίμηση στηρίζεται στο χαλαρωμένο γραμμικό πρόγραμμα, δηλαδή οι μεταβλητές x_i μπορούν να πάρουν όλες τιμές στο $[0,1]$. Το χαλαρωμένο αυτό γραμμικό πρόγραμμα επιλύεται εύκολα σε $O(n \log n)$ από το γνωστό άπληστο αλγόριθμο όταν τα

Θέση μετά την ταξινόμηση	Δείκτης αντικειμένου	c_i	a_i	Σχετική αξία c_i/a_i
1	3	30	15	2
2	1	27	18	1.5
3	4	16	16	1
4	2	9	12	0.75
5	5	6.5	13	0.5

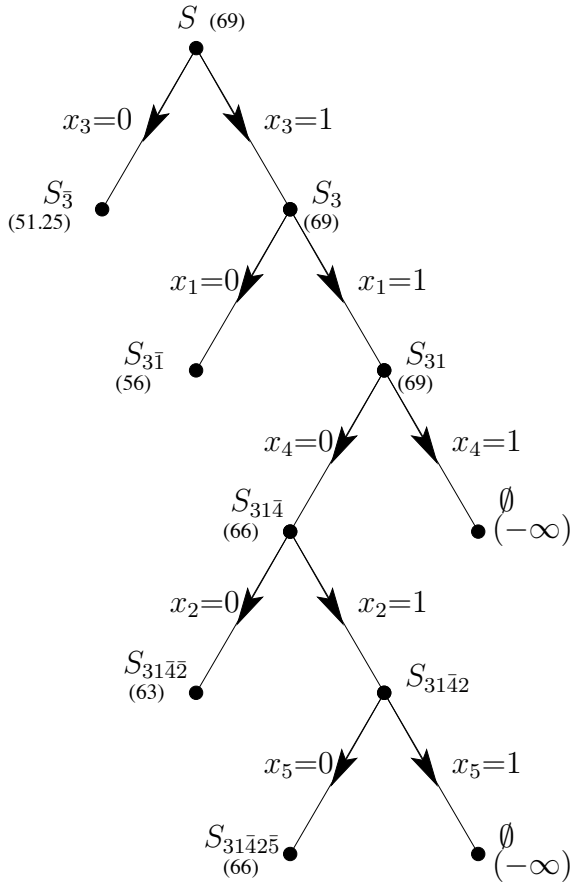
Σχήμα 4.8: Περιγραφή ενός στιγμιοτύπου του Knapsack μετά την ταξινόμηση και σχετική αξία των αντικειμένων

αντικείμενα είναι ταξινομημένα κατά μη αύξουσα σειρά της σχετικής τους αξίας c_i/a_i. Στο σχήμα 4.8 έχουμε την περιγραφή του στιγμιοτύπου και τις σχετικές αξίες των αντικειμένων.

Ο άπληστος αλγόριθμος θα δώσει ένα πάνω φράγμα 69 έχοντας επιλέξει $x_3 = 1, x_1 = 1, x_4 = 12/16 = 3/4$. Η τιμή αυτή είναι η αποτίμηση στη ρίζα του δέντρου της Branch and Bound. Σε ένα ενδιάμεσο κόμβο του δέντρου επιλύουμε το εναπομένον πρόβλημα με τα ελεύθερα ακόμη αντικείμενα και την εναπομείνουσα ακόμη χωρητικότητα του σάκκου και προσθέτουμε την τιμή που βρίσκουμε σε αυτήν των ήδη επιλεγμένων αντικειμένων. Θα θεωρήσουμε ότι δε διαθέτουμε αρχική λύση και επομένως στη ρίζα του δέντρου η τρέχουσα λύση έχει τιμή $-\infty$.

4.9.3 Επιλογή του κόμβου για διαχώριση

Ας επιλέξουμε τη στρατηγική την ''αρχικά η καλύτερη'' δηλαδή τον κόμβο με τη μέγιστη αποτίμηση. Το δέντρο που αναπτύσσεται φαίνεται στο Σχήμα 4.9. Ας παρατηρήσουμε ότι ένας κόμβος όπως το δεξιό παιδί του s_{31}, που παραβιάζει τον περιορισμό της χωρητικότητας, μπορεί να διαγραφεί αμέσως (και θέτουμε αποτίμηση $-\infty$). Ο κόμβος $s_{31\bar{4}2\bar{5}}$ δίνει την πρώτη λύση με τιμή 66 έχοντας επιλέξει τα αντικείμενα 1, 2 και 3 και ο σάκκος είναι γεμάτος. Εδώ τελειώνει και η αναζήτηση αφού τα φύλλα σε αναμονή (κόμβοι s_5, $s_{3\bar{1}}$ και $s_{31\bar{4}\bar{2}}$) είναι μικρότερης αποτίμησης και πρέπει να διαγραφούν.



Σχήμα 4.9: Το δέντρο αναζήτησης για το δεύτερο παράδειγμα. Η στρατηγική διάσχισης είναι ‘‘αρχικά η καλύτερη’’. Η διαχώριση γίνεται στη μεταβλητή με μέγιστη σχετική αξία c_i/a_i στον εκάστοτε κόμβο του δέντρου. Οι κόμβοι με αποτίμηση $-\infty$ έχουν χώρο λύσεων κενό.

4.10 Αμιλτονιακό Μονοπάτι (Hamiltonial Path)

Ας δούμε ένα άλλο παράδειγμα ελαχιστοποίησης με την Branch and Bound μέθοδο όπου το δέντρο αναζήτησης (search tree) δεν είναι δυαδικό.

Έστω ένας βεβαρημένος γράφος G(V,E) με μη κατευθυνόμενες ακμές $|V| = n$. Θεωρούμε ένα ελάχιστο δέντρο δέντρο επικάλυψης T στον G. Για κάθε κόμβο i συμβολίζουμε σαν d_i^T τον βαθμό του στο T.

Ορίζουμε την ποσότητα

$$E_T = \sum_{i=1, d_i^T > 2}^{n} \left(d_i^T - 2\right)$$

ως την εγγύτητα του T σε Hamiltonian μονοπάτι.

Προφανώς, Για ένα Hamiltonian μονοπάτι H θα ισχύει $E_H = 0$.

Θεωρούμε ως πρόβλημα να βρούμε ένα ελάχιστο δένδρο επικάλυψης T χωρίς κανένας κόμβος i να έχει βαθμό $d_i^T > 2$, $d_i \neq 0 \Rightarrow d_i = \begin{cases} 1 \\ 2 \end{cases}$.

Θεωρούμε πως υπάρχουν q ακμές βαθμού 1 και n-q ακμές βαθμού 2. Έστω m_T οι ακμές του δέντρου επικάλυψης T. Θα είναι:

$$m_T = \frac{1}{2} \sum_{i=1}^{n} d_i^T = \frac{1}{2} \left[q + 2\left(n - q\right)\right] = n - \frac{q}{2}.$$

Αφού όμως το T είναι δέντρο με n κόμβους θα είναι

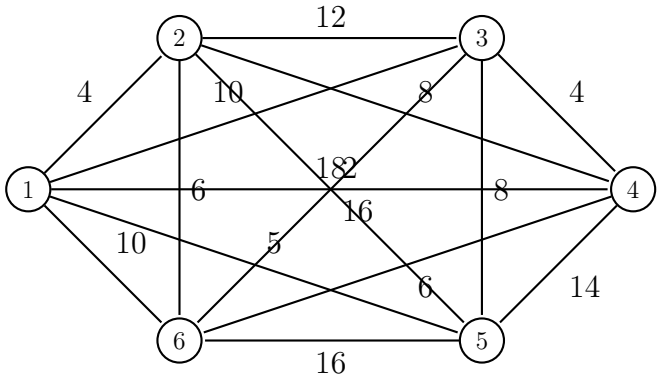
$$\left.\begin{array}{l} m_T = n - 1 \\ m_T = n - \frac{q}{2} \end{array}\right\} \Rightarrow n - 1 = n - \frac{q}{2} \Rightarrow q = 2.$$

Έστω γράφος G(V,E,C)

$C =$

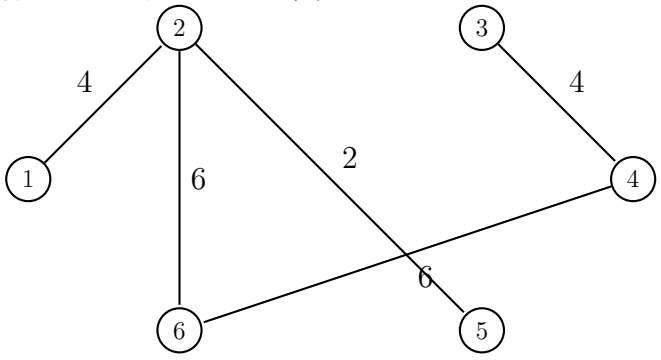
	1	2	3	4	5	6
1	0	4	10	18	5	10
2	4	0	12	8	2	6
3	10	12	0	4	8	16
4	18	8	4	0	14	6
5	5	2	8	14	0	16
6	10	6	16	6	16	0

Ο παραπάνω γράφος έχει την ακόλουθη μορφή:



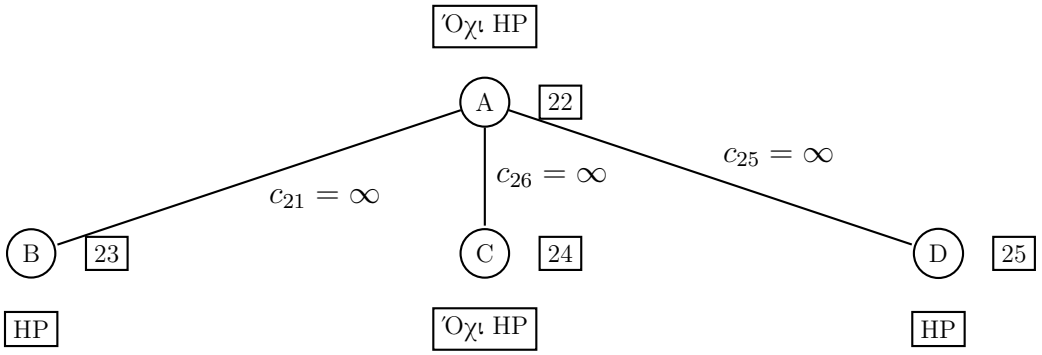
Θέλουμε τώρα να βρούμε το ελάχιστο δέντρο επικάλυψης (με την βοήθεια κάποιου αλγορίθμου εύρεσης δέντρου επικάλυψης ελαχίστου κόστους, όπως ο αλγόριθμος του Prim ή του Kruskal). Αν υπάρχει κόμβος με βαθμό $d > 2$ πρέπει να αφαιρέσουμε τουλάχιστον $d - 2$ ακμές. Θα εφαρμόσουμε τον αλγόριθμο Branch and Bound πάνω στις μεταβλητές απόφασης των ακμών για την εύρεση Hamiltonian μονοπατιού.

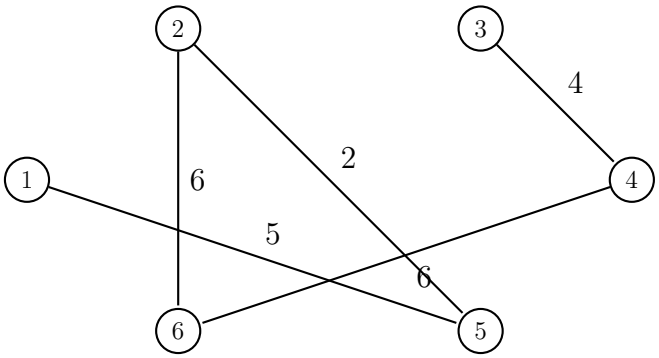
Ας υποθέσουμε ότι με κάποιον αλγόριθμο της επιλογής μας, βρήκαμε το ακόλουθο ελάχιστο δέντρο επικάλυψης:



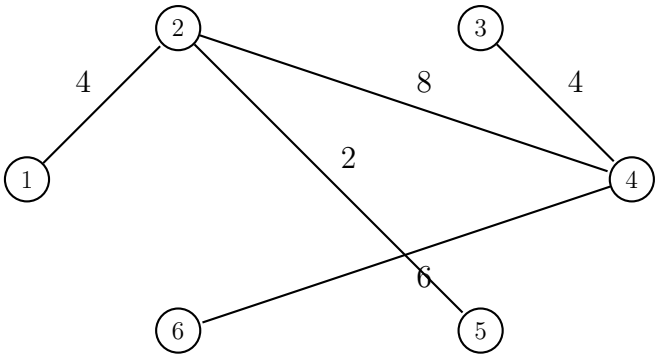
Παρατηρούμε ότι $T^*(A) = 22$ και $d_2^T = 3$, επομένως μια από τις ακμές $[2,1], [2,5], [2,6]$ πρέπει να διαγραφεί.

Κατά την εκτέλεση του αλγορίθμου Branch and Bound δημιουργείται το ακόλουθο δέντρο:

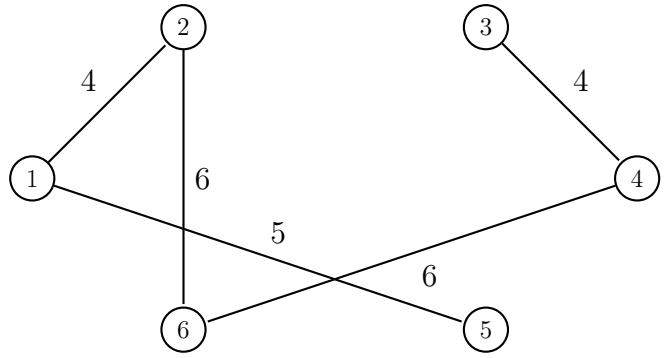




Βλέπουμε ότι $T^*(B) = 23$.



Βλέπουμε ότι $T^*(C) = 24$.



Βλέπουμε ότι $T^*(D) = 25$.

Από τους 3 κλάδους του δέντρου της Branch and Bound βλέπουμε ότι οι 2 οδηγούν σε Hamiltonian μονοπάτι και άρα πλησιάζουμε στην τελική λύση.

Κεφάλαιο 5

Προβλήματα ακέραιου προγραμματισμού

Τα περισσότερα προβλήματα συνδυαστικής βελτιστοποίησης μπορούν να γραφούν σαν προβλήματα ακέραιου προγραμματισμού. Ας δούμε μερικά παραδείγματα.

5.1 Πρόβλημα Παραγωγής

Έστω ένα πρόβλημα παραγωγής n προϊόντων όπου το κόστος παραγωγής $F_j(x_j)$ ενός προϊόντος j συνίσταται σε

- ένα κόστος καθορισμένο d_j ανεξάρτητο από την ποσότητα παραγωγής x_j.
- ένα κόστος c_j ανά παραγόμενη μονάδα.

Έχουμε λοιπόν:

$$F_j(x_j) = \begin{cases} d_j + x_j c_j & \text{αν } x_j > 0 \\ 0 & \text{αν } x_j = 0 \end{cases}$$

Αναζητούμε να ελαχιστοποιήσουμε το συνολικό κόστος παραγωγής: $\min \sum_{j=1}^{n} F_j(x_j)$

Εισάγοντας τις δυαδικές μεταβλητές y_j με

$$y_j = 0 \text{ αν } x_j = 0$$
$$y_j = 1 \text{ αν } x_j > 0$$

το πρόβλημα γράφεται:

$$\begin{aligned}
&\min z = \sum_{j=1}^{n}(c_j x_j + d_j y_j) \\
&x_j(1 - y_j) = 0 \\
&x_j \geq 0 \\
&y_j = 0 \text{ ή } 1
\end{aligned}$$

Αν ένα φράγμα (πεπερασμένο) M_j για το x_j είναι γνωστό, μπορούμε να γράψουμε το πρόβλημα:

$$\begin{aligned}
&\min z = \sum_{j=1}^{n}(c_j x_j + d_j y_j) \\
&x_j \leq M_j y_j \\
&y_j = 0 \text{ ή } 1 \\
&x_j \geq 0
\end{aligned}$$

Μια ειδική περίπτωση είναι το παρακάτω πρόβλημα:

5.2 Εγκατάσταση αποθηκών

Πρόκειται να επιλέξουμε τον τόπο εγκατάστασης αποθηκών για εμπορεύματα, με ελάχιστο κόστος, και ταυτόχρονα να ικανοποιήσουμε τις απαιτήσεις των πελατών μας. Γνωρίζουμε:

- τις ανάγκες b_j σε εμπορεύματα κάθε πελάτη $j = 1, 2, \ldots, n$
- το καθορισμένο κόστος εγκατάστασης d_i μιας αποθήκης στην περιοχή $i = 1, 2, \ldots, m$
- το κόστος μεταφοράς c_{ij} ανά μονάδα εμπορεύματος από την αποθήκη i προς τον πελάτη j.
- τη χωρητικότητα a_i της αποθήκης i.

Έχουμε λοιπόν:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} d_i y_i$$
$$\sum_{i=1}^{m} x_{ij} = b_j, \quad \forall j = 1, \ldots, n$$
$$\sum_{j=1}^{n} x_{ij} \leq a_i y_i, \quad \forall i = 1, \ldots, m$$
$$x_{ij} \geq 0$$
$$y_i = 0 \text{ ή } 1$$

Προφανώς έχουμε:

$$\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j$$
$$y_i = \begin{cases} 1 \text{ αν η αποθήκη } i \text{ ανοίγει} \\ 0 \text{ διαφορετικά} \end{cases}$$

5.3 Πρόβλημα σακιδίου (knapsack)

Ένας ορειβάτης ετοιμάζεται να γεμίσει το σακίδιό του. Δεδομένου του βάρους του απαραίτητου εξοπλισμού, το απομένον βάρος στο σακίδιό του για τη διατροφή του δεν θα πρέπει να ξεπεράσει $w = 11$ κιλά. A priori διαθέτει 5 τύπους αντικειμένων των οποίων τα χαρακτηριστικά περιγράφονται στον παρακάτω πίνακα:

Τύπος j	Αριθμός διαθέσιμων αντικ. n_j	Βάρος ενός αντικειμένου w_j	Αξία ενός αντικειμένου E_j
1	3	4	10
2	2	3	9
3	2	3	9
4	2	3	8
5	2	2	7

Για ένα τύπο αντικειμένου, η μοναδιαία τιμή E_j αντιπροσωπεύει το επίπεδο χρησιμότητας του αντικειμένου του τύπου j. Αν λοιπόν, ο ορειβάτης αποφασίσει να πάρει y_j αντικείμενα του τύπου $j (j = 1, 2, \ldots, 5)$, η αξία του σακιδίου θα είναι:

$$v = \sum_{j=1}^{5} E_j y_j$$

Στη γενική περίπτωση έχουμε το πρόβλημα:

$$\begin{aligned}
&\max u = \sum_{j=1}^{n} E_j y_j \\
&\sum_{j=1}^{n} w_j y_j \leq w \\
&y_j \leq n_j \quad (j = 1, \ldots, n) \\
&y_j \in \mathbb{N}
\end{aligned}$$

Ο στόχος του ορειβάτη μας είναι να καθορίσει τις ποσότητες $y_j (j = 1, 2, \ldots, 5)$ που μεγιστοποιούν τη συνάρτηση v. Έχουμε λοιπόν ένα πρόβλημα ακέραιου γραμμικού προγραμματισμού το οποίο γράφεται για το παραπάνω παράδειγμα:

$$\begin{aligned}
&\max 10y_1 + 9y_2 + 9y_3 + 8y_4 + 7y_5 \\
&4y_1 + 3y_2 + 3y_3 + 3y_4 + 2y_5 \leq 11 \\
&y_1 \leq 3 \\
&y_2 \leq 2 \\
&y_3 \leq 2 \\
&y_4 \leq 2 \\
&y_5 \leq 2 \\
&y_j \in \mathbb{N} \quad j = 1, \ldots, 5
\end{aligned}$$

5.4 Το μονοδιάστατο πρόβλημα κοπής (cutting stock problem)

Ένας προμηθευτής έχει στις αποθήκες p τύπους από ορθογώνιες μεταλλικές πλάκες (p διαφορετικά μήκη $L_1, L_2, L_3, \ldots, L_p$, τα πλάτη είναι όλα ίδια). Σε κάθε πλάκα ενός τύπου $k \in \{1, \ldots, p\}$ αντιστοιχεί ένα κόστος c_k περιέχον μεταξύ άλλων την κατασκευή της πλάκας, τη μεταφορά και την αποθήκευση.

Οι παραγγελίες με ίδια χαρακτηριστικά, από τους διάφορους πελάτες, είναι ομαδοποιημένες σε m κέντρα παραγγελιών, που αντιστοιχούν σε d_i πλάκες με μήκος $l_i (i = 1, 2, \ldots, m)$, οι οποίες θα πρέπει να κοπούν από τις πλάκες που υπάρχουν αποθηκευμένες.

Υποθέτοντας ότι το κόστος ετοιμασίας των $\sum_{i=1}^{m} d_i$ πλακών είναι ίσο με το άθροισμα

του κόστους των χρησιμοποιηθέντων αποθηκευμένων πλακών (δηλαδή τα περισσεύματα από την κοπή κομμάτια μιας αποθηκευμένης πλάκας που δεν αντιστοιχούν στην παραγγελία ενός πελάτη χαρακτηρίζονται σαν άχρηστα), το πρόβλημα κοπής σε μια διάσταση είναι η ικανοποίηση των m παραγγελιών με το ελάχιστο κόστος.

5.4.1 Μοντελοποίηση του προβλήματος

Ο πίνακας A των περιορισμών κατασκευάζεται ως εξής: για κάθε τύπο αποθηκευμένης πλάκας - μήκους $L_k, k \in \{1, \ldots, p\}$, προβλέπονται όλες οι δυνατότητες κοπής, που είναι ικανοποιητικές (δηλ. που καταλήγουν σε άχρηστη ποσότητα λιγότερη από ένα κατώφλι M δεδομένο) για να δημιουργήσουν με τον τρόπο που ακολουθεί $|J_k|$ κολόνες του A:

$$\forall j \in J_k \;\; 0 \leq L_k - \sum_{i=1}^{m} a_{ij} l_i \leq M$$

όπου a_{ij} εκφράζει το πλήθος παραγελιών μήκους l_i μέσα στη j δυνατότητα το οποίο λέγεται *μοντέλο κοπής* με

$$a_{ij} \in N, \; \forall i, \; \forall j$$

Το πρόβλημα κοπής γράφεται:

$$(CS) \begin{cases} \sum_{k=1}^{p} c_k \sum_{j \in J_k} x_j \\ \sum_{k=1}^{p} \sum_{j \in J_k} a_{ij} x_j = d_i, \;\; i = 1, \ldots, m \\ x_j \in N, \; \forall \, j \in \bigcup_{k=1}^{p} J_k \end{cases}$$

όπου x_j υποδεικνύει τον αριθμό χρησιμοποίησης του j-οστού μοντέλου κοπής (περιγραφόμενο από την κολόνα A^j) $\forall j \in \bigcup_{k=1}^{p} J_k$. Το πλήθος των κολόνων του A θα είναι $n = \sum_{k=1}^{p} |J_k|$.

5.4.2 Χρησιμοποίηση του προβλήματος σακιδίου

Ο πίνακας A στο παραπάνω πρόβλημα περιέχει ένα μεγάλο αριθμό κολόνων - συνηθισμένα μεγέθη προβλημάτων της τάξης $m = 20$ δίνουν $n = 24000$ (μικρό μέγεθος) και $m = 30$ δίνουν $n = 190000$ (για μέγεθος λογικό) - και επομένως θα εμφανιστούν

σημαντικά προβλήματα μνήμης στην επίλυση του (CS). Μπορούμε εν τούτοις, χρησιμοποιώντας την κλασική μέθοδο της Simplex και με τη βοήθεια της επίλυσης μιας σειράς από προβλήματα knapsack, να ξεπεράσουμε το εμπόδιο του μεγάλου αριθμού κολόνων.

Ας υπενθυμίσουμε ότι το πέρασμα από μια εφικτή βάση B σε μια γειτονική βάση στη Simplex πραγματοποιείται από την επίλυση δύο γραμμικών συστημάτων (Υπενθυμίζουμε ότι A^s συμβολίζει την κολόνα στη θέση s του πίνακα A):

i) εύρεση διανύσματος $y : yB = c_B$ (βλέπε Simplex, Παρ. 3.7.2 για την περίπτωση ελαχιστοποίησης).

ii) αν δεν υπάρχει j εκτός βάσης με $yA^j > c_j$ τότε η βάση B είναι βέλτιστη (τέλος)

διαφορετικά, έστω j κολόνα εκτός βάσης με $c_j < yA^j$

iii) επίλυση συστήματος $Bd = A^j$

αν $d \leq 0$ τότε (CS) έχει τιμή άπειρο (τέλος)

διαφορετικά καθόρισε την εξερχόμενη κολόνα r:

$x_r/d = \min\{x_i/d_i, d_i > 0\}$

iv) θέσε $B \equiv B - \{r\} + \{j\}$, υπολόγισε $Bx = b$ και πήγαινε στο (i).

Η φάση της αναζήτησης $j \notin B : yA^j > c_j$ (ο καθορισμός του βέλτιστου στο βήμα (ii) και η εύρεση της κολόνας j επιταχύνεται με την επίλυση μιας σειράς προβλημάτων knapsack (p το πολύ στην περίπτωση βέλτιστου) ως εξής:

Θεωρούμε ότι $k \in \{1, \ldots, p\}$ και ορίζουμε το ακόλουθο knapsack με $u = y$:

$$(B^k) \begin{cases} uz^k = \max u_1 z_1 + u_2 z_2 + \cdots + u_m z_m \\ l_1 z_1 + l_2 z_2 + \cdots + l_m z_m \leq L_k \\ z_j \in N,\ j = 1, \ldots, m \end{cases}$$

Το βήμα (ii) της μεθόδου Simplex γίνεται λοιπόν:

1. $k := 1$
2. αν $uz^k \leq c_k$ τότε συνέχισε στο βήμα 3

   διαφορετικά όρισε $j \in J : A^j = z^k$ και συνέχισε στο βήμα (iii) της Simplex.

3. αν $k < p$ τότε $k := k + 1$ και συνέχισε στο βήμα 2
διαφορετικά B είναι βέλτιστη βάση, ΤΕΛΟΣ.

Αυτή η μέθοδος είναι η *μέθοδος του γενικευμένου γραμμικού προγραμματισμού* και μας επιτρέπει να επιλύσουμε το πρόβλημα (CS) χωρίς να γνωρίζουμε πλήρως τον A και επομένως χωρίς να αποθηκεύσουμε ολόκληρο τον πίνακα A.

5.4.3 Παράδειγμα (Το μονοδιάστατο πρόβλημα κοπής)

Διαθέτουμε σε απόθεμα ράβδους μήκους 7,4 και 7,0 μονάδων, των οποίων τα κόστη είναι 7,3 και 7 νομισματικές μονάδες, αντίστοιχα.

Οι παραγγελίες πελατών συνίστανται σε 100 κομμάτια μήκους 2,9, σε 100 κομμάτια μήκους 2,1 και σε 100 κομμάτια μήκους 1,5.

Να βρεθεί το πλήθος των ράβδων οι οποίες πρέπει να κοπούν έτσι ώστε το κόστος να είναι ελάχιστο και επίσης να ικανοποιηθούν όλες οι παραγγελίες.

Για να επιλύσουμε το πρόβλημα αρχίζουμε από την κατασκευή μιας λίστας από διαφόρους τρόπους κοπής κομματιών μήκους $l_1 = 2{,}9$, $l_2 = 2{,}1$ και $l_3 = 1{,}5$ από τις ράβδους μήκους 7,4 και 7 έτσι ώστε τα ‘‘άχρηστα’’ (μη χρησιμοποιούμενα) κομμάτια να είναι μικρότερα από 1,5. Αναζητούμε λοιπόν τα διανύσματα (a_1, a_2, a_3) τέτοια ώστε:

$$2,9a_1 + 2,1a_2 + 1,5a_3 + b = L = 7,4 \text{ ή } 7 \text{ με } \ b < 1,5$$

Οι ράβδοι μπορούν να κοπούν π.χ. με τον ακόλουθο τρόπο:

$$\begin{array}{ll} 2l_1 + 0l_2 + 1l_3 = 7,3 & 1l_1 + 1l_2 + 1l_3 = 6,5 \\ 1l_1 + 2l_2 + 0l_3 = 7,1 & 0l_1 + 3l_2 + 0l_3 = 6,3 \\ 1l_1 + 0l_2 + 3l_3 = 7,4 & 0l_1 + 1l_2 + 3l_3 = 6,6 \\ 0l_1 + 2l_2 + 2l_3 = 7,2 & 0l_1 + 0l_2 + 4l_3 = 6,5 \end{array}$$

Ευρίσκουμε λοιπόν τον ακόλουθο πίνακα:

Μήκη παραγγελιών	δυνατές κοπές								Ποσότητες παραγγελιών
2,9	2	1	1	1	0	0	0	0	100
2,1	0	2	1	0	3	2	1	0	100
1,5	1	0	1	3	0	2	3	4	100
κόστη	7,3	7,3	7	7,3	7	7,3	7	7	

Έστω $x_i, i \in (1, \ldots, 8)$ το πλήθος χρησιμοποίησης του μοντέλου κοπής μιας ''μεγάλης'' ράβδου i, με $x_i \in \mathbb{N}$.

Αν $x_1 = 5, x_2 = 1, x_3 = x_4 = x_5 = x_6 = 0, x_7 = 1, x_8 = 0, x_8 = 0$

Θα έχουμε π.χ.

$$2 \times 5 + 1 \times 1 + 0 \times 1 = 11 \text{ κομμάτια μήκους } 2,9$$
$$0 \times 5 + 2 \times 1 + 1 \times 1 = 3 \text{ κομμάτια μήκους } 2,1$$
$$1 \times 5 + 0 \times 1 + 3 \times 1 = 8 \text{ κομμάτια μήκους } 1,5$$

Το πρόβλημα ελαχιστοποίησης γράφεται:

$$\begin{array}{cccccccccccccccc}
2x_1 & + & x_2 & + & x_3 & + & x_4 & & & & & & & & & \geq 100 \\
 & & 2x_2 & + & x_3 & & & + & 3x_5 & + & 2x_6 & + & x_7 & & & \geq 100 \\
x_1 & + & & & x_3 & + & 3x_4 & & & + & 2x_6 & + & 3x_7 & + & 4x_8 & \geq 100 \\
min 7,3x_1 & + & 7,3x_2 & + & 7x_3 & + & 7,3x_4 & + & 7x_5 & + & 7,3x_6 & + & 7x_7 & + & 7x_8 & \\
 & & & & & & x_i \in N & & & & & & & & &
\end{array}$$

5.5 Προβλήματα Επικάλυψης, Διαμέρισης και Πακεταρίσματος (Covering, Partitioning and Packing Problems)

Έστω ένα σύνολο $I = \{1, \ldots, m\}$ και $P = \{P_1, \ldots, P_n\}$ με $P_j \subseteq I$, $\forall j \in J = \{1, \ldots, n\}$ μία οικογένεια υποσυνόλων του I.

Μια επικάλυψη R του I είναι μία υποοικογένεια του P, με δείκτες $J^* \subseteq J$ έτσι ώστε:

$$\bigcup_{j \in J^*} P_j = I$$

Αν $\forall j,\ k \in J^*$ έχουμε $P_j \bigcap P_k = \emptyset$ τότε R είναι μια διαμέριση του I.

5.5.1 Επικάλυψη ελαχίστου κόστους (Set Cover - SC)

Έστω c_j το κόστος που αντιστοιχεί στο υποσύνολο P_j. Το πρόβλημα επικάλυψης ελαχίστου κόστους ως πρόβλημα ακέραιου προγραμματισμού γράφεται ως εξής:

$$(SC) \begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m \\ x_j = 0 \text{ ή } 1 \end{cases}$$

όπου $x_j = \begin{cases} 1 \text{ αν } P_j \text{ ανήκει στη λύση} \\ 0 \text{ διαφορετικά} \end{cases}$

και $a_{ij} = \begin{cases} 1 \text{ αν το στοιχείο } i \in P_j \\ 0 \text{ διαφορετικά} \end{cases}$

5.5.2 Διαμέριση ελαχίστου κόστους (Set Partitioning - SP)

Το πρόβλημα διαμέρισης ελαχίστου κόστους γράφεται ως πρόβλημα ακέραιου προγραμματισμού:

$$(SP) \begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j = 1, i = 1, \ldots, m \\ x_j \in \{0, 1\} \end{cases}$$

όπου $x_j = \begin{cases} 1 \text{ αν } P_j \text{ ανήκει στη λύση} \\ 0 \text{ διαφορετικά} \end{cases}$

και $a_{ij} = \begin{cases} 1 \text{ αν το στοιχείο } i \in P_j \\ 0 \text{ διαφορετικά} \end{cases}$

5.5.3 Πακετάρισμα μεγίστου κόστους (Set Packing Problem - SPP)

Το πρόβλημα πακεταρίσματος μεγίστου κόστους γράφεται:

$$(SPP)\begin{cases} \max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq 1, i = 1, \ldots, m \\ x_j \in \{0, 1\} \end{cases}$$

όπου $x_j = \begin{cases} 1 \text{ αν } P_j \text{ ανήκει στη λύση} \\ 0 \text{ διαφορετικά} \end{cases}$

και $a_{ij} = \begin{cases} 1 \text{ αν το στοιχείο } i \in P_j \\ 0 \text{ διαφορετικά} \end{cases}$

Σε όλα τα παραπάνω προβλήματα, αν $c_j = 1$ τότε έχουμε την ειδική περίπτωση ελαχιστοποίησης ή μεγιστοποίησης του πληθικού αριθμού της υποοικογένειας που επιλέγουμε (cardinality case).

5.6 Το edge disjoint path πρόβλημα

Έστω ένας γράφος $G(V, E)$ και ένα σύνολο T από αιτήσεις σύνδεσης (commodities). Κάθε αίτηση σύνδεσης στο T είναι ένα ζευγάρι κορυφών (s_i, t_i) , $1 \leq i \leq K$. Στο edge (vertex) - disjoint path στόχος είναι να συνδέσουμε τον μέγιστο αριθμό ζευγαριών (s_i, t_i) μέσω μονοπατιών που δεν χρησιμοποιούν κοινές ακμές (κορυφές) στον γράφο G. Μια γενικότερη εκδοχή του παραπάνω προβλήματος προκύπτει αν αναθέσουμε ένα βάρος w_i σε κάθε αίτηση σύνδεσης, οπότε αναζητούμε να δρομολογήσουμε έναν αριθμό αιτήσεων μεγίστου βάρους μέσω μονοπατιών που δεν χρησιμοποιούν κοινές ακμές (κορυφές). Επικεντρωνόμαστε στην περίπτωση του edge disjoint path (EDP) προβλήματος χωρίς βάρη στις αιτήσεις σύνδεσης (cardinality case).

Για κάθε αίτηση σύνδεσης διαθέτουμε έναν αριθμό μονοπατιών που μπορούν να ικανοποιήσουν αυτή την σύνδεση. Έστω P_i το σύνολο των μονοπατιών που μπορούν να ικανοποιήσουν την σύνδεση i και c_i το πλήθος των μονοπατιών σ΄ αυτό το σύνολο. Έστω P_i^j το j - οστό μονοπάτι της i - οστής αίτησης σύνδεσης (θεωρούμε κάποια

τυχαία αρίθμηση των μονοπατιών στο σύνολο P_i).

Για να μοντελοποιήσουμε το πρόβλημα ως πρόβλημα ακέραιου προγραμματισμού κατασκευάζουμε ένα πίνακα A ως εξής :

- Στις γραμμές του πίνακα A τοποθετούμε τις ακμές $(e_1, e_2, \ldots, e_m) \in E$ του γράφου G.

- Στις στήλες του πίνακα έχουμε όλα τα δυνατά μονοπάτια $P_i^j, \quad 1 \leq j \leq c_i, \quad 1 \leq i \leq K$.

- Θεωρούμε 1 σε μία θέση του πίνακα αν η αντίστοιχη ακμή περιέχεται στο αντίστοιχο μονοπάτι. Αν θέσουμε $\lambda = \sum_{k=1}^{i-1} c_k + j$ έχουμε:

$$A(\kappa, \lambda) = \begin{cases} 1 & \text{αν} \quad e_\kappa \in P_i^j \\ 0 & \text{αν} \quad e_\kappa \notin P_i^j \end{cases}$$

Ο αριθμός των γραμμών του πίνακα A είναι ίσος με το πλήθος των ακμών του γράφου ($|m| = |E|$) και ο αριθμός των στηλών είναι ίσος με το σύνολο των μονοπατιών μεταξύ όλων των αιτήσεων σύνδεσης ($|\cup_{i=1}^{K} P_i|$).

Σε κάθε μονοπάτι P_i^j αντιστοιχούμε μια δυαδική μεταβλητή $x_i^j = \{0, 1\}$.Η μεταβλητή αυτή είναι ίση με ένα αν το μονοπάτι στο οποίο αντιστοιχεί (το j - οστό μονοπάτι της i - οστής αίτηση σύνδεσης) επιλέγεται στην τελική λύση και 0 αλλιώς.

$$x_i^j = \begin{cases} 1 & \text{αν το} \quad P_i^j \quad \text{επιλεχθεί στη λύση} \\ 0 & \text{αν το} \quad P_i^j \quad \text{δεν επιλεχθεί στη λύση} \end{cases}$$

Επειδή θέλουμε τα μονοπάτια που θα επιλεχθούν στην τελική λύση να μην χρησιμοποιούν κοινές ακμές στον γράφο G απαιτούμε το άθροισμα των στοιχείων κάθε γραμμής του πίνακα A να είναι μικρότερο είτε ίσο με ένα. Επίσης απαιτείται η χρήση ενός περιορισμού που να μας εμποδίζει να επιλέξουμε παραπάνω από ένα μονοπάτια για κάποια αίτηση σύνδεσης. Έτσι το edge disjoint path μοντελοποιείται ως εξής :

$$\text{maximize} \quad \sum_{i=1}^{K}\sum_{j=1}^{c_i} x_i^j, \qquad \text{subject to}$$

$$A \cdot x \quad \leq \quad 1 \tag{5.1}$$

$$\sum_{j=1}^{c_i} x_i^j \quad \leq 1 \qquad \forall 1 \leq i \leq K \tag{5.2}$$

$$x_i^j = \{0,1\} \tag{5.3}$$

όπου το x είναι διάνυσμα διάστασης $|P| = |\cup_{i=1}^{K} P_i|$ περιέχων τις μεταβλητές $x_i^j = \{0,1\}$. Ο περιορισμός 5.1 εξασφαλίζει τη μη χρησιμοποίηση κοινών ακμών ενώ ο περιορισμός 5.2 εξασφαλίζει ότι θα επιλεχθεί ένα μόνο μονοπάτι για κάθε αίτηση σύνδεσης.

Σ΄ αυτό το σημείο αξίζει να σημειώσουμε ότι στην ειδική εκδοχή του προβλήματος όπου μας δίνεται για κάθε αίτηση σύνδεσης ένα μοναδικό μονοπάτι που ικανοποιεί τη σύνδεση αυτή το ακέραιο πρόγραμμα γίνεται:

$$\text{maximize} \quad \sum_{i=1}^{K} x_i, \qquad \text{subject to}$$

$$A \cdot x \quad \leq \quad 1 \tag{5.4}$$

$$x_i = \{0,1\} \tag{5.5}$$

όπου οι μεταβλητές γράφονται πιο απλά $x_i = 1$ (αν P_i επιλέγεται στη λύση) και $x_i = 0$ διαφορετικά επειδή $c_i = 1, 1 \leq i \leq K$. Ο πίνακας A είναι με στοιχεία $a(i,j) = 1$ αν $e_i \in P_i$ και $a(i,j) = 0$ αν $e_i \notin P_i$.

5.7 Το πρόβλημα του περιοδεύοντος πωλητή (Traveling Salesman Problem)

Ένας πωλητής πρέπει να επισκεφθεί $n+1$ πόλεις αριθμημένες $0, 1, \ldots, n$ περνώντας μία φορά και μόνο μία από κάθε πόλη πριν να επιστρέψει στην αρχική πόλη αναχώρησης.

Με ποια σειρά πρέπει να επισκεφθεί ο πωλητής τις πόλεις για να ελαχιστοποιήσει

τη διανυθείσα διαδρομή;

Στη θεωρία γράφων, το πρόβλημα είναι γνωστό ως αναζήτηση ενός κύκλου Hamilton με το μικρότερο βάρος σε ένα γράφο, του οποίου οι κόμβοι είναι οι πόλεις και του οποίου οι πλευρές έχουν βάρη ίσα με τις αποστάσεις ανάμεσα στις συνδεόμενες πόλεις.

5.7.1 Μοντελοποίηση σαν πρόβλημα μικτού ακέραιου γραμμικού προγραμματισμού.

Έστω $0, 1, \ldots, n$ οι πόλεις και $[c_{ij}]$ οι αποστάσεις μεταξύ δύο πόλεων i και j (ο πίνακας δεν είναι αναγκαστικά συμμετρικός).

έστω $x_{ij} = \begin{cases} 1 \text{ αν η πλευρά } (i,j) \text{ είναι στον κύκλο (λύση)} \\ 0 \text{ διαφορετικά} \end{cases}$

Τότε μπορούμε να γράψουμε το πρόβλημα ως εξής:

$$\min z = \sum_{\substack{i,j=0 \\ i \neq j}}^{n} c_{ij} x_{ij}$$

$$x_{ij} \in \{0,1\}, \ \forall i, j$$

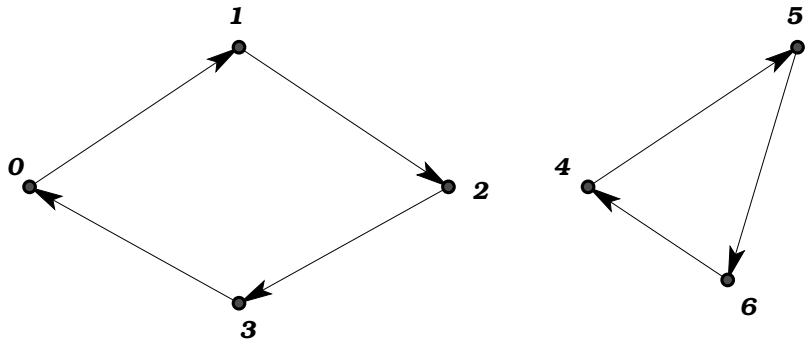
$$\text{(a)} \sum_{i=1}^{m} x_{ij} = 1, j = 0, \ldots, n$$

$$\text{(b)} \sum_{j=1}^{n} x_{ij} = 1, i = 1, \ldots, n$$

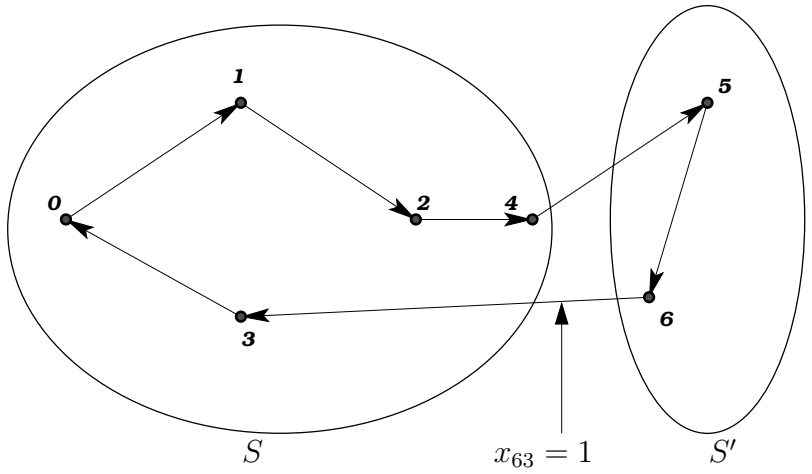
Οι ανισότητες στο (a) εκφράζουν το γεγονός ότι ακριβώς ένα μόνο τόξο εισέρχεται σε κάθε κόμβο και οι ανισότητες (b) ότι ακριβώς ένα τόξο εγκαταλείπει κάθε ένα από τους κόμβους $1, \ldots, n$ (προφανώς και τον κόμβο 0). Μπορούμε να δούμε όμως ότι οι (a) και (b) δεν δίνουν πάντα εφικτές λύσεις για το πρόβλημα. Πράγματι, ξένοι κύκλοι (ή υπό-κύκλοι) όπως στο παράδειγμά μας, εκφράζονται επίσης από τους περιορισμούς (a) και (b) (σχήμα 5.1).

Οι δύο υπό-κύκλοι είναι εφικτή λύση για το παραπάνω γραμμικό πρόγραμμα αλλά δεν είναι εφικτή λύση για το πρόβλημά μας. Άρα, η δυσκολία του προβλήματός μας δεν εμπεριέχεται πλήρως στο γραμμικό πρόγραμμα που γράψαμε. Μία πρώτη λύση είναι να προσθέσουμε στο γραμμικό σύστημα τους εξής περιορισμούς:

Ας θεωρήσουμε μια μη εκφυλισμένη διαμέριση $(S, \bar{S})$ των πόλεων $0, 1, \ldots, n$, και για κάθε τέτοια διαμέριση απαιτούμε ότι $\sum_{\substack{i \in S \\ j \in \bar{S}}} x_{ij} \geq 1 (*)$.



Σχήμα 5.1: Οι $2n + 1$ περιορισμοί (a) και (b) δεν επαρκούν για να έχουμε εφικτές λύσεις για το πρόβλημα του περιοδεύοντος πωλητή.



Σχήμα 5.2: Επιπλέον περιορισμοί για κάθε μη εκφυλισμένη διαμέριση επιτρέπουν να εξασφαλίσουμε εφικτές λύσεις στο πρόβλημα του περιοδεύοντος πωλητή.

Προσθέτοντας λοιπόν όλους τους περιορισμούς, έναν για κάθε μη εκφυλισμένη διαμέριση, στο γραμμικό μας σύστημα, μπορούμε να εξαλείψουμε τους υπό-κύκλους και να έχουμε μόνο εφικτές λύσεις που αντιστοιχούν σε κύκλους (σχήμα 5.2).

Υπάρχει όμως ένα πρόβλημα με αυτή τη λύση: το πλήθος των περιορισμών που προσθέτουμε. Βρείτε, λοιπόν αυτό το πλήθος! (και διαπιστώστε ότι αυτή η μέθοδος δεν είναι ενδιαφέρουσα)!

Μια καλύτερη λύση συνίσταται στο να αντικαταστήσουμε τους περιορισμούς που εξαλείφουν τους υπό-κύκλους (∗) από τους περιορισμούς:

$$u_i - u_j + nx_{ij} \leq n-1, \quad 1 \leq i \neq j \leq n (**)$$

όπου $u_i, i = 1, \ldots, n$ είναι πραγματικές μεταβλητές.

Θα δείξουμε στη συνέχεια ότι αυτές οι ανισότητες όχι μόνο περιορίζουν τις λύσεις σε κύκλους, αλλά ακόμη δεν εξαιρούν κανένα κύκλο.

Πράγματι, ας αποδείξουμε κατ΄ αρχήν ότι κάθε εφικτή λύση είναι ένας κύκλος. Για να δούμε αυτό, δείχνουμε ότι κάθε κύκλος περνά από την πόλη 0.

Ας υποθέσουμε ότι υπάρχει κύκλος που δε διασχίζει την πόλη 0. Ας υποθέσουμε ότι η σειρά $i_1, \ldots, i_k$ των πόλεων είναι ένας κύκλος που εξαιρεί την πόλη 0.

Μπορούμε να γράψουμε τους περιορισμούς $(**)$ κατά μήκος του κύκλου:

$$\begin{aligned} &u_{i_j} - u_{i_{j+1}} + n \leq n-1, \quad j = 1, \ldots, k-1 \\ &u_{i_k} - u_{i_1} + n \leq n-1 \end{aligned}$$

Προσθέτοντας τις τελευταίες σχέσεις, οδηγούμαστε σε μια αντίφαση (ποια είναι;)

Ας αποδείξουμε τώρα, ότι για κάθε νόμιμο κύκλο υπάρχουν τιμές των u_i που ικανοποιούν τις σχέσεις $(**)$.

Πράγματι, έστω $u_0 = 0$ και $u_i = t$ αν η πόλη i είναι στη θέση t του κύκλου, $t = 1, \ldots, n$.

Αν $x_{ij} = 0$, έχουμε:

$$u_i - u_j \leq n-1, \quad 1 \leq i \neq j \leq n$$

το οποίο είναι πάντα αληθές. Η περίπτωση $u_i = n$ και $u_j = 0$ εξαιρείται αφού το τόξο $(i, 0)$ είναι στον κύκλο και επομένως $x_{i0} = 1$.

Αν $x_{ij} = 1$, έχουμε

$$u_i - u_j + n \leq n-1, \quad 1 \leq i \neq j \leq n$$

το οποίο αληθεύει επειδή $u_i - u_j = -1$ αν i και j είναι διαδοχικές πόλεις στον κύκλο. Βέβαια η περίπτωση $u_i = n$ και $u_j = 0$ εξαιρείται από τη σχέση $(**)$.

Η παραπάνω μοντελοποίηση εμπεριέχει μεταβλητές x_{ij} οι οποίες περιορίζονται να είναι ακέραιες, και μεταβλητές u_i οι οποίες δεν είναι ακέραιες. Ένα τέτοιο πρόβλημα καλείται *Μικτό Ακέραιο Γραμμικό πρόβλημα* (mixed integer linear programming

problem).

5.8 Ροές πολλαπλών αγαθών (Multicommodity Flows)

Το πρόβλημα των ροών πολλαπλών αγαθών (multicommodity flows) εμφανίζεται σε πολλές εφαρμογές όταν, διαφορετικά φυσικά αγαθά (commodities) (π.χ οχήματα ή μηνύματα) το καθένα υποκείμενο στους δικούς του περιορισμούς ροής, μοιράζονται το δίκτυο. Για παράδειγμα σε ένα τηλεφωνικό δίκτυο, οι κλήσεις μεταξύ συγκεκριμένων ζευγαριών κόμβων ορίζουν διαφορετικές ευκολίες - αγαθά. Αν τα αγαθά δεν αλληλεπιδρούν με κανένα τρόπο μεταξύ τους, τότε το πρόβλημα ανάγεται στην επίλυση του single commodity flow [1] (ροή ενός αγαθού) προβλήματος για κάθε ένα αγαθό ξεχωριστά. Ωστόσο όταν τα αγαθά μοιράζονται κοινούς πόρους (facilities) του δικτύου, τότε για να βρούμε βέλτιστη ροή πρέπει να λύσουμε το πρόβλημα λαμβάνοντας υπ΄ όψιν τους περιορισμούς που προκύπτουν από την χρήση των διαμοιραζόμενων πόρων (facilities). Στη συνέχεια μελετάμε τις διάφορες μορφές των multicommodity flow προβλημάτων.

5.8.1 Πρόβλημα ροής πολλαπλών αγαθών ελαχίστου κόστους (Minimum Cost multicommodity flow)

Έστω ένα δίκτυο $G = (N, A)$ όπου N είναι το σύνολο κόμβων και E το σύνολο των κατευθυνόμενων ακμών. Στο μοντέλο που μελετάμε τα διαφορετικά αγαθά μοιράζονται κοινές χωρητικότητες στις κατευθυνόμενες πλευρές. Κάθε ακμή έχει μια χωρητικότητα u_{ij} που περιορίζει την συνολική ροή όλων των αγαθών σ΄ αυτή την ακμή. Έστω x_{ij}^k η ροή του αγαθού k στην ακμή (i, j), x^k το διάνυσμα ροής και c^k το διάνυσμα κόστους ανά μονάδα ροής για το αγαθό $k_j, 1 \leq k \leq K$. Χρησιμοποιώντας αυτό τον συμβολισμό το multicommodity flow μοντελοποιείται ως εξής :

[1] Στο single commodity flow δεδομένου ενός γράφου, με χωρητικότητες στις ακμές, αναζητούμε να μεγιστοποιήσουμε ή να ελαχιστοποιήσουμε την ροή ενός αγαθού από ένα κόμβο πηγή (source) s σε ένα κόμβο προορισμό (sinc) t χωρίς να παραβιάσουμε τις χωρητικότητες των ακμών

$$\text{Minimize} \sum_{1\le k\le K} c^k x^k \qquad \text{subject to} \tag{5.6}$$

$$\sum_{1\le k\le K} x_{ij}^k \le u_{ij}, \qquad \forall (i,j)\in A, \quad x_{ij}^k \in \mathbb{R} \tag{5.7}$$

$$\mathcal{N} x^k = b^k, \qquad \text{για} \quad k = 1,2,\ldots,K \tag{5.8}$$

$$0 \le x_{ij}^k \le u_{ij}^k, \qquad \forall (i,j)\in A \quad ,\forall \quad k = 1,2,\ldots,K \tag{5.9}$$

5.8.2 Κατανάλωση Χωρητικότητας μεγαλύτερης της μονάδας, από τα αγαθά

Στο min cost multicommodity flow υποθέσαμε ότι κάθε μονάδα ροής κάθε αγαθού (commoditie) χρησιμοποιεί μία μονάδα χωρητικότητας σε κάθε τόξο από το οποίο διέρχεται. Καταργώντας αυτή την υπόθεση προκύπτει ένα γενικότερο μοντέλο, το οποίο επιτρέπει στο αγαθό k να καταναλώνει ένα δοσμένο ποσό ρ_{ij}^k της χωρητικότητας κάθε κατευθυνόμενης ακμής. Τότε αλλάζοντας τον περιορισμό 5.7 το πρόβλημα γίνεται:

$$\text{Minimize} \sum_{1\le k\le K} c^k x^k \qquad \text{subject to} \tag{5.10}$$

$$\sum_{1\le k\le K} \rho_{ij}^k x_{ij}^k \le u_{ij} \qquad \forall (i,j)\in A, \quad x_{ij}^k \in \mathbb{R} \tag{5.11}$$

$$\mathcal{N} x^k = b^k, \qquad \text{για} \quad k = 1,2,\ldots,K \tag{5.12}$$

$$0 \le x_{ij}^k \le u_{ij}^k, \qquad \forall (i,j)\in A \quad ,\forall \quad k = 1,2,\ldots,K \tag{5.13}$$

5.8.3 Μεταβλητά κόστη στις ακμές - Μη γραμμική συνάρτηση κόστους

Επίσης υποθέσαμε παραπάνω ότι έχουμε μια σταθερή χωρητικότητα σε κάθε ακμή και ότι το κόστος σε κάθε τόξο είναι γραμμικό. Σε κάποιες εφαρμογές τα αγαθά αλληλεπιδρούν με πιο περίπλοκο τρόπο, δηλαδή όσο η ροή (οποιουδήποτε αγαθού) αυξάνει σε μία κατευθυνόμενη ακμή επιφέρει ένα μη γραμμικά αυξανόμενο κόστος σ΄

αυτό το τόξο[2]. Ένα congestion model multicommodity flow μπορεί να περιέχει τους περιορισμούς 5.11 και 5.12 και μία μη γραμμική συνάρτηση κόστους, όπως φαίνεται παρακάτω :

$$\text{Minimize} \sum_{(i,j)\in A} \frac{x_{ij}}{u_{ij} - x_{ij}} \qquad \text{subject to} \tag{5.14}$$

$$\sum_{1\le k\le K} \rho_{ij}^k x_{ij}^k \le u_{ij} \qquad \forall (i,j) \in A, \quad x_{ij}^k \in \mathbb{R} \tag{5.15}$$

$$\mathcal{N} x^k = b^k, \qquad \text{για} \quad k = 1, 2, \ldots, K \tag{5.16}$$

$$0 \le x_{ij}^k \le u_{ij}^k, \qquad \forall (i,j) \in A \quad , \forall \quad k = 1, 2, \ldots, K \tag{5.17}$$

Σ΄ αυτό το μοντέλο u_{ij} είναι η μέγιστη χωρητικότητα της ακμής (i, j). Όσο η συνολική ροή $x_{ij} = \sum_k x_{ij}^k$ σε ένα τόξο πλησιάζει την μέγιστη χωρητικότητα της ακμής, η καθυστέρηση πλησιάζει $+\infty$.

5.8.4 Ακέραιο multicommodity flow

Τα μοντέλα multicommodity flow προβλημάτων που έχουμε παρουσιάσει ως τώρα χρησιμοποιούν πραγματικές τιμές ($x_{ij}^k \in \mathbb{R}$) στις μεταβλητές ροής. Σε αρκετές εφαρμογές απαιτούνται ακέραιες μεταβλητές ροής και έτσι ορίζουμε το ακέραιο multicommodity flow problem (integer multicommodity flow - IMFP).

Ορισμοί και συμβολισμοί

Έστω ένας κατευθυνόμενος γράφος $G = (N, A)$ και ένα σύνολο $K = 1, 2, \ldots, k$ από αιτήσεις δρομολόγησης αγαθών. Σε κάθε κόμβο $i \in N$, για κάθε αίτηση σύνδεσης k αντιστοιχούμε μια μεταβλητή b_i^k. Αν $b_i^k > 0$ τότε ο κόμβος i είναι κόμβος πηγή (source) για την αίτηση σύνδεσης k, με προσφορά b_i^k μονάδες. Αν $b_i^k < 0$ τότε ο κόμβος i είναι κόμβος προορισμός (sink) για την αίτηση σύνδεσης k, με ζήτηση $-b_i^k$ μονάδες. Αν $b_i^k = 0$ τότε ο κόμβος i είναι κόμβος αναμετάδοσης για την αίτηση σύνδεσης k.

Σε κάθε κατευθυνόμενη ακμή (i, j), για κάθε αίτηση σύνδεσης k αντιστοιχούμε μια

[2]εμφανίζεται συχνά σε δίκτυα μελέτης της κυκλοφορίας (traffic network), όσο αυξάνει η κίνηση αυξάνει το κόστος

μεταβλητή c_{ij}^k που αντιστοιχεί στο μοναδιαίο κόστος ροής του συγκεκριμένου αγαθού στην συγκεκριμένη ακμή. Υποθέτουμε ότι $c_{ij}^k > 0 \quad \forall (i,j) \in A, \forall k \in K$. Κάθε ακμή (i,j) έχει μια μέγιστη χωρητικότητα u_{ij}, που είναι το μέγιστο ποσό συνολικής ροής που μπορεί να διασχίσει την ακμή. Υποθέτουμε ότι κάθε μονάδα ροής που διέρχεται από μια ακμή καταλαμβάνει μια μονάδα από την χωρητικότητά του.

Το σύνολο όλων των απλών μονοπατιών για όλα τα ζευγάρια πηγή - προορισμός της αίτηση σύνδεσης k συμβολίζονται ως P^k και το σύνολο όλων των πηγών και προορισμών της σύνδεσης k συμβολίζονται ως Q^k. Αν η ακμή (i,j) ανήκει στο μονοπάτι p της αίτησης k, τότε $\delta_{ij}^k = 1$, αλλιώς $\delta_{ij}^k = 0$.

Το σύνολο όλων των απλών κύκλων της σύνδεσης k συμβολίζεται με C^k. Αν η ακμή (i,j) ανήκει στον κύκλο c της σύνδεσης k τότε $\delta_{ij}^{ck} = 1$, αλλιώς $\delta_{ij}^{ck} = 0$. Το μοναδιαίο κόστος ροής στον κύκλο c της σύνδεσης k είναι

$$c_c^k = \sum_{(i,j)\in A} \delta_{ij}^{ck} c_{ij}^k, \forall c \in C^k, \forall k \in K$$

Αν ο κόμβος i είναι η αρχή του μονοπατιού p της αίτησης σύνδεσης k τότε $\gamma_i^{pk} = 1$. Αν ο κόμβος i είναι ο προορισμός του μονοπατιού p της αίτησης σύνδεσης k τότε $\gamma_i^{pk} = -1$. Σε κάθε άλλη περίπτωση $\gamma_i^{pk} = 0$. Το κόστος ανά μονάδα ροής του αγαθού k στο μονοπάτι p είναι

$$c_c^k = \sum_{(i,j)\in A} \delta_{ij}^{ck} c_{ij}^k, \forall c \in C^k, \forall k \in K.$$

Node - Arc Formulation

Μια μορφή του ακέραιου multicommodity flow μπορεί να παραχθεί χρησιμοποιώντας μεταβλητές απόφασης που αναπαριστούν την ροή σε όλες τις ακμές για όλες τις αιτήσεις σύνδεσης. Αυτές οι μεταβλητές απόφασης συμβολίζονται ως x_{ij}^k. Το ακέραιο πρόγραμμα είναι :

$$\begin{aligned}
&\text{Minimize} \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k x_{ij}^k && \text{subject to} \\
&\sum_{(i,j)\in A} x_{ij}^k - \sum_{(i,j)\in A} x_{ji}^k = b_i^k, && \forall i \in N, \quad \forall k \in K && (5.18)\\
&\sum_{k\in K} x_{ij}^k \le u_{ij}, && \forall (i,j) \in A && (5.19)\\
&x_{ij}^k \ge 0 \quad \text{και ακέραιος}, && \forall (i,j) \in A, \quad \forall k \in K
\end{aligned}$$

Το σύνολο περιορισμών (5.18) είναι οι περιορισμοί διατήρησης ροής. Δηλώνουν ότι σε κάθε κόμβο η συνολική ροή που εισέρχεται και εξέρχεται σε ένα κόμβο είναι πάντοτε ίση με την προσφορά / ζήτηση του κόμβου αυτού (δεν υπάρχει δηλαδή ποτέ απώλεια ροής σε ένα κόμβο).

Οι περιορισμοί (5.19) δηλώνουν ότι η συνολική ροή σε μια ακμή δεν μπορεί να ξεπερνά τη χωρητικότητα του τόξου αυτού. Χωρίς αυτούς τους περιορισμούς το πρόβλημα θα μπορούσε να λυθεί επιλύοντας ένα πρόβλημα ελάχιστης ροής για κάθε αίτηση σύνδεσης ξεχωριστά.

Arc - Path Formulation

Μια άλλη μορφή του ακέραιου multicommodity flow μπορεί να παραχθεί χρησιμοποιώντας μεταβλητές απόφασης y_p^k που αναπαριστούν την ροή της σύνδεσης k στο μονοπάτι p. Το ακέραιο πρόγραμμα είναι:

$$\begin{aligned}
&\text{Minimize} \sum_{k\in K}\sum_{p\in P^k} c_p^k y_p^k && \text{subject to} \\
&\sum_{p\in P^k} \gamma_i^{pk} y_p^k = b_i^k, && \forall i \in Q^k, \quad \forall k \in K && (5.20)\\
&\sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^{pk} y_p^k \le u_{ij}, && \forall (i,j) \in A && (5.21)\\
&y_p^k \ge 0 \quad \text{και ακέραιος}, && \forall p \in P^k, \quad \forall k \in K
\end{aligned}$$

Οι περιορισμοί (5.20) εκφράζουν τη διατήρηση της ροής, ενώ οι περιορισμοί (5.21)

δηλώνουν ότι η ροή σε μία ακμή δεν μπορεί να ξεπερνά την χωρητικότητα της ακμής.

Η μορφή που παρουσιάσαμε σε αυτή την παράγραφο προκύπτει από την node - arc μορφή με εφαρμογή του θεωρήματος αποσύνθεσης ροής.

Θεώρημα (Αποσύνθεση Ροής)

(i) Ένα σύνολο ροών σε μονοπάτια και κύκλους είναι ισοδύναμο με το σύνολο ροών σε ακμές που προκύπτει από την εφαρμογή της ακόλουθης έκφρασης:

$$x_{ij}^k = \sum_{p \in P^k} \delta_{ij}^{pk} y_p^k + \sum_{c \in C^k} \delta_{ij}^{ck} y_c^k, \quad \forall (i,j) \in A, \quad \forall k \in K \tag{5.22}$$

όπου y_p^k η ροή στο μονοπάτι p της σύνδεσης k και y_c^k η ροή στον κύκλο c της σύνδεσης k.

(ii) Κάθε σύνολο ροών σε μονοπάτια αντιστοιχεί σε ένα σύνολο (όχι απαραίτητα μοναδικό) ροών σε μονοπάτια και κύκλους.

(iii) Όλα τα μονοπάτια με θετική ροή έχουν την πηγή τους σε ένα κόμβο προσφοράς ($b_i^k > 0$) και τον προορισμό σε ένα κόμβο ζήτησης ($b_i^k < 0$).

Αν υποθέσουμε ότι όλα τα μοναδιαία κόστη είναι μη αρνητικά, τότε δεν θα υπάρχουν κύκλοι με αρνητικό κόστος και η βέλτιστη λύση υπακούει στην ακόλουθη πρόταση:

Πρόταση Ένα στιγμιότυπο του ακέραιου multicommodity flow στο οποίο όλοι οι κύκλοι έχουν μη αρνητικό κόστος και το σύνολο των εφικτών λύσεων είναι μη κενό έχει μία βέλτιστη λύση ώστε :

$$y_c^k = 0, \quad \forall k \in K, \quad \forall c \in C^k.$$

5.8.5 Maximum Concurrent Flow

Έστω ένας γράφος $G = (V, E)$ με χωρητικότητες στις ακμές $u : E \to \mathbb{R}^+$ και ένα σύνολο $K = \{1, 2, \ldots, k\}$ από αιτήσεις σύνδεσης. Αν $i \in K$ είναι μια αίτηση σύνδεσης τότε (s_i, t_i) είναι το ζευγάρι πηγή - προορισμός της αίτησης αυτής. Για κάθε αίτηση έχουμε μια απαίτηση (demand) $d_i > 0$. Θέλουμε να βρούμε το μέγιστο ποσοστό z, ώστε

τουλάχιστον z της εκατό ποσοστό κάθε απαίτησης να μπορεί να δρομολογηθεί, χωρίς να παραβιαστούν οι περιορισμοί χωρητικότητας. Η μοντελοποίηση του προβλήματος ως γραμμικό πρόγραμμα είναι :

$$\begin{aligned} & \text{maximize} \quad \lambda && \text{subject to} \\ \sum_{(i,j)\in E} f_{ij}^k - \sum_{(j,i)\in E} f_{ji}^k = 0, & && \forall i \notin \{s_k, t_k\} && (5.23) \\ \sum_{(i,j)\in E} f_{ij}^k = \lambda d_k, & && \text{για} \quad i = s_k && (5.24) \\ f_{ij} \leq u_{ij}, & && \forall (i,j) \in E && (5.25) \end{aligned}$$

όπου :

f_{ij}^k εκφράζει τη ροή της αίτησης σύνδεσης k στην ακμή (i,j), αν η ροή έχει την ίδια κατεύθυνση με την ακμή (i,j), $f_{ij}^k > 0$, αλλιώς $f_{ij}^k < 0$.

f_{ij} είναι η συνολική ροή στην ακμή (i,j) $f_{ij} = \sum_{1\leq k\leq K} |f_{ij}^k|$.

Το παραπάνω πρόβλημα με χρήση του θεωρήματος αποσύνθεσης ροής μπορεί να μετασχηματιστεί ως εξής :

$$\begin{aligned} & \text{maximize} \quad \lambda && \text{subject to} \\ \sum_{P:(i,j)\in P} x(P) \leq u_{ij}, & && \forall (i,j) \in E && (5.26) \\ \sum_{P\in P_k} x(P) \geq \lambda d_k, & && \forall k && (5.27) \\ x(P) \geq 0, & && \forall P && (5.28) \end{aligned}$$

όπου :

P_k το σύνολο των μονοπατιών μεταξύ των κόμβων (s_k, t_k).

P το σύνολο όλων των μονοπατιών όλων των αιτήσεων σύνδεσης $P = \cup_k P_k$

$x(P)$ το ποσό της ροής που διέρχεται από το μονοπάτι P

Ισοδύναμο με το πρόβλημα, που μόλις περιγράψαμε, είναι το πρόβλημα του υπολογισμού του μικρότερου λόγου (ratio), κατά τον οποίο πρέπει να αυξηθούν οι χωρητικότητες ώστε να δρομολογηθούν όλες οι απαιτήσεις (demand). Τα αντίστοιχα προβλήματα είναι:

$$\begin{aligned} & \text{minimize} \quad \xi && \text{subject to} \\ \sum_{(i,j)\in E} f_{ij}^k - \sum_{(j,i)\in E} f_{ji}^k = 0, & && \forall i \notin \{s_k, t_k\} && (5.29) \\ \sum_{(i,j)\in E} f_{ij}^k = d_k, & && \gamma\iota\alpha i = s_k && (5.30) \\ f_{ij} \leq \xi u_{ij}, & && \forall (i,j) \in E && (5.31) \end{aligned}$$

και με την χρήση μονοπατιών :

$$\begin{aligned} & \text{minimize} \quad \xi && \text{subject to} \\ \sum_{P:(i,j)\in P} x(P) \leq \xi u_{ij}, & && \forall (i,j) \in E && (5.32) \\ \sum_{P\in P_k} x(P) \geq d_k, & && \forall k && (5.33) \\ x(P) \geq 0, & && \forall P && (5.34) \end{aligned}$$

5.8.6 Προβλήματα πολλαπλών πηγών αδιάσπαστης ροής (Multiple source unsplittable flow)

Έστω ένας γράφος $G = (V, E)$ και ένα σύνολο $T = \{(s_i, t_i) : i = 1, \ldots, k\}$ από k ζευγάρια κορυφών (αιτήσεις σύνδεσης). Κάθε αίτηση σύνδεσης έχει μια απαίτηση d_i και ένα βάρος (κέρδος) w_i. Επίσης κάθε ακμή $(i, j) \in E$ έχει μια χωρητικότητα u_{ij}. Αναζητούμε ένα υποσύνολο ζευγαριών του T, που δρομολογούν όλη την απαίτηση τους σε μοναδικά μονοπάτια χωρίς να παραβιάζουν τις χωρητικότητες των ακμών και μεγιστοποιούν το συνολικό βάρος. Η ελάχιστη χωρητικότητα στις ακμές έχει τιμή $maxd_i$. Αν $u_{ij} = 1$ και $w_i = 1$ τότε προκύπτει το edge disjoint path πρόβλημα.

Το πρόβλημα μοντελοποιείται ως ένα πρόβλημα ακέραιου προγραμματισμού ως εξής:

$$\text{maximize} \quad \sum_{i=1}^{k} w_i x_i, \qquad \text{subject to}$$

$$\sum_{\pi \in P_i} f_\pi = x_i, \qquad i = 1, \ldots, k \tag{5.35}$$

$$\sum_{i=1}^{k} \sum_{\pi \in P_i : \pi \ni (i,j)} d_i f_\pi \leq u_{ij}, \qquad (i,j) \in E \tag{5.36}$$

$$x_i \in \{0,1\}, \qquad i = 1, \ldots, k$$

$$f_\pi \in \{0,1\}, \qquad \pi \in \cup_{i=1}^{k} P_i \tag{5.37}$$

όπου

P_i Το σύνολο των μονοπατιών (s_i, t_i)

f_π Η ροή στο μονοπάτι $\pi \in P_i$

x_i Είναι μια δυαδική μεταβλητή. Αν $x_i = 1$ η αίτηση i μπαίνει στην τελική λύση, αλλιώς όχι.

5.8.7 Προβλήματα μοναδικής πηγής αδιάσπαστης ροής Single source unsplittable flow)

Παρόμοια με το παραπάνω πρόβλημα ορίζεται το πρόβλημα μοναδικής πηγής αδιάσπαστης ροής. Το πρόβλημα διαφοροποιείται σε σχέση με το προηγούμενο μόνο στον ορισμό του συνόλου T. Σ′ αυτή την περίπτωση διαθέτουμε μόνο ένα κόμβο πηγή s, έτσι $T = \{(s, t_i) : i = 1, \ldots, k\}$. Προσπαθούμε να δρομολογήσουμε την απαίτηση d_i κάθε αγαθού i σε ένα μοναδικό μονοπάτι ροής $s-t_i$ χωρίς να παραβιάζονται οι χωρητικότητες των ακμών. Υπάρχουν 3 εκδόσεις βελτιστοποίησης του παραπάνω προβλήματος.

Ελαχιστοποίηση συμφόρησης (minimum congestion) Αναζητούμε μια αδιάσπαστη ροή f που ικανοποιεί όλες τις απαιτήσεις και ελαχιστοποιεί την συμφόρηση στο δίκτυο, για παράδειγμα την ποσότητα:

$$max\{\{max_e\{\frac{f_e}{u_e}\}, 1\}\}$$

Δηλαδή αναζητούμε τον μικρότερο αριθμό $\alpha \geq 1$ κατά τον οποίο αν πολλαπλασιάσουμε όλες τις χωρητικότητες θα ικανοποιηθούν όλες οι απαιτήσεις.

Μέγιστη δρομολογήσιμη απαίτηση (maximum routable demand) Αναζητούμε ένα υποσύνολο αιτήσεων σύνδεσης με μέγιστη συνολική απαίτηση, που μπορούμε να δρομολογήσουμε σε μοναδικά μονοπάτια, ικανοποιώντας τους περιορισμούς χωρητικότητας.

Ελάχιστος αριθμός κύκλων (minimum number of rounds) Διαμερίζουμε το σύνολο των αιτήσεων σύνδεσης σε ένα ελάχιστο αριθμό υποσυνόλων, που αποκαλούμε κύκλους (rounds), ώστε οι αιτήσεις που βρίσκονται στο ίδιο υποσύνολο να μπορούν να δρομολογηθούν σε μοναδικά μονοπάτια χωρίς να παραβιάζουν τους περιορισμούς χωρητικότητας.

Στη γενική περίπτωση αυτό το πρόβλημα μοντελοποιείται σαν ένα πρόβλημα ακέραιου προγραμματισμού σύμφωνα με το μοντέλο 5.8.6.

Κεφάλαιο 6

Εναλλακτικές Μέθοδοι Επίλυσης Γραμμικών Προγραμμάτων

Στα προηγούμενα κεφάλαια παρουσιάσαμε αναλυτικά την μέθοδο Simplex, για την επίλυση προβλημάτων γραμμικού προγραμματισμού. Σ΄ αυτό το κεφάλαιο παρουσιάζουμε εναλλακτικές μεθόδους επίλυσής τους. Οι μέθοδοι που θα εξετάσουμε αφορούν την επίλυση γραμμικών προβλημάτων με μεγάλο αριθμό μεταβλητών. Η πρώτη προσέγγιση καλείται γένεση στηλών (column generation) και είναι στην ουσία μια τροποποίηση της μεθόδου simplex. Η βασική ιδέα αυτής της προσέγγισης είναι ότι ποτέ δεν διατηρούμε σαφώς τις στήλες του γραμμικού (ή ακέραιου) προβλήματος αλλά τις δημιουργούμε τη στιγμή που τις χρειαζόμαστε. Για την παρουσίαση αυτών των μεθόδων θα χρησιμοποιήσουμε τα μοντέλα multicommodity flow που παρουσιάσαμε στο προηγούμενο κεφάλαιο. Ήδη έχουμε δει ένα παράδειγμα στο Κεφάλαιο 5 επιλύοντας το μονοδιάστατο πρόβλημα κοπής.

6.1 Γένεση Κολόνας (Column Generation)

Μια από τις βασικές ιδιότητες της μεθόδου simplex είναι ότι δεν απαιτείται ο πίνακας των περιορισμών A να είναι συνεχώς διαθέσιμος. Πραγματικά στη simplex οι στήλες του A χρειάζονται μόνο για να εξετάσουμε αν τα reduced costs $\hat{c_j} = c_j - y^T A_j$ είναι αρνητικά και αν δεν είναι να δημιουργήσουμε μια στήλη A_t που να παραβιάζει αυτή τη συνθήκη. Αυτό που είναι αναγκαίο είναι η ύπαρξη μιας τεχνικής που να προσδιορίζει

αν μια βασική λύση είναι βέλτιστη και αν όχι να παράγει μια στήλη που παραβιάζει τις συνθήκες βέλτιστου. Όταν ο πίνακας περιορισμών A έχει μια συγκεκριμένη γνωστή δομή, είναι δυνατό να βρεθεί ένα A_t χωρίς σαφή γνώση όλων των στηλών του πίνακα A. Αυτή η τεχνική που δημιουργεί στήλες μόνο όταν είναι αναγκαίο και επιλύοντας ένα άλλο βοηθητικό πρόβλημα π.χ. το πρόβλημα του σακιδίου (knapsack) ή το πρόβλημα συντομότερου μονοπατιού ονομάζεται column generation (βλ. Κεφάλαιο 5, το μονοδιάστατο πρόβλημα κοπής).

6.2 Μέθοδοι Αποσύνθεσης Τιμής (Price Decomposition Methods)

Σε κάποιες εφαρμογές γραμμικού προγραμματισμού οι περιορισμοί του προβλήματος χωρίζονται σε δύο κατηγορίες. Μία κατηγορία ''εύκολων'' περιορισμών και μια κατηγορία ''δύσκολων'' περιορισμών. Αυτό συμβαίνει στα multicommodity flow προβλήματα όπου οι περιορισμοί που περιγράφουν το δίκτυο θεωρούνται οι ''εύκολοι'' περιορισμοί, ενώ οι επιπρόσθετοι περιορισμοί της πιο γενικής μορφής θεωρούνται οι ''δύσκολοι'' περιορισμοί. Αν αυτοί οι δύσκολοι περιορισμοί μπορούσαν να παραληφθούν, τότε θα μπορούσαν να χρησιμοποιηθούν πιο αποδοτικές μέθοδοι για την επίλυση του προβλήματος.

Η αποσύνθεση είναι ένα εργαλείο για την επίλυση γραμμικών προβλημάτων της δομής που περιγράφηκε παραπάνω και είναι ένα παράδειγμα της column generation τεχνικής. Η αποσύνθεση χρησιμοποιεί μια αλλαγή των μεταβλητών για να μετασχηματίσει το αρχικό γραμμικό πρόγραμμα σε ένα νέο γραμμικό πρόγραμμα που περιέχει μόνο τους ''δύσκολους'' περιορισμούς. Αν ο αριθμός των ''δύσκολων'' περιορισμών είναι μικρός τότε το πρόβλημα είναι δυνατό να λυθεί με τη βοήθεια της simplex σε λίγες επαναλήψεις. Ωστόσο ο έλεγχος βέλτιστου για το νέο γραμμικό πρόγραμμα απαιτεί την λύση ενός άλλου γραμμικού προγράμματος που εμπεριέχει τους ''εύκολους'' περιορισμούς. Στη συνέχεια παρουσιάζουμε κάποιες τεχνικές επίλυσης που βασίζονται στην παραπάνω ιδέα.

Για την παρουσίαση των μεθόδων θα χρησιμοποιήσουμε το μοντέλο του minimum cost multicommodity flow που περιγράφεται στην παράγραφο 5.8.1. Αν υποθέσουμε ότι $c_j^k > 0$ είναι δυνατό να προστεθούν μη αρνητικές μεταβλητές x^{k+} και x^{k-} τέτοιες

ώστε $x_j^k = x^{k+} - x^{k-}$ και $|x_j^k| = x^{k+} + x^{k-}$ παίρνουμε το παρακάτω ισοδύναμο πρόβλημα.

$$NMP = \begin{cases} \text{Minimize} \quad \sum_{1\le k\le K}[C^k]^T x^k \quad \text{subject to} \\ \\ \sum_{1\le k\le K} Dx^k \le u, \quad \forall (i,j) \in A \qquad (1) \\ \\ \mathcal{B}x^k = b^k, \quad \text{για} \quad k = 1,2,\ldots,K \qquad (2) \\ \\ x^k \ge 0, \quad \forall \quad k = 1,2,\ldots,K \end{cases}$$

όπου

- $[C^k]^T = \left[[c^k]^T [c^k]^T\right]$
- $[x^k]^T = \left[[x^{k+}]^T [c^{k-}]^T\right]$
- $B = \left[\mathcal{N} \quad (-\mathcal{N})\right]$
- $D = \left[I \quad I\right]$
- $u = (u_1, u_2, \ldots, u_j)^T$

Παρατηρούμε ότι στην αρχική του μορφή το πρόβλημα έχει $M = Kn + j$ περιορισμούς και $(2K+1)j$ μεταβλητές.

Αν αγνοήσουμε το σύνολο περιορισμών σύζευξης (2) του NMP το πρόβλημα θα αποσυντήθονταν σε ένα πρόβλημα εύρεσης ελαχίστου μονοπατιού για κάθε αίτηση σύνδεσης. Υποθέτουμε ότι έχουμε βρει αυτά τα ελάχιστα μονοπάτια. Αν κανένας από τους περιορισμούς δεν παραβιάζεται τότε το πρόβλημα έχει λυθεί. Αλλιώς μπορούμε να τροποποιήσουμε τα κόστη δρομολόγησης εισάγοντας τιμές (ποινές) για την χρήση των συνδέσεων που παραβιάζουν τους περιορισμούς, λύνοντας με αυτό τον τρόπο ένα νέο πρόβλημα ελαχίστων μονοπατιών, το οποίο προκύπτει. Οι πληροφορίες των ελαχίστων μονοπατιών, που υπολογίζονται σε κάποια επανάληψη, μπορούν να χρησιμοποιηθούν για τον υπολογισμό νέων τιμών (ποινών) και για την εύρεση τελικά μιας εφικτής λύσης για το NMP.

6.2.1 Lagrangian Relaxation

Η Lagrangian Relaxation (χαλάρωση Lagrange) είναι μια ελκυστική τεχνική για την εύρεση κάτω φραγμάτων στη βέλτιστη τιμή ακεραίων προγραμμάτων με πολλές μεταβλητές σε πολυωνυμικό χρόνο. Η μέθοδος τοποθετεί ποινές (πολλαπλασιαστές Lagrange) στους περιορισμούς σύζευξης και τις συμπεριλαμβάνει στην αντικειμενική συνάρτηση για να αποσυντεθεί το πρόβλημα σε ξεχωριστά προβλήματα εύρεσης ροής ελαχίστου κόστους για κάθε αίτηση σύνδεσης. Έστω ένα διάνυσμα πολλαπλασιαστών Lagrange $\lambda \in \mathbb{R}^{j+}$, το οποίο σχετίζεται με τους περιορισμούς σύζευξης του NMP, τότε παίρνουμε το ακόλουθο ισοδύναμο πρόβλημα:

$$L(\lambda) = \sum_{1 \le k \le K} [C^k]^T x^k + \lambda^T \Big(\sum_{1 \le k \le K} D x^k - u \Big) \quad \text{subject to} \tag{6.1}$$

$$\mathcal{B} x^k = b^k, \quad \text{για} \quad k = 1, 2, \ldots, K \tag{6.2}$$

$$x^k \ge 0, \quad \forall \quad k = 1, 2, \ldots, K \tag{6.3}$$

Για οποιαδήποτε σταθερή τιμή των πολλαπλασιαστών, ο όρος $\lambda^T u$ είναι μια σταθερά που μπορεί να αγνοηθεί. Το υποπρόβλημα Lagrange ελαττώνεται σε K υποπροβλήματα της μορφής :

$$SP_k(\lambda) : \pi_k(\lambda) = min\{([C^k]^T + \lambda^T D)x^k : x^k \in P_k\} \text{ όπου}$$
$$P_k = \{x^k : Bx^k = b^k, x^k \ge 0\}.$$

Το $SP_k(\lambda)$ μπορεί να λυθεί χρησιμοποιώντας έναν αλγόριθμο εύρεσης ελαχίστου μονοπατιού μεταξύ των κόμβων s_k και t_k σε μη κατευθυνόμενο γράφο. Το προκύπτον ελάχιστο μονοπάτι αντιστοιχεί σε ένα κατευθυνόμενο μονοπάτι στο γράφο το συνδεδεμένο με το πρόβλημα $SP_k(\lambda)$. Η λύση x^k που προκύπτει από την δρομολόγηση της συνολικής απαίτησης d_k σ′ αυτό το μονοπάτι είναι μια βέλτιστη λύση στο $SP_k(\lambda)$ και καλείται *βέλτιστη λύση ελαχίστου μονοπατιού*. Η δυϊκή συνάρτηση $L(\lambda)$ είναι ένα κάτω φράγμα στη βέλτιστη λύση του NMP για οποιαδήποτε επιλογή του διανύσματος λ. Η συνάρτηση αυτή είναι κοίλη και συνεχής αλλά όχι διαφορίσιμη. Για να μεγιστοποιήσουμε αυτή την συνάρτηση απαιτούνται subgradient μέθοδοι που είναι αρκετά εύκολες στην υλοποίησή τους. Ωστόσο ακόμα και όταν το διάνυσμα πολλαπλασιαστών Lagrange είναι βέλτιστο (π.χ. ίσο με το διάνυσμα λ^* των βέλτιστων δυϊκών μεταβλητών

που έχουν ανατεθεί στους περιορισμούς σύζευξης), η λύση x, που προκύπτει από την επίλυση των K προβλημάτων ελαχίστου μονοπατιού, δεν είναι εφικτή για το NMP. Για μια αίτηση σύνδεσης k η βέλτιστη λύση x^k στο NMP αντιστοιχεί σε μια βέλτιστη λύση του $SP_k(\lambda^*)$, που προκύπτει ως ένας κυρτός συνδυασμός βέλτιστων λύσεων ελαχίστων μονοπατιών του $SP_k(\lambda^*)$. Τα αποτελέσματα της χαλάρωσης Lagrange μπορούν να χρησιμοποιηθούν για την επίτευξη μιας αρχικής εφικτής βάσης για την εύρεση μιας προσεγγιστικής λύσης που θα βασίζεται στη μέθοδο simplex. Επίσης μπορούν να χρησιμοποιηθούν για την σταθεροποίηση κάποιων μεταβλητών σε μηδέν.

Η βέλτιστη τιμή του υποπροβλήματος $SP_k(\lambda)$ βρίσκεται σε ένα ακρότατο σημείο του πολυέδρου P_k. Επιπρόσθετα επειδή ο πίνακας B είναι ένας πίνακας δικτύου, κάθε βασική λύση αντιστοιχεί σε ένα δέντρο επικάλυψης (spanning tree) και κάθε τέτοιο δέντρο αντιστοιχεί σε μια λύση ελαχίστων μονοπατιών που ορίζουν ένα ακρότατο σημείο στο πολύεδρο. Σημειώνουμε ότι κάθε ακρότατο σημείο του πολυέδρου αντιστοιχεί σε πολλές βάσεις. Επειδή όλα τα ακραία σημεία του P_K είναι διανύσματα με τιμές στο διάστημα $0, d_k$, θεωρούμε τα κανονικοποιημένα ακρότατα σημεία $y_1^k, y_2^k, \ldots, y_{n_k}^k$ του P_k που αντιστοιχούν στο χαρακτηριστικό διάνυσμα των λύσεων ελαχίστων μονοπατιών.

Το δυϊκό πρόβλημα Lagrange $max\{L(\lambda) : \lambda \geq 0\}$ μπορεί να μετασχηματιστεί με βάση αυτά τα ακρότατα σημεία ως εξής:

$$DLP = \begin{cases} \text{Maximize} \quad \sum_{1 \leq k \leq K} d_k \pi_k - \lambda^T u \quad \text{subject to} \\ \\ \pi_k - \lambda^T D y_l^k \leq [C^k]^T y_l^k, \quad k = 1, \ldots, K, \qquad l = 1, \ldots, n_k \\ \\ \lambda \geq 0 \end{cases}$$

Η κανονικοποίηση δημιουργεί ένα πρόβλημα όπου όλοι οι συντελεστές των περιορισμών είναι $-1, 0, 1$. Το ενδιαφέρον του μοντέλου DLP είναι ότι το σύνολο των κορεσμένων περιορισμών στη βέλτιστη λύση δίνει όλα τα ακρότατα σημεία λύσεων στα υποπροβλήματα $SP_k(\lambda^*)$ και ότι οι αντίστοιχες τιμές των δυϊκών μεταβλητών μας δίνουν πληροφορία για να κατασκευάσουμε μια βέλτιστη λύση του NMP. Το DLP έχει εκθετικό αριθμό περιορισμών. Ωστόσο δεν είναι συνέχεια απαραίτητοι στη λύση του προβλήματος και είναι δυνατό να χρησιμοποιηθούν μέθοδοι γένεσης περιορισμών για την επίλυση του DLP ή μέθοδοι γένεσης στηλών για την επίλυση του δυϊκού του.

6.2.2 Dantzig Wolfe αποσύνθεση

Η μέθοδος ξεκινάει μετασχηματίζοντας το NMP σε μια εναλλακτική μορφή, η οποία είναι γνωστή σαν node path formulation του multicommodity flow προβλήματος. Δεσμεύουμε μια δυϊκή μεταβλητή α_l^k σε κάθε περιορισμό, (k,l) του DLP και προκύπτει το ακόλουθο δυϊκό πρόβλημα.

$$\text{minimize} \quad \sum_{k,l} [C_k]^T y_l^k \alpha_l^k \qquad \text{subject to} \tag{6.4}$$

$$\sum_{l} \alpha_l^k = d_k, \qquad k = 1, \ldots, K \tag{6.5}$$

$$\sum_{l,k} \alpha_l^k D y_l^k \leq u \tag{6.6}$$

$$\alpha_l^k \geq 0, \qquad k = 1, \ldots, K \tag{6.7}$$

Οι περιορισμοί (6.5) στο παραπάνω μοντέλο δηλώνουν ότι η μεταβλητή x^k είναι ένας κυρτός συνδυασμός των ακραίων σημείων του P^k για όλα τα k. Οι περιορισμοί (6.6) είναι οι μετασχηματισμένοι περιορισμοί σύζευξης που προκύπτουν όταν αντικαταστήσουμε τις αρχικές μεταβλητές με τους κυρτούς συνδυασμούς.

Αν εισάγουμε διανύσματα p_l^k που ορίζονται ως $p_{l_j}^k = y_{l_j}^k + y_{l_{j+|E|}}^k, j = 1, \ldots, |E|$ τότε προκύπτει το:

$$NMP = \begin{cases} \text{minimize} \quad \sum_{k,l} [c_k]^T p_l^k \alpha_l^k \quad \text{subject to} \\ \sum_l \alpha_l^k = d_k, \quad k = 1, \ldots, K \\ \sum_{l,k} \alpha_l^k p_l^k \leq u \\ \alpha_l^k \geq 0, \quad k = 1, \ldots, K \end{cases}$$

όπου

- p_l^k είναι το χαρακτηριστικό διάνυσμα ενός μονοπατιού που συνδέει s_k με t_k. Για παράδειγμα το j - οστό στοιχείο του διανύσματος είναι 1 αν η ακμή j ανήκει στο μονοπάτι p_l^k και 0 αλλιώς.

- $[c_k]^T p_l^k$ είναι το κόστος του μονοπατιού ανά μονάδα ροής που δρομολογείται σ΄ αυτό.

- Τα δύο σύνολα περιορισμών εκφράζουν ότι d_k μονάδες ροής πρέπει να δρομολο-

γηθούν μεταξύ (s_k, t_k) και ότι δεν πρέπει να παραβιάζονται οι χωρητικότητες.

Το NMP έχει εκθετικό αριθμό στηλών, αλλά δεν είναι απαραίτητο να εκφράζουμε όλες τις στήλες του γραμμικού προγράμματος. Το reduced cost της μεταβλητής α_l^k είναι $w_l^k = [c^k + \lambda]^T p_l^k - \pi_k$, όπου λ είναι το διάνυσμα των πολλαπλασιαστών simplex που σχετίζονται με τους περιορισμούς χωρητικότητας και π_k το διάνυσμα των πολλαπλασιαστών simplex που σχετίζονται με τους περιορισμούς κυρτότητας που αντιστοιχούν στην αίτηση σύνδεσης k. Έτσι το $min_l w_l^k$ μπορεί να υπολογιστεί με την εύρεση ενός ελαχίστου μονοπατιού μεταξύ (s_k, t_k) στο μη κατευθυνόμενο γράφο όπου τα μήκη των ακμών δίνονται από $c_k + \lambda$.

Στον αλγόριθμο αποσύνθεσης το NMP καλείται *κυρίαρχο πρόβλημα* (master problem). Τα K προβλήματα ελαχίστων μονοπατιών που λύνονται για να δημιουργήσουμε τις στήλες του κυρίαρχου προβλήματος καλούνται *δορυφόροι* (satellite problems). Σε κάθε βήμα του αλγορίθμου επιλύεται ένα πρόγραμμα που περιέχει ένα υποσύνολο των στηλών του κυρίαρχου προβλήματος (reduced master problem - rmp) . Το πρώτο rmp αρχικοποιείται με ένα σύνολο στηλών που αντιστοιχούν σε ένα πίνακα βάση και οι στήλες αντιστοιχούν στις χαλαρωμένες (στους πραγματικούς αριθμούς) μεταβλητές. Αν δεν υπάρχει ένα τέτοιο αρχικό σύνολο στηλών, προστίθενται φανταστικές μεταβλητές στο NMP ώστε να πάρουμε μια αρχική εφικτή βάση του προβλήματος. Όταν λύσουμε ένα rmp βέλτιστα, επιλύουμε τα K προβλήματα δορυφόρους για να ελέγξουμε αν η λύση είναι εφικτή για το NMP. Αν τα reduced costs που αντιστοιχούν στην λύση των προβλημάτων δορυφόρων είναι μη αρνητικά τότε ο αλγόριθμος τερματίζει. Αλλιώς προσθέτουμε νέες στήλες που είναι υποψήφιες να μπουν στη βάση και η simpex συνεχίζει. Όταν βρίσκουμε μια λύση σε ένα rmp τα reduced costs όλων των μεταβλητών του προγράμματος είναι μη αρνητικά, δηλαδή $\lambda \geq 0$ και τα μήκη των ακμών στα προβλήματα δορυφόρους είναι μη αρνητικά. Έτσι τα προβλήματα δορυφόροι μπορούν να λυθούν με χρήση του αλγορίθμου Dijkstra.

Κεφάλαιο 7

Δυναμικός Προγραμματισμός (Dynamic Programming)

Ο Δυναμικός προγραμματισμός είναι μία μέθοδος επίλυσης των προβλημάτων βελτιστοποίησης, η οποία βασίζεται σε μια νοερή απαρίθμηση των λύσεων. Για να εφαρμοσθεί με χρησιμότητα, απαιτεί τα επεξεργαζόμενα προβλήματα να έχουν μια ειδική δομή, σειριακού τύπου.

Πολλά συνδυαστικά προβλήματα μπορούν να γραφούν υπό αυτή τη μορφή και να επιλυθούν έτσι με δυναμικό προγραμματισμό.

Η βασική αρχή του δυναμικού προγραμματισμού είναι:

- Εμφυτεύουμε το πρόβλημα μέσα σε μια οικογένεια προβλημάτων της ίδιας φύσης.
- Συνδέουμε με μια αναδρομική σχέση τις βέλτιστες λύσεις αυτών των προβλημάτων.

Ας θεωρήσουμε όμως δύο απλά παραδείγματα.

7.1 Αναζήτηση συντομότερου μονοπατιού

Αναζήτηση του συντομότερου μονοπατιού από τον κόμβο 1 στον κόμβο K σε ένα γράφο $G = [X, \Gamma],\ |X| = n$ και $\Gamma : X \to X$ η συνάρτηση γειτόνων.

- Εμφυτεύουμε αυτό το πρόβλημα μέσα στην οικογένεια των n προβλημάτων: αναζήτηση του συντομότερου μονοπατιού από τον κόμβο 1 στους n κόμβους $1, 2, \ldots, n$ του γράφου G.

- Αν π_i είναι η τιμή του συντομότερου μονοπατιού από τον κόμβο 1 στον κόμβο i (βέλτιστη λύση του i-στου προβλήματος της οικογένειας), έχουμε την αναδρομική σχέση:

$$\pi_i = \min_{j \in \Gamma^{-1}(i)} \{\pi_j + l_{ji}\} \tag{7.1}$$

όπου $\Gamma(i) = \{k|(i,k)$ ανήκει στο γράφο $G\}$, δηλ. $\Gamma(i)$ είναι το σύνολο των άμεσων διαδόχων του κόμβου i και $\Gamma^{-1}(i) = \{m|(m,i)$ ανήκει στο γράφο $G\}$, δηλ. είναι το σύνολο των προηγουμένων κόμβων του κόμβου i. Η αναδρομική σχέση εκφράζει ότι το συντομότερο μονοπάτι από το 1 στον i περνά υποχρεωτικά από έναν από τους προηγούμενους κόμβους του κόμβου i. Η τιμή της βέλτιστης λύσης από τον κόμβο 1 στον κόμβο K είναι π_K και η αρχική συνθήκη είναι $\pi_1 = 0$. Θα επανέλθουμε όμως μετά το δεύτερο παράδειγμα.

7.2 Το πρόβλημα του σακιδίου (Knapsack)

Το πρόβλημα του σακιδίου (knapsack) σαν γραμμικό πρόγραμμα έχει ως εξής. Υποθέτουμε ότι όλοι οι συντελεστές είναι ακέραιοι.

$$P_n(b) = \begin{cases} \max \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_j x_j \leq b \\ x_j \in \{0,1\} \end{cases}$$

- Εμφυτεύουμε το πρόβλημα στην οικογένεια των $n(b+1)$ ακολούθων προβλημάτων:

$$p_k(y) = \{\max \sum_{j=1}^{k} c_j x_j \text{ με } \sum_{j=1}^{k} a_j x_j \leq y, x_j = 0 \text{ ή } 1\} \tag{7.2}$$

όπου y μεταβάλλεται από 0 στο b και k από 1 στο n.

- Αν $f_k(y)$ είναι η τιμή της βέλτιστης λύσης του προβλήματος $p_k(y)$ έχουμε την αναδρομική σχέση για $1 \leq k \leq n-1$ και $a_{k+1} \leq y \leq b$:

$$f_{k+1}(y) = \max\{f_k(y), f_k(y - a_{k+1}) + c_{k+1}\} \tag{7.3}$$

Η σχέση εκφράζει ότι, μέσα στη βέλτιστη λύση του $p_{k+1}(y)$ προβλήματος, έχουμε $x_{k+1} = 0$ ή $x_{k+1} = 1$.

Απομένει να επιλύσουμε την αναδρομική σχέση (7.3) για να βρούμε την τιμή της βέλτιστης λύσης $f_n(b)$ του αρχικού προβλήματος $P_n(b)$.

Ο αλγόριθμος δυναμικού προγραμματισμού έχει τώρα ως εξής ξεκινώντας από τη σχέση (7.3) και αφού ορίσουμε τις αρχικές συνθήκες.

- θεωρούμε $f_k(0) = 0$, $1 \leq k \leq n$ και $f_0(y) = -\infty$, $\forall y | 0 \leq y \leq b$
- υπολογίζουμε $f_1(y)$ για όλα τα y με $0 < y \leq b$. Θα έχουμε $f_1(y) = c_1$ αν $a_1 \leq y$ και $f_1(y) = -\infty$ αν $a_1 > y$.
- υπολογίζουμε $f_2(y)$ σε συνάρτηση του f_1 από την (7.3), και ούτω καθ' εξής μέχρι $f_n(y)$, $0 < y \leq b$.

Στο τέλος $f_n(b)$ είναι η τιμή της βέλτιστης λύσης.

Ο προηγούμενος αλγόριθμος επιτρέπει να βρούμε την τιμή της βέλτιστης λύσης. Απομένει να συγκεκριμενοποιήσουμε αυτή τη βέλτιστη λύση, δηλαδή να βρούμε τη δομή της λύσης.

Γι' αυτό ορίζουμε, από τη σχέση (7.3), το διάνυσμα $x_k(y)$, $0 \leq k \leq n-1$, $0 \leq y \leq b$ ως εξής:

$$x_{k+1}(y) = \begin{cases} 0 \text{ αν } f_{k+1}(y) = f_k(y) \\ 1 \text{ διαφορετικά} \end{cases} \tag{7.4}$$

Από τα διανύσματα $x_1(y), x_2(y), \ldots, x_n(y)$ για $0 \leq y \leq b$ μπορούμε τότε να επιτύχουμε τη δομή της βέλτιστης λύσης x^* με τον ακόλουθο αλγόριθμο (Σχήμα 7.1):

```
for k := n to 1 do
  begin
    x*_k = x_k(b)
    b = b − a_k x*_k
  end
```

Σχήμα 7.1: Εύρεση της δομής της βέλτιστης λύσης

Πολυπλοκότητα

Ο υπολογισμός των $f_k(y)$ και των $x_k(y)$ γίνεται σε χρόνο $O(nb)$ και ο αλγόριθμος επομένως χρειάζεται συνολικά χρόνο $O(nb)$.

Οι συναρτήσεις f_k και f_{k+1} δεν χρειάζονται παρά μόνο $O(b)$ μνήμη αλλά η αποθήκευση των διανυσμάτων $x_k(y)$ απαιτεί μνήμη $O(nb)$. Μπορούμε όμως και την εύρεση της δομής της βέλτιστης λύσης να την επιτύχουμε σε $O(b)$ μνήμη (άσκηση).

Το παράδειγμα 2 επιλύεται λοιπόν με δυναμικό προγραμματισμό σε ψευδοπολυωνυμικό χρόνο $O(nb)$ και σε μνήμη $O(b)$.

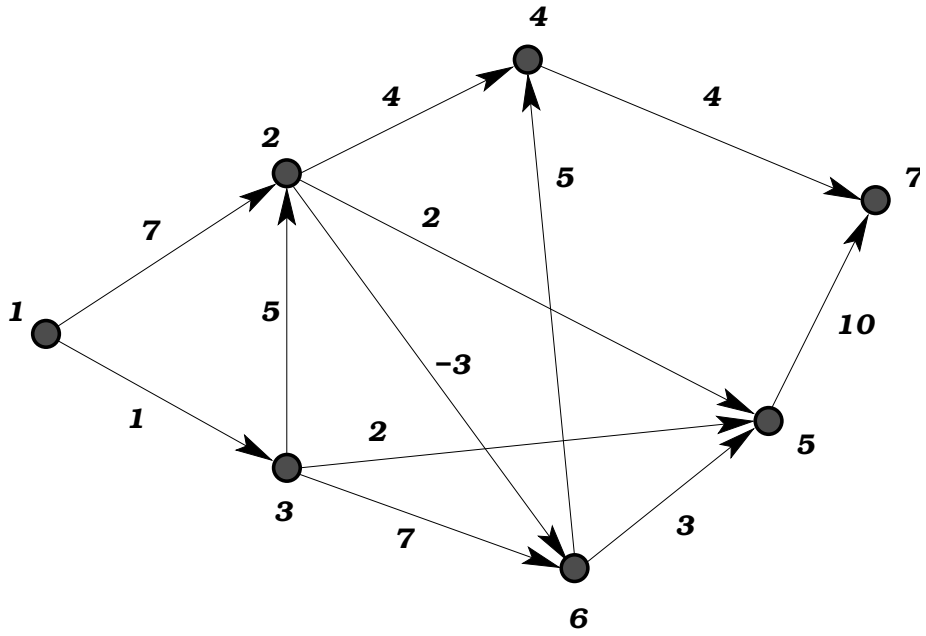
Ας επιστρέψουμε τώρα στο πρώτο παράδειγμα. Η αναδρομική σχέση (7.1) δεν είναι ευθέως σειριακού τύπου και το παράδειγμα 1 δεν επιλύεται γενικά τόσο εύκολα. Ας θεωρήσουμε κατ΄ αρχήν μια πολύ εύκολη περίπτωση: ο γράφος G είναι χωρίς κύκλο και ο κόμβος 1 είναι ο μοναδικός κόμβος του G με εσωτερικό βαθμό 0 (αυτός ο γράφος χωρίς κύκλο θα παριστάνει π.χ. το διάγραμμα εξέλιξης ενός συστήματος μέσα στο χρόνο). Ένας τέτοιος γράφος ανήκει στην κατηγορία γνωστή ως Κατευθυνόμενοι Ακυκλικοί Γράφοι (Directed Acyclic Graphs - DAG).

Σε αυτή την περίπτωση, μπορούμε να ορίσουμε πάνω στους κόμβους του γράφου μια συνάρτηση Layer έτσι ώστε όλοι οι κόμβοι με Layer $r+1$ να μην επιδέχονται προηγούμενους παρά μόνο κόμβους με Layer $\leq r$.

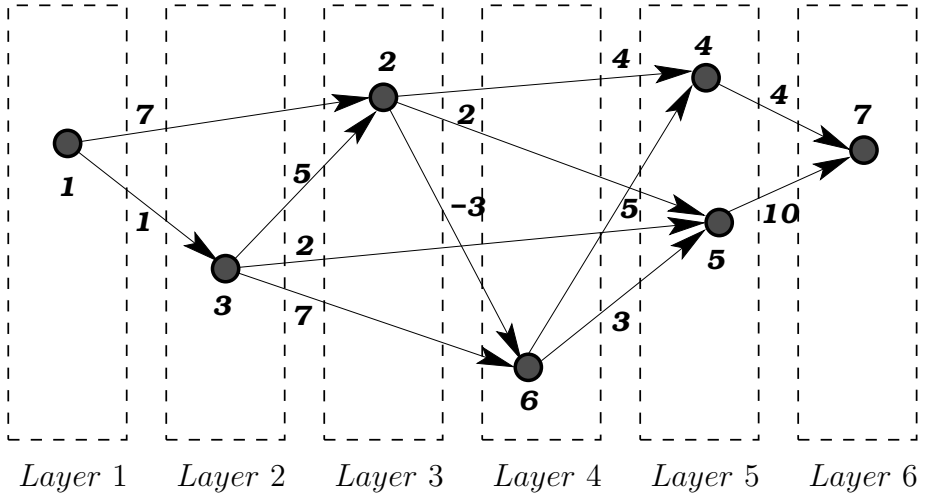
Υπό αυτή την προϋπόθεση ένας αλγόριθμος δυναμικού προγραμματισμού μπορεί απ΄ ευθείας να ορισθεί από τη σχέση (7.1). Το πλήθος των υποπροβλημάτων είναι n.

- Ο κόμβος 1 με επίπεδο 1 είναι έτσι με τιμή $\pi_1 = 0$.
- Υπολογίζουμε τα π_i επιπέδου 2 από το π_1 με την (7.1), τα π_i επιπέδου 3 από τα π_i επιπέδου 1 και 2 με την (7.1), κ.λ.π.

Ευρίσκουμε έτσι την τιμή π^*_K της βέλτιστης λύσης. Εδώ ακόμη, για να βρούμε τη



Σχήμα 7.2: Ένας γράφος που μπορεί να παρουσιασθεί σε επίπεδα.



Σχήμα 7.3: Η παρουσίαση του προηγούμενου γράφου σε επίπεδα.

δομή της βέλτιστης λύσης και όχι μόνο την τιμή της, ορίζουμε από τη σχέση (7.1) το διάνυσμα $P(i)$ ως εξής:

$$P(i) = j_0 \quad \text{αν } \pi_i = \pi_{j_0} + \ell_{j_0 i} \tag{7.5}$$

Επομένως το βέλτιστο μονοπάτι επιτυγχάνεται με τον ακόλουθο αλγόριθμο (Σχήμα 7.4):

```
(Δίδονται K και το διάνυσμα P(i))

τελικός κόμβος x_K; i = K
επανέλαβε για όλα τα i
      i ← P(i); προηγούμενος κόμβος x_i
      αν i = 1 τέλος
```

Σχήμα 7.4: Εύρεση της δομής του συντομότερου μονοπατιού σε ένα DAG.

Αν M είναι ο αριθμός τόξων του γράφου G ο προηγούμενος αλγόριθμος δυναμικού προγραμματισμού είναι χρόνου $O(M)$ αφού χρειαζόμαστε $O(\sum_{i=1}^{n} \Gamma^+(i)) = O(M)$ συγκρίσεις για την εύρεση της τιμής της βέλτιστης λύσης και $O(n)$ συγκρίσεις για την εύρεση της δομής της λύσης. Επίσης η απαιτούμενη μνήμη είναι συνολικά $O(n)$.

Ας θεωρήσουμε τώρα την περίπτωση όπου ο γράφος G είναι οποιοσδήποτε. Για να εισαγάγουμε ένα χαρακτήρα σειριακό στην (7.1), θέτουμε:

π_i^k τιμή του συντομότερου μονοπατιού από 1 στον i χρησιμοποιώντας το πολύ k ακμές.

Έχουμε τότε:

$$\pi_i^{k+1} = \min_{j \in \Gamma^{-1}(i)} \{\pi_j^k + l_{ji}, \pi_i^k\} \tag{7.6}$$

Η αναδρομική σχέση (7.6) είναι σειριακού τύπου, εκφράζεται σε συνάρτηση του π_i^k και η τιμή των συντομότερων μονοπατιών π_i^* προς τον κόμβο i είναι ίση με π_i^{n-1} αφού κάθε απλό μονοπάτι (μονοπάτι που περνά μία και μόνο μία φορά από κάθε ακμή) εμπεριέχει το πολύ $n-1$ ακμές. Παρατηρούμε ότι η εισαγωγή του πλήθους των χρησιμοποιούμενων ακμών κρίνεται απαραίτητη γιατί στην περίπτωση ύπαρξης κύκλου αρνητικού συνολικού βάρους θα οδηγούσε στη διαρκή μείωση του συντομότερου μονοπατιού προς το $-\infty$.

Ένας αλγόριθμος δυναμικού προγραμματισμού μπορεί λοιπόν να ορισθεί απ΄ ευθείας από τη σχέση (7.6). Το πλήθος των υποπροβλημάτων είναι $n(n-1)$.

(Δίδονται K και ο πίνακας $P^k(i)$)

Τελικός κόμβος $x_K; i = K; k = n-1$
επανέλαβε για όλα τα i
    $i \leftarrow P^k(i)$; προηγούμενος κόμβος $x_i; k = k-1$
    αν $i = 1$ τέλος

Σχήμα 7.5: Εύρεση της δομής του συντομότερου μονοπατιού σε οποιονδήποτε γράφο.

- υπολογίζουμε π_i^1 για $i = 1, 2, 3, \ldots, n$ χρησιμοποιώντας το πολύ μία ακμή
  (έχουμε $\pi_i^1 = +\infty$ αν $i \notin \Gamma(1)$)
- υπολογίζουμε π_i^2 σε συνάρτηση των π_i^1 από την (7.6), π_i^3 σε συνάρτηση των π_i^2 και ούτω καθ' εξής,
  Τελικά π_K^{n-1} είναι η τιμή της βέλτιστης λύσης για τον κόμβο K.

Εδώ ακόμη, για να βρούμε τη δομή της βέλτιστης λύσης, ορίζουμε από τη σχέση (7.6) τα διανύσματα $P^k(i)$ ως εξής:

$$P^{k+1}(i) = j_0 \quad \text{αν } \pi_i^{k+1} = \pi_{j_0}^k + l_{j_0 i}, \text{ διαφορετικά } P^{k+1}(i) = P^k(i) \tag{7.7}$$

Από τα διανύσματα $P^1(i), P^2(i), \ldots, P^{n-1}(i)$ μπορούμε να βρούμε τη δομή της βέλτιστης λύσης με τον αλγόριθμο του Σχήματος 7.5.

Ο αλγόριθμος είναι χρόνου $O(Mn)$ αφού χρειάζονται $n-1$ επαναλήψεις και σε κάθε επανάληψη $\sum_{i=1}^n \Gamma^{-1}(i) = M$ συγκρίσεις για την εύρεση της τιμής της βέλτιστης λύσης και $O(n)$ για την εύρεση της δομής της λύσης. Επίσης η απαιτούμενη μνήμη για τον υπολογισμό της τιμής π_K^{n-1} είναι $O(n^2)$ και για την αποθήκευση των $P^k(i)$ είναι επίσης $O(n^2)$. Στην πραγματικότητα όμως αρκεί $O(n)$ χώρος μνήμης και για την εύρεση της τιμής αλλά και τη δομή της βέλτιστης λύσης αντικαθιστώντας τα διανύσματα $P^k(i)$ από το διάνυσμα $P(i)$ ως εξής:

$$P(i) = j_0 \quad \text{αν } \pi_i^{k+1} = \pi_{j_0}^k + l_{j_0 i} \tag{7.8}$$

Η αναζήτηση της δομής του βέλτιστου μονοπατιού απλοποιείται και είναι τότε ίδια με αυτήν σε ένα γράφο χωρίς κύκλο (Σχήμα 7.4).

Συμπερασματικά από το παράδειγμα 1, θα παρατηρήσουμε ότι αν το πρόβλημα δεν είναι εκ φύσεως σειριακό (γράφος περιέχων κύκλους) το πρόβλημα μπορεί να οδηγηθεί σε αυτή τη μορφή με τη χρήση κάποιων τεχνασμάτων (εξίσωση (7.6) π.χ.) αλλά ο χρόνος υπολογισμού, και συχνά επίσης ο χώρος μνήμης, μπορούν να μεγαλώσουν σημαντικά.

Γενικά, όπως είδαμε στα δύο προηγούμενα παραδείγματα, ένας αλγόριθμος δυναμικού προγραμματισμού, θα αποτελείται από δύο φάσεις. Στην πρώτη φάση, διανύουμε τη σειριακή δομή μέχρι να βρούμε την τιμή της βέλτιστης λύσης. Στη δεύτερη φάση, διανύουμε τη σειριακή δομή στην αντίθετη κατεύθυνση ξεκινώντας από τη βέλτιστη τιμή για να καταγράψουμε τη δομή της λύσης. Τα δύο προηγούμενα παραδείγματα είναι πολύ εύκολα. Ας δούμε τώρα ένα πρόβλημα λίγο πιο σύνθετο.

7.3 Το πολυδιάστατο πρόβλημα σακιδίου

Το πρόβλημα υπό μορφή γραμμικού προγραμματισμού γράφεται:

$$\max \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, i = 1, \ldots, m$$

$$x_{ij} \in \{0, 1\}$$
$$c_j, a_{ij}, b_i \in \mathbb{N}$$

- Εμφυτεύουμε το πρόβλημα στην οικογένεια των

$$n(b_1 + 1) \cdot (b_2 + 1) \cdots (b_m + 1)$$

ακολούθων προβλημάτων:

$$P_k(y_1, \ldots, y_m) \equiv \{\max \sum_{j=1}^{k} c_j x_j \ \text{ με } \sum_{j=1}^{k} a_{ij} x_j \leq y_i, \ x_i \in \{0, 1\}, \ i = 1, \ldots, m\}$$

$$\text{με} \quad \left.\begin{array}{c} 0 \le y_1 \le b_1 \\ 0 \le y_2 \le b_2 \\ \vdots \\ 0 \le y_m \le b_m \end{array}\right\} \quad \text{και } 1 \le k \le n$$

- Αν $f_k(y_1, y_2, \ldots, y_m)$ είναι η τιμή της βέλτιστης λύσης του προβλήματος $P_k(y_1, y_2, \ldots, y_m)$ έχουμε την αναδρομική σχέση:

$$f_{k+1}(y_1, \ldots, y_m) = \max\{f_k(y_1, \ldots, y_m), f_k(y_1 - a_{1,k+1}, \ldots, y_m - a_{m,k+1}) + c_{k+1}\} \tag{7.9}$$

για $a_{i,k+1} \le y_i \le b_i, 1 \le i \le m$

Ο αλγόριθμος δυναμικού προγραμματισμού που βασίζεται στην αναδρομική σχέση (7.9) θα είναι χρονικής πολυπλοκότητας

$$O\left(n \prod_{i=1}^{m} (b_i + 1)\right)$$

και πολυπλοκότητας μνήμης

$$O\left(n \prod_{i=1}^{m} (b_i + 1)\right)$$

Εν τούτοις, μπορούμε να ελαττώσουμε το χώρο μνήμης σε:

$$O\left(\prod_{i=1}^{m} (b_i + 1)\right)$$

Παρ΄ όλα αυτά, ο αλγόριθμος δυναμικού προγραμματισμού παραμένει για το παράδειγμα 3 πολύ ακριβός και σε χρόνο και σε μνήμη.

7.4 Μείωση διάστασης προβλημάτων συνδυαστικής βελτιστοποίησης

Το μειονέκτημα αυξημένου χρόνου και μνήμης, εμφανίζεται συχνά σε πολλά προβλήματα συνδυαστικής όταν επιχειρούμε να τα επιλύσουμε με την τεχνική του δυναμικού προγραμματισμού. Σε μερικές όμως περιπτώσεις μπορούμε να μειώσουμε αισθητά το χρόνο και την απαιτούμενη μνήμη. Μια πρώτη ιδέα είναι να περιορίσουμε το διάστημα μεταβολής των τιμών των μεταβλητών. Η μέθοδος ή πιο απλή θα είναι να οριστικοποιήσουμε μερικές μεταβλητές, όπως κάνουμε στις μεθόδους της δενδρικής αναζήτησης με διαχωρισμό και εκτίμηση. Ας θεωρήσουμε το συνδυαστικό πρόβλημα:

$$\begin{cases} \min z = cx \\ \\ x \in X \\ \\ X = \{x | Ax = b, x \geq 0\} \bigcap Y \end{cases}$$

όπου Y είναι ένα διακριτό σύνολο π.χ. Z^n ή $\{0,1\}^n$

Ας υποθέσουμε ότι γνωρίζουμε μια εφικτή λύση $\hat{x}$ με τη βοήθεια ενός ευριστικού αλγόριθμου.

Αν u^* είναι η λύση του γραμμικού προγράμματος:

$$\begin{cases} \max ub \\ \\ uA \leq c \end{cases}$$

τότε η z γράφεται:

$$z = u^*b + (c - u^*A)x = u^*b + c'x \text{ με } c'_j \geq 0 \text{ όπου } c'_j = c_j - u^*A_j$$

Μπορούμε από εδώ να συνάγουμε άνω φράγματα για τις μεταβλητές x_j. Πράγματι, αφού για κάθε λύση x καλύτερη από την $\hat{x}$, πρέπει να έχουμε

$$u^*b + c'x < z(\hat{x})$$

συμπεραίνουμε ότι $c'x < z(\hat{x}) - u^*b$ δηλαδή $\forall j \quad c'_j x_j < z(\hat{x}) - u^*b$ και επομένως

αφού η x_j παίρνει ακέραιες τιμές θα πρέπει:

$$x_j \leq \left\lfloor \frac{z(\hat{x}) - u^* b}{c'_j} \right\rfloor$$

Ειδικότερα, έχουμε $x_j = 0$ για όλα τα j τέτοια ώστε

$$c'_j > z(\hat{x}) - u^* b$$

Η οριστικοποίηση σε μηδέν αυτών των μεταβλητών μπορεί να οδηγήσει εκ νέου στην οριστικοποίηση άλλων μεταβλητών. Θα εφαρμόσουμε τελικά το δυναμικό προγραμματισμό στο μειωμένο έτσι πρόβλημα.

Ας δούμε ένα απλό παράδειγμα για αυτή τη μέθοδο ελάττωσης της διάστασης.

Έστω το παρακάτω στιγμιότυπο του προβλήματος του σακιδίου:

$$\max z = 14x_1+13x_2+15x_3+10x_4+8x_5+13x_6+10x_7+10x_8+5x_9+7x_{10}+7x_{11}+6x_{12}$$

με

$$7x_1 + 7x_2 + 10x_3 + 7x_4 + 6x_5 + 10x_6 + 8x_7 + 9x_8 + 5x_9 + 10x_{10} + 12x_{11} + 13x_{12} \leq 53$$

Με ένα γνωστό από τα προηγούμενα ευριστικό αλγόριθμο ευρίσκουμε την προσεγγιστική λύση $\hat{x}$ με $z(\hat{x}) = 78$ (βλ. 4.8) με διάταξη των $\frac{c_j}{a_j}$ κατά φθίνουσα σειρά αφού πρόκειται για μεγιστοποίηση.

Η συνεχής βέλτιστη λύση δίνει $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 1, x_6 = 1$ και $x_7 = \frac{6}{8}$ με τιμή $73 + \frac{6}{8} \times 10 = 80.5 = u^* b$. Η z τότε γράφεται υπολογίζοντας $c'_j = c_j - u^* A_j \ \forall j$ και $u^* = \frac{10}{8}$ (πηλίκο της μεταβλητής x_7):

$$\begin{aligned} z = \ & 80,5 - 5,25\bar{x}_1 - 4,5\bar{x}_2 - 2,5\bar{x}_3 - 1,25\bar{x}_4 - 0,5\bar{x}_5 - 0,5\bar{x}_6 - \\ & -1,25x_8 - 1,25x_9 - 5,5x_{10} - 8x_{11} - 10,25x_{12} - 1,25y \end{aligned}$$

όπου $\bar{x}_j = 1 - x_j$ και όπου $y \geq 0$ είναι η μεταβλητή απόκλισης.

Μπορούμε λοιπόν να οριστικοποιήσουμε στο μηδέν τις μεταβλητές των οποίων το μειωμένο κόστος είναι πιο μεγάλο (ή ίσο αν αναζητούμε όλες τις βέλτιστες λύσεις) από

80,5 - 78 = 2,5.

Θα οριστικοποιήσουμε λοιπόν

$$x_1 = 1(\bar{x}_1 = 0), x_2 = 1(\bar{x}_2 = 0)$$
$$x_3 = 1(\bar{x}_3 = 0), x_{10} = 0, x_{11} = 0 \text{ και } x_{12} = 0$$

Θα επιλύσουμε επομένως με δυναμικό προγραμματισμό το μειωμένο πρόβλημα σακιδίου:

$$\max z = 10x_4 + 8x_5 + 13x_6 + 10x_7 + 10x_8 + 5x_9$$

$$7x_4 + 6x_5 + 10x_6 + 8x_7 + 9x_8 + 5x_9 \leq 29$$

Το μέγεθος του προβλήματος πέρασε από $n = 12$ σε $n = 6$ και το δεύτερο μέλος από $b = 53$ στο $b = 29$. Θα κερδίσουμε λοιπόν σχεδόν μια τάξη 4 πάνω στο δυναμικό προγραμματισμό. Για προβλήματα σοβαρών διαστάσεων, το κέρδος μπορεί να είναι ιδιαίτερα αισθητό.

Εν τούτοις, παρά τη μείωση του προβλήματος, ο δυναμικός προγραμματισμός μπορεί να οδηγήσει σε χρόνους υπολογισμού και απαιτήσεις μνήμης υπερβολικές. Για να βελτιώσουμε την αποτελεσματικότητά του είναι ενδιαφέρον να τον συνδυάσουμε με τις μεθόδους της δενδρικής αναζήτησης με διαχωρισμό και εκτίμηση.

Κεφάλαιο 8

Τοπική Αναζήτηση (Local Search)

8.1 Εισαγωγή

Η Τοπική Αναζήτηση είναι μια γενική μέθοδος η οποία χρησιμοποιείται ευρύτατα σήμερα, κυρίως στην αντιμετώπιση των NP-δύσκολων προβλημάτων βελτιστοποίησης (βλ. Κεφάλαιο 2). Η ιδέα είναι αρκετά απλή και συνίσταται στο εξής: Αφού στα NP-δύσκολα προβλήματα είναι μάλλον απίθανο να βρούμε κάποιον αποδοτικό αλγόριθμο που να υπολογίζει την καλύτερη από όλες τις εφικτές λύσεις του προβλήματος, ας προσπαθήσουμε να βρούμε μία λύση που αν τη διαταράξουμε, με κάποιο καθορισμένο τρόπο, να μην παράγεται κάποια άλλη λύση καλύτερη από αυτή. Ο κλασικός αλγόριθμος τοπικής αναζήτησης κάνει ακριβώς αυτό. Ξεκινάει από μία αυθαίρετη λύση, τη διαταράσσει και αν προκύψει καλύτερη λύση επαναλαμβάνει τη διαταραχή στη νέα λύση αλλιώς τερματίζει.

Διαισθητικά η έννοια της διαταραχής μπορεί να φανεί στο εξής παράδειγμα. Έστω ότι η φύση ψάχνει για τον οργανισμό που έχει τη μεγαλύτερη ικανότητα να επιβιώσει στο περιβάλλον. Ξεκινάει από κάποιους αρχικούς οργανισμούς με απλό DNA και μέσω του ζευγαρώματος και της διαδικασίας της φυσικής επιλογής παράγει καλύτερους, ως προς την προσαρμοστικότητά τους, οργανισμούς. Το DNA κάθε οργανισμού αποτελεί μία λύση (του προβλήματος της φύσης) και το ζευγάρωμα παίζει το ρόλο της διαταραχής, η οποία εδώ συμβαίνει με το συνδυασμό δύο λύσεων. Παρόμοια μέθοδο προσπαθούν να προσομοιώσουν οι γενετικοί αλγόριθμοι για να επιλύσουν προβλήματα συνδυαστικής βελτιστοποίησης.

Πιο τυπικά, ένα *πρόβλημα τοπικής αναζήτησης* Π ορίζεται ως ένα σύνολο στιγμιοτύπων $\mathcal{I}_\Pi$, τέτοιο ώστε για κάθε στιγμιότυπο $x \in \mathcal{I}_\Pi$ να υπάρχει

- ένα *σύνολο εφικτών λύσεων* $\mathcal{J}_\Pi(x)$,
- μία *συνάρτηση κόστους* $f_\Pi : \mathcal{J}_\Pi(x) \to \mathcal{Q}_+$, η οποία για κάθε εφικτή λύση $i \in \mathcal{J}_\Pi(x)$ να επιστρέφει το *κόστος* της $f_\Pi(i, x)$
- μία *συνάρτηση γειτονιάς* $\mathcal{N}_\Pi : \mathcal{J}_\Pi(x) \to 2^{\mathcal{J}_\Pi(x)}$, η οποία για κάθε εφικτή λύση $i \in \mathcal{J}_\Pi(x)$ να επιστρέφει τις γειτονικές της λύσεις.[1]

Το ερώτημα σε αυτά τα προβλήματα, όταν πρόκειται για ελαχιστοποίηση, είναι: ''Δεδομένου ενός στιγμιοτύπου $x \in \mathcal{I}_\Pi$, βρείτε μία λύση $\hat{\imath} \in \mathcal{J}_\Pi(x)$, τέτοια ώστε $f_\Pi(\hat{\imath}, x) \leq f_\Pi(i, x)$ για κάθε $i \in \mathcal{N}_\Pi(\hat{\imath}, x)$''. Στις περιπτώσεις μεγιστοποίησης πρέπει $f_\Pi(\hat{\imath}, x) \geq f_\Pi(i, x)$.

Ο *κλασικός ευριστικός αλγόριθμος τοπικής αναζήτησης* για ένα πρόβλημα ελαχιστοποίησης $x \in \mathcal{I}_\Pi$ είναι ο εξής:

1. Βρες μία αρχική λύση $s \in \mathcal{J}_\Pi(x)$
2. Όσο υπάρχει $s' \in \mathcal{N}_\Pi(s, x)$ με $f_\Pi(s', x) < f_\Pi(s, x)$ τότε
   $s := s'$
3. Επέστρεψε s

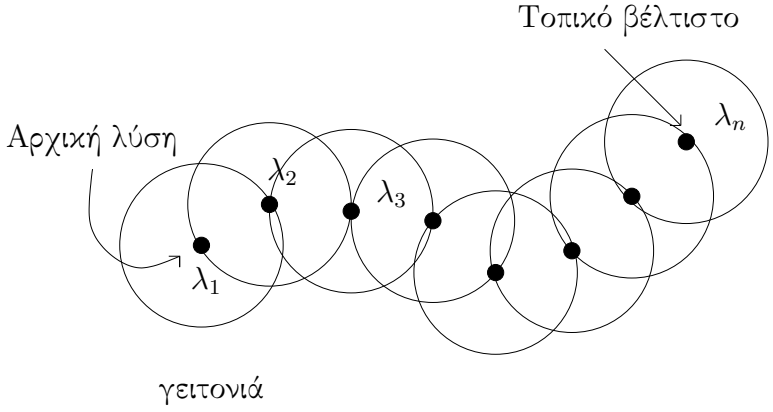
Σχήμα 8.1: Κλασικός αλγόριθμος τοπικής αναζήτησης

Μία απλοϊκή αναπαράσταση των εννοιών που αναφέραμε παραπάνω φαίνεται στο Σχήμα 8.2. Στο (α) υποθέτουμε ότι οι λύσεις ενός προβλήματος είναι σημεία στο επίπεδο. Κάθε κύκλος συμβολίζει τα όρια της γειτονιάς της λύσης που βρίσκεται στο κέντρο του. Ο ευριστικός αλγόριθμος τοπικής αναζήτησης θα ξεκινούσε από την αρχική λύση λ_1, θα εξέταζε τις λύσεις μέσα στα όρια του κύκλου-γειτονιάς, θα επέλεγε μία καλύτερη και με βάση αυτήν θα επαναλάμβανε τη διαδικασία. Εδώ, όπως βλέπουμε, ο αλγόριθμος σταμάτησε στη λύση λ_n, η οποία έχει το χαρακτηριστικό ότι καμία γειτονική της λύση δεν είναι καλύτερη.

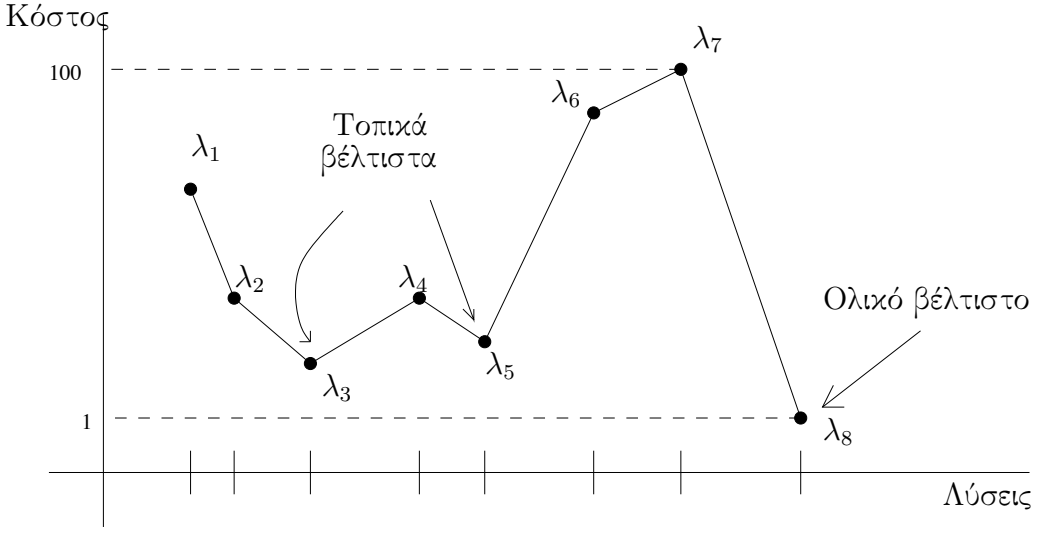
Στο (b) μπορούμε να δούμε καλύτερα τι εννοούμε με τη φράση ''καλύτερη γειτονική λύση''. Έστω τώρα ότι κάθε λύση (σημείο στο επίπεδο) έχει δύο το πολύ γείτονες,

[1] Το $2^{\mathcal{J}}$ σημαίνει: ''Όλα τα υποσύνολα του $\mathcal{J}$''.

έτσι ώστε να μπορούμε να τοποθετήσουμε τις λύσεις στη σειρά. Το ύψος κάθε λύσης αντιπροσωπεύει το κόστος της. Επομένως στη συγκεκριμένη περίπτωση ο ευριστικός αλγόριθμος, αν θέλαμε να ελαχιστοποιήσουμε το κόστος και ξεκινούσαμε από την λύση λ_1, θα κατέληγε στην τοπικά βέλτιστη λύση λ_3. Παρατηρήστε ότι η ολικά βέλτιστη λύση είναι η λ_8. Με αυτήν τη γειτονιά και αυτόν τον αλγόριθμο θα μπορούσαμε, ίσως, να τη βρίσκαμε αν παίρναμε αρχική λύση τη λ_7.



(α) Διάταξη λύσεων στο επίπεδο



(β) Διάταξη λύσεων στην ευθεία

Σχήμα 8.2: Απλές διατάξεις λύσεων.

Όταν ο σκοπός μας είναι να επιλύσουμε ένα πρόβλημα βελτιστοποίησης, το οποίο δεν φαίνεται να λύνεται πολυωνυμικά, και χρησιμοποιούμε έναν αλγόριθμο τοπικής αναζήτησης, επιθυμούμε να γνωρίζουμε την πολυπλοκότητα αυτού του αλγόριθμου καθώς και το πόσο κοντά στο βέλτιστο θα είναι η λύση που θα μας δώσει. Αν και πειραματικά οι αλγόριθμοι τοπικής αναζήτησης συγκλίνουν πολύ γρήγορα επιστρέφοντας αρκετά καλής ποιότητας λύσεις, αποδεικνύεται ότι υπάρχουν προβλήματα για τα οποία ο αλγόριθμος απαιτεί εκθετικό χρόνο, αλλά και ότι οι λύσεις μπορεί να είναι οσοδήποτε μακριά από το βέλτιστο. Αντίθετα, υπάρχουν πολλά προβλήματα για τα οποία αποδεικνύεται ότι ο αλγόριθμος επιστρέφει λύσεις κοντά στο βέλτιστο, σε πολυωνυμικό χρόνο. Αυτά τα ζητήματα θα μελετήσουμε στις δύο επόμενες ενότητες, ξεκινώντας από την πολυπλοκότητα της μεθόδου.

Σε αυτό το σημείο πρέπει να τονιστεί η διαφορά της *πολυπλοκότητας ενός προβλήματος τοπικής αναζήτησης* (δηλαδή της εύρεσης ενός τοπικού βέλτιστου με οποιοδήποτε τρόπο) και της *πολυπλοκότητας του ευριστικού αλγορίθμου τοπικής αναζήτησης*, (δηλαδή της εύρεσης τοπικού βέλτιστου με τον κλασικό επαναληπτικό αλγόριθμο). Για παράδειγμα, μπορεί ένα πρόβλημα να λύνεται σε πολυωνυμικό χρόνο, ενώ ένας ευριστικός αλγόριθμός τοπικής αναζήτησης να έχει εκθετική πολυπλοκότητα στη χείριστη περίπτωση.

Χαρακτηριστικό είναι το παράδειγμα του Γραμμικού Προγραμματισμού. Υπάρχουν άμεσες μέθοδοι, όπως οι αλγόριθμοι των ελλειψοειδών ή του εσωτερικού σημείου, οι οποίες λύνουν το πρόβλημα σε πολυωνυμικό χρόνο. Όπως γνωρίζετε, ο Γραμμικός Προγραμματισμός μπορεί να θεωρηθεί και ως ένα πρόβλημα συνδυαστικής βελτιστοποίησης του οποίου οι λύσεις είναι οι κορυφές ενός πολυτόπου, αυτού που σχηματίζεται από τους περιορισμούς. Αν θεωρήσουμε γειτονικές τις κορυφές που συνδέονται με μία ακμή του πολυτόπου τότε ο αλγόριθμος τοπικής αναζήτησης που προκύπτει είναι η Simplex, η οποία κινείται από κορυφή σε κορυφή. Στο συγκεκριμένο πρόβλημα το τοπικό βέλτιστο συμπίπτει με το ολικό και ο κλασικός αλγόριθμος τοπικής αναζήτησης έχει δειχθεί ότι απαιτεί έναν εκθετικό αριθμό επαναλήψεων στη χείριστη περίπτωση (για τους περισσότερους κανόνες μετάβασης). Παρ΄ όλ΄ αυτά ο τελευταίος προτιμάται αφού στην πράξη τελειώνει πολύ γρηγορότερα από τις πολυωνυμικές μεθόδους! Γενικότερα οι αλγόριθμοι τοπικής αναζήτησης είναι γρηγορότεροι από αυτούς των ελλειψοειδών και του εσωτερικού σημείου και προτιμώνται έναντι των τελευταίων. Σημειώστε ότι οι γειτονιές που έχουν την ιδιότητα τα τοπικά τους βέλτιστα να είναι και ολικά ονομάζονται

ακριβείς (exact).

Παρομοίως οι αλγόριθμοι που χρησιμοποιούνται για την επίλυση των maximum matching, maximum weighted matching, minimum cost perfect matching, maximum flow και minimum cost flow είναι αλγόριθμοι τοπικής αναζήτησης και η γειτονιές που χρησιμοποιούν είναι ακριβείς.

Ας δούμε τώρα ένα σύνηθες πρόβλημα βελτιστοποίησης και κάποιες από τις γειτονιές που χρησιμοποιούνται για την επίλυσή του.

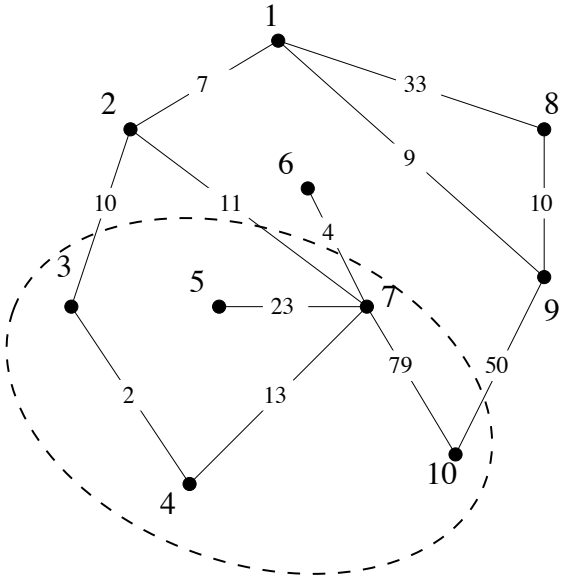
8.1.1 Διαμέριση Γράφου

Το πρόβλημα της Διαμέρισης Γράφου (Graph Partitioning) είναι το εξής: Δεδομένου ενός γράφου $G = (V, E)$ με $2n$ κόμβους, που σε κάθε ακμή έχει ένα βάρος $w_e \in \mathbb{Z}^+$, να βρεθεί μια διαμέριση του συνόλου των κόμβων V σε δύο σύνολα A, B με $|A| = |B| = n$, τέτοια ώστε να ελαχιστοποιείται (ή να μεγιστοποιείται[2]) το κόστος της τομής $w(A, B)$. Ένα στιγμιότυπο x για αυτό το πρόβλημα αποτελείται από μια κωδικοποίηση του γράφου G (π.χ. από τον πίνακα γειτνίασής του) και μια λίστα των βαρών των ακμών του w_e, γραμμένα σε δυαδική μορφή. Το κόστος μιας λύσης i (μιας διαμέρισης των κόμβων σε δύο ίσα σύνολα) καθορίζεται από την είσοδο x και την λύση i. Γι΄ αυτό το λόγο το x εμφανίζεται ως όρισμα και στο σύνολο των λύσεων $\mathcal{J}$ και στην συνάρτηση κόστους f. Καμιά φορά παραλείπουμε τον δείκτη Π όταν είναι προφανές από τα συμφραζόμενα σε ποιο πρόβλημα αναφερόμαστε.

Για παράδειγμα, έστω ο γράφος $G = (V, E, W)$ του Σχήματος 8.3 και η διαμέριση (A, B) του γράφου, όπου $A = \{3, 4, 5, 7, 10\}$ και $B = \{1, 2, 6, 8, 9\}$. Το κόστος της διαμέρισης που θέλουμε να ελαχιστοποιήσουμε ή να μεγιστοποιήσουμε, ανάλογα, είναι $w(A, B) = 10 + 11 + 4 + 50 = 75$. Το ερώτημα είναι, για ποια διαμέριση (A^*, B^*) του γράφου το $w(A^*, B^*)$ είναι ελάχιστο (ή μέγιστο);

Ορίζοντας, τώρα, μία συνάρτηση γειτονιάς στο σύνολο των λύσεων του προβλήματος συνδυαστικής βελτιστοποίησης, θα προκύψει ένα πρόβλημα τοπικής αναζήτησης. Πιο συγκεκριμένα, δεδομένου ενός στιγμιοτύπου x, σε κάθε λύση $i \in \mathcal{J}_\Pi(x)$ αναθέτουμε ένα σύνολο $\mathcal{N}_\Pi(i, x) \subseteq \mathcal{J}_\Pi(x)$ γειτονικών λύσεων. Οι γείτονες καθορίζονται με βάση το εκάστοτε στιγμιότυπο x και την εκάστοτε λύση i. Αν για ένα πρόβλημα συν-

[2] Δεν κάνουμε διάκριση μεταξύ ελαχιστοποίησης και μεγιστοποίησης γιατί τα δύο προβλήματα είναι ισοδύναμα.

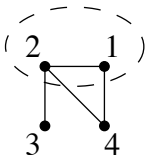


Σχήμα 8.3: Παράδειγμα Διαμέρισης Γράφου

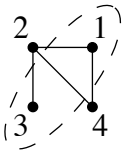
δυαστικής βελτιστοποίησης χρησιμοποιηθούν διαφορετικές συναρτήσεις γειτονιάς θα προκύψουν διαφορετικά προβλήματα τοπικής αναζήτησης, γι΄ αυτό και τα συμβολίζουμε με ΠΣΒ/$\mathcal{N}$.

Παρόλο που ο όρος γειτονιά υποδηλώνει ότι οι γειτονικές λύσεις βρίσκονται κατά μία έννοια ''κοντά'', δηλαδή ότι μπορούμε να πάρουμε την μία από την άλλη με κάποια μικρή ''διατάραξη'', αυτό δεν είναι απόλυτο. Η συνάρτηση γειτονιάς μπορεί να είναι αρκετά πολύπλοκη, ενώ δεν χρειάζεται να είναι ούτε καν συμμετρική. Είναι πιθανό μία λύση i να έχει γείτονα μία άλλη λύση j, αλλά η j να μην έχει γείτονα την i. Για παράδειγμα στο συγκεκριμένο πρόβλημα, της Διαμέρισης Γράφου, μια πολύ απλή γειτονιά είναι η *Swap*, η οποία ορίζεται ως εξής: μία ισομερής διαμέριση (A, B) του συνόλου των κόμβων V έχει για γείτονές της όλες τις διαμερίσεις που παράγονται από την εναλλαγή ενός κόμβου του A με ένα κόμβο του B. Στο Σχήμα 8.4 βλέπουμε τους γείτονες της διαμέρισης (a) ενός απλού γράφου σύμφωνα με τη γειτονιά Swap. Οι γείτονες της (a) είναι οι (b),(c),(d),(e) ενώ η (f) δεν είναι γείτονας αφού προκύπτει από την εναλλαγή και των δύο στοιχείων του (a).[3]

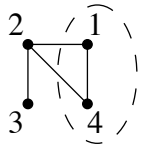
[3]Λόγω της απλότητας του συγκεκριμένου γράφου υπάρχει μια συμμετρία στις διαμερίσεις. Έτσι, στο συγκεκριμένο παράδειγμα, οι λύσεις (b),(e) και (c),(d) είναι όμοιες και κατά συνέπεια έχουν το



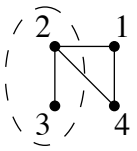
(a) (A={1,2}, B={3,4})



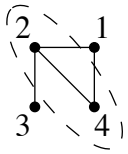
(b) (A={1,3}, B={2,4})



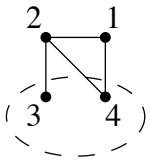
(c) (A={1,4}, B={2,3})



(d) (A={2,3}, B={1,4})



(e) (A={2,4}, B={1,3})



(f) (A={3,4}, B={1,2})

ΟΧΙ ΓΕΙΤΟΝΑΣ

Σχήμα 8.4: Παράδειγμα γειτονιάς μιας διαμέρισης γράφου. Οι λύσεις (b), (c), (d) και (e) είναι γείτονες της λύσης (a).

Μια πολύ πιο σύνθετη γειτονιά εξερευνάται από τον ευριστικό αλγόριθμο των Kernighan και Lin. Αυτή η γειτονιά δεν είναι συμμετρική και εξαρτάται από τα βάρη των ακμών. Ο ευριστικός των Kernighan-Lin (KL για συντομία) κινείται από μία διαμέριση (A, B) σε μια γειτονική διαμέριση μέσα από μία ακολουθία ''άπλειστων'' (greedy) εναλλαγών. Σε κάθε βήμα της ακολουθίας επιλέγουμε να εναλλάξουμε το καλύτερο δυνατό ζευγάρι κόμβων ανάμεσα σε αυτά τα ζευγάρια που δεν έχουν μετακινηθεί σε προηγούμενα βήματα της ακολουθίας. Με το ''καλύτερο'' εννοούμε ότι η εναλλαγή παράγει την ''ελάχιστη διαφορά κόστους'', δηλαδή το βάρος της τομής ελαττώνεται όσο είναι δυνατόν ή έστω αυξάνεται όσο λιγότερο γίνεται, όταν έχουμε προβλήματα ελαχιστοποίησης. Το αντίθετο συμβαίνει σε προβλήματα μεγιστοποίησης. Σε περίπτωση ισοπαλίας για την καλύτερη λύση ενός βήματος, ένας tie-breaking κανόνας χρησιμοποιείται για να επιλεγεί ένα μοναδικό ζευγάρι κόμβων. Έτσι από μία διαμέριση (A, B) παράγεται μια ακολουθία από διαμερίσεις (A_i, B_i), $i = 1, \ldots, n$, όπου $|A_i - A| = |B_i - B|$. Όλες αυτές οι διαμερίσεις είναι γείτονες της αρχικής διαμέρισης (A, B).

Προφανώς αυτή είναι μία ισχυρότερη γειτονιά από την απλή Swap. Παρατηρούμε

ίδιο κόστος. Επίσης και η (f) είναι στην ουσία η ίδια διαμέριση με την (a).

ότι αν μία διαμέριση είναι τοπικά βέλτιστη κάτω από την Kernighan-Lin τότε θα είναι τοπικά βέλτιστη και κάτω από τη Swap, επειδή στην αντίθετη περίπτωση ο πρώτος της γείτονας θα ήταν καλύτερος. Επομένως το να βρει κανείς τοπικά βέλτιστα κάτω από την Kernighan-Lin είναι τουλάχιστον τόσο δύσκολο όσο και κάτω από τη Swap. Προκαλεί έκπληξη βέβαια, όπως θα δούμε, ότι τα δύο προβλήματα είναι στην πραγματικότητα πολυωνυμικά ισοδύναμα.

Γενικά όσο πιο ισχυρή είναι μία γειτονιά τόσο δυσκολότερο είναι να εξερευνηθεί και να βρεθούν τοπικά βέλτιστα, αλλά πιθανώς τα βέλτιστα να έχουν καλύτερη ποιότητα. Όπως είδαμε προηγουμένως, η πιο ισχυρή γειτονιά είναι μία ακριβής (exact) γειτονιά. Σε μια ακριβή γειτονιά, όπου ισχύει ότι κάθε τοπικό βέλτιστο είναι και ολικό βέλτιστο, το πρόβλημα τοπικής αναζήτησης ταυτίζεται με το πρόβλημα βελτιστοποίησης. Σε κάθε πρόβλημα βελτιστοποίησης, βέβαια, μπορεί κάποιος να κάνει τα τοπικά με τα ολικά βέλτιστα να συμπέσουν επιλέγοντας αρκετά μεγάλες γειτονιές. Το πρόβλημα, όμως, είναι ότι τότε μπορεί να είναι δύσκολο να εξερευνηθεί η γειτονιά. Δηλαδή το να καθορισθεί αν μια λύση είναι τοπικά βέλτιστη και εάν όχι να βρεθεί ένας καλύτερος γείτονάς της, να απαιτεί εκθετικό χρόνο. Επομένως μια βασική απαίτηση είναι να μπορεί η γειτονιά να εξερευνάτε αποτελεσματικά.

Μια συνάρτηση γειτονιάς δεν καθορίζει μοναδικά τον αλγόριθμο τοπικής αναζήτησης. Μερικές λύσεις μπορεί να έχουν περισσότερους από έναν καλύτερους γείτονες, οπότε ένας αλγόριθμος έχει την δυνατότητα να επιλέξει όποιον από αυτούς τους γείτονες θέλει για να κινηθεί. Ένας κανόνας που αναθέτει έναν καλύτερο γείτονα σε κάθε λύση που δεν είναι τοπικά βέλτιστη λέγεται *κανόνας μετάβασης (pivoting rule)*. Η επιλογή ενός κανόνα μετάβασης μπορεί να επηρεάσει δραστικά την πολυπλοκότητα ενός αλγορίθμου τοπικής αναζήτησης. Για παράδειγμα θεωρείστε το Πρόβλημα Μέγιστης Ροής με συνάρτηση γειτονιάς αυτή που χρησιμοποιεί την προσαύξηση μέσω ενός μονοπατιού. Είναι γνωστό ότι αν διαλέγουμε βελτιώνουσες αλυσίδες τυχαία τότε μπορεί να απαιτηθεί εκθετικό πλήθος προσαυξήσεων. Αν, όμως, πάντα προσαυξάνουμε μέσω των μικρότερων μονοπατιών, το βέλτιστο θα βρεθεί μετά από πολυωνυμικό πλήθος επαναλήψεων. Έτσι, στην εξέταση των αλγορίθμων τοπικής αναζήτησης, θέλουμε να αναλύσουμε την πολυπλοκότητά τους για διαφορετικούς κανόνες μετάβασης και να βρούμε τον καλύτερο δυνατό κανόνα.

8.2 Πολυπλοκότητα

Καταρχήν, ας παρατηρήσουμε ότι ο κλασικός αλγόριθμος τοπικής αναζήτησης είναι ψευδοπολυωνυμικός. Αυτό σημαίνει ότι η πολυπλοκότητά του στη χείριστη περίπτωση εξαρτάται από τους αριθμούς (βάρη, κόστη) που υπάρχουν στο στιγμιότυπο. *Εάν το πλήθος των διαφορετικών κοστών των λύσεων είναι πολυωνυμικό τότε θα μπορεί να γίνει το πολύ πολυωνυμικό πλήθος επαναλήψεων και ο αλγόριθμος θα τερματίζει σε πολυωνυμικό χρόνο* (αυτό ισχύει αφού, όπως είπαμε, σε κάθε βήμα πάμε πάντα σε έναν αυστηρά καλύτερο γείτονα). Παράδειγμα είναι οι μη βεβαρημένες εκδοχές των περισσότερων προβλημάτων βελτιστοποίησης, όπως η Διαμέριση Γράφου, που είδαμε προηγουμένως, χωρίς βάρη όμως στις ακμές του γράφου. Στη γενική περίπτωση όπου υπάρχει ένα εκθετικό εύρος κοστών λύσεων δεν υπάρχει εκ των προτέρων κανένα φράγμα στο πλήθος των επαναλήψεων καλύτερο από το εκθετικό. Σε μια τέτοια περίπτωση θέλουμε να ξέρουμε αν ένα πρόβλημα τοπικής αναζήτησης λύνεται πολυωνυμικά με κάποιο τρόπο ή όχι.

Γενικά, για πολλά ενδιαφέροντα προβλήματα, η πολυπλοκότητα της εύρεσης μιας τοπικά βέλτιστης λύσης παραμένει ανοιχτή, δηλαδή δεν ξέρουμε αν μπορεί να γίνει σε πολυωνυμικό χρόνο ή όχι. Το 1988 οι Johnson, Παπαδημητρίου και Γιαννακάκης εισήγαγαν την κλάση πολυπλοκότητας PLS (Polynomial-Time Local Search) για να ομαδοποιήσουν αυτά τα προβλήματα. *Τα προβλήματα που δεν ξέρουμε αν λύνονται πολυωνυμικά, αλλά που η γειτονιά τους μπορεί να αναζητηθεί πολυωνυμικά, ανήκουν στην PLS* (δηλαδή το polynomial-time της PLS αφορά την αναζήτηση της γειτονιάς). Αυτή είναι μια ελάχιστη απαίτηση που ικανοποιείται από τις γειτονιές που χρησιμοποιούνται από τους συνήθεις ευριστικούς αλγόριθμους τοπικής αναζήτησης.

Έπομένως η κλάση PLS περιλαμβάνει τα προβλήματα που έχουν τις ακόλουθες κοινές ιδιότητες:

- Βρίσκουμε αρχικές λύσεις σε πολυωνυμικό χρόνο,
- Υπολογίζουμε τα κόστη των λύσεων σε πολυωνυμικό χρονο και
- Ψάχνουμε μια γειτονιά ''εύκολα'' δηλαδή σε πολυωνυμικό χρόνο.

Έχει δειχθεί ότι πολλά σημαντικά προβλήματα τοπικής αναζήτησης είναι πλήρη για την PLS κάτω από μία κατάλληλα ορισμένη αναγωγή. Γι΄ αυτό και η PLS χαρακτηρίζει την πολυπλοκότητα των προβλημάτων τοπικής αναζήτησης με την ίδια έννοια που

η NP χαρακτηρίζει την πολυπλοκότητα των ''δύσκολων'' προβλημάτων συνδυαστικής βελτιστοποίησης. Επιπλέον, η θεωρία της PLS-πληρότητας μας δίνει τη δυνατότητα να αναλύουμε την πολυπλοκότητα πολλών δημοφιλών ευριστικών αλγόριθμων τοπικής αναζήτησης, όπως θα δούμε στη συνέχεια.

Τυπικά η PLS ορίζεται ως εξής: Ας θεωρήσουμε ένα πρόβλημα Τοπικής Αναζήτησης Π. Υποθέτουμε ότι τα στιγμιότυπά του είναι κωδικοποιημένα σε δυαδικές συμβολοσειρές και ότι για κάθε στιγμιότυπο x, οι λύσεις του $s \in \mathcal{J}_\Pi(x)$ είναι επίσης δυαδικές συμβολοσειρές, με μήκος φραγμένο από ένα πολυώνυμο του μήκους του x. Χωρίς απώλεια της γενικότητας μπορούμε να υποθέσουμε ότι όλες οι λύσεις είναι κωδικοποιημένες σαν συμβολοσειρές του ίδιου μήκους $p(|x|)$. Επίσης υποθέτουμε για απλοποίηση ότι τα κόστη είναι μη αρνητικοί ακέραιοι (η θεωρία επεκτείνεται άμεσα και σε ρητά κόστη).

Ορισμός 8.2.1 (PLS) *Ένα πρόβλημα Τοπικής Αναζήτησης* Π *ανήκει στην PLS εάν υπάρχουν τρεις πολυωνυμικού χρόνου αλγόριθμοι* A_Π, B_Π, C_Π *με τις ακόλουθες ιδιότητες:*

1. *Δεδομένης μιας συμβολοσειράς* $x \in \{0,1\}^*$*, ο αλγόριθμος* A_Π *καθορίζει εάν το* x *είναι ένα στιγμιότυπο του* Π *και σε αυτήν την περίπτωση παράγει μια λύση* $s_0 \in \mathcal{J}_\Pi(x)$*.*

2. *Δεδομένου ενός στιγμιοτύπου* x *και μιας συμβολοσειράς* s*, ο αλγόριθμος* B_Π *καθορίζει εάν η* $s \in \mathcal{J}_\Pi(x)$ *και αν ναι, τότε υπολογίζει το κόστος* $f_\Pi(s,x)$ *της λύσης* s*.*

3. *Τέλος, δεδομένου ενός στιγμιοτύπου* x *και μιας λύσης* s*, ο αλγόριθμος* C_Π *καθορίζει εάν η* s *είναι τοπικό βέλτιστο, και στην περίπτωση που δεν είναι, ο* C_Π *δίνει ένα γείτονα* $s' \in \mathcal{N}_\Pi(s,x)$ *με αυστηρά καλύτερο κόστος, δηλαδή* $f_\Pi(s',x) < f_\Pi(s,x)$ *για ένα πρόβλημα ελαχιστοποίησης και* $f_\Pi(s',x) > f_\Pi(s,x)$ *για ένα πρόβλημα μεγιστοποίησης.*

Από τον ορισμό μπορούμε να κατασκευάσουμε έναν αλγόριθμο τοπικής αναζήτησης ο οποίος ξεκινάει με αρχική λύση $s_0 = A_\Pi(x)$ και εφαρμόζει επαναληπτικά τον αλγόριθμο C_Π μέχρι να φτάσει σε ένα τοπικό βέλτιστο.

Σημειώστε ότι η Διαμέριση Γράφου, όπως και όλα τα συνήθη προβλήματα τοπικής αναζήτησης, ανήκουν στην PLS.

8.2.1 Που ανήκει η PLS

Η κλάση PLS βρίσκεται κάπου ανάμεσα στις γνωστές μας κλάσεις P και NP.[4] Από τη μια πλευρά δεν είναι δύσκολο να δούμε πως κάθε πρόβλημα αναζήτησης $\mathbb{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ της P μπορεί να διατυπωθεί σαν ένα PLS πρόβλημα. Υπενθυμίζουμε ότι αν ένα πρόβλημα αναζήτησης $\mathbb{R}$ ανοίκει στην P τότε υπάρχει πολυωνυμικός αλγόριθμος, ο οποίος δεδομένου ενός στιγμιοτύπου x του προβλήματος, είτε βγάζει σαν έξοδο ένα y τέτοιο ώστε $(x,y) \in \mathbb{R}$ ή αναφέρει (ορθά) ότι δεν υπάρχει τέτοιο y. Έτσι, μπορούμε να ορίσουμε για κάθε στιγμιότυπο x του $\mathbb{R}$ ένα σύνολο από λύσεις $\mathcal{J}(x)$, μία συνάρτηση κόστους $f(y,x)$ και μια συνάρτηση γειτονιάς $\mathcal{N}(y,x)$ μαζί με τους αντίστοιχους αλγόριθμους A, B, C, οι οποίοι να ικανοποιούν τις συνθήκες στον ορισμό της PLS, έτσι ώστε $(x,y) \in \mathbb{R}$ αν και μόνο αν το y είναι ένα τοπικό βέλτιστο για το x. Απλά, έστω $\mathcal{J}(x) = \{y : (x,y) \in \mathbb{R}\}$ και για κάθε $y \in \mathcal{J}(x)$ έστω $f(y,x) = 0$ και $\mathcal{N}(y,x) = \{y\}$. Ο αλγόριθμος A του ορισμού 8.2.1 είναι ο πολυωνυμικός αλγόριθμος που λύνει το πρόβλημα $\mathbb{R}$, ο αλγόριθμος B χρησιμοποιεί τον αλγόριθμο που αναγνωρίζει τα μέλη του $\mathbb{R}$ και ο αλγόριθμος C είναι τετριμμένος.

Από την άλλη κάθε πρόβλημα Π στην PLS ανήκει επίσης και στην NP αφού η σχέση $\{(x,y) :$ το y είναι τοπικό βέλτιστο για το $x\}$ είναι πολυωνυμικά αναγνωρίσιμη από τον αλγόριθμο C_Π του ορισμού 8.2.1. Έτσι έχουμε το ακόλουθο θεώρημα

Θεώρημα 8.2.1 *$P \subseteq PLS \subseteq NP$*

Στο άνω άκρο έχουμε ισχυρές ενδείξεις από τη θεωρία της πολυπλοκότητας ότι η PLS περιέχεται αυστηρά στην NP. Γνωρίζουμε ότι η NP περιέχει NP-δύσκολα προβλήματα. Για παράδειγμα το πρόβλημα αναζήτησης που αποτελείται από τα ζεύγη $\{x =$ένας γράφος, $y =$ένας κύκλος Hamilton του $x\}$ ανήκει στην NP. Η επίλυση αυτού του προβλήματος αναζήτησης είναι NP-δύσκολη αφού περιέχει το πρόβλημα του κύκλου του Hamilton. Το ακόλουθο γεγονός δείχνει ότι είναι εξαιρετικά απίθανο η PLS να περιέχει NP-δύσκολα προβλήματα.

Θεώρημα 8.2.2 *Αν ένα PLS-πρόβλημα* Π *είναι NP-δύσκολο τότε NP=co-NP.* [5]

[4] Πιο σωστά, αντί για τις κλάσεις P και NP, οι οποίες αναφέρονται σε προβλήματα απόφασης, θα έπρεπε να χρησιμοποιηθούν δύο αντίστοιχες κλάσεις για προβλήματα αναζήτησης στα οποία ανοίκουν και τα προβλήματα τοπικής αναζήτησης, όμως αυτό δεν επηρεάζει τις σχέσεις των κλάσεων που εξετάζουμε.

[5] Η κλάση πολυπλοκότητας co-NP είναι η κλάση των προβλημάτων απόφασης των οποίων το συ-

8.2.2 Προβλήματα που ανήκουν στην PLS

Ας ορίσουμε τώρα μερικά προβλήματα βελτιστοποίησης με κάποιες από τις συνηθισμένες γειτονιές τους και θα μελετήσουμε την πολυπλοκότητά τους στη συνέχεια.

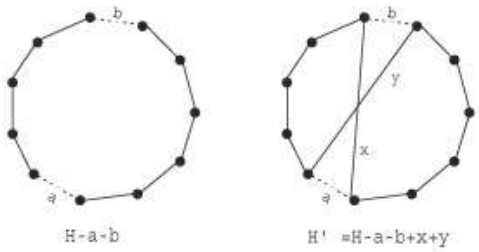
Travelling Salesman Problem (TSP)
Στο Πρόβλημα του Περιοδεύοντος Πωλητή μας δίνεται ένας πλήρης γράφος n κόμβων με θετικά ακέραια βάρη $w(e)$ πάνω στις ακμές του και θέλουμε να βρούμε τη διαδρομή ελάχιστου κόστους που περνάει ακριβώς μία φορά από κάθε κόμβο ή ισοδύναμα ψάχνουμε για έναν απλό κύκλο μήκους n. Σε αυτό το πρόβλημα οι κόμβοι καλούνται πόλεις και τα βάρη των ακμών αποστάσεις.

Με βάση το αρχικό TSP πρόβλημα βελτιστοποίησης μπορούμε να ορίσουμε ένα πλήθος από γειτονιές και να το μετατρέψουμε σε πρόβλημα τοπικής αναζήτησης. Η απλούστερη γειτονιά είναι η *2-Opt*, η οποία έχει ως εξής: *αντικατέστησε 2 ακμές (a,b), (c,d) με δύο άλλες ακμές (a,c) και (b,d) για να σχηματιστεί μία νέα διαδρομή* (Βλέπε Σχήμα 8.5).

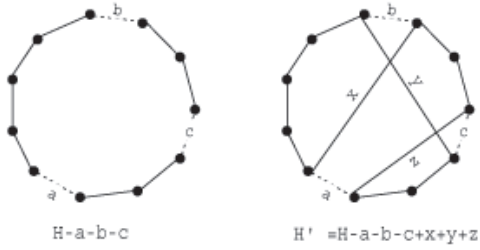
Σχήμα 8.5: Η γειτονιά 2-Opt

Αυτή η γειτονιά μπορεί να επεκταθεί στην 3-Opt, και ακόμα γενικότερα, στην *k-Opt*, όπου αντικαθιστούμε το πολύ k ακμές της τρέχουσας διαδρομής με k νέες ακμές. Πιο συγκεκριμένα ο αλγόριθμος των k-Opt γειτονιών έχει ως εξής. Ξεκινάει με έναν τυχαίο Χαμιλτονιακό κύκλο (μία αρχική περιοδεία) H. Διαγράφει k ακμές από τον H και έτσι σχηματίζονται k ασύνδετα μονοπάτια (μερικά από αυτά μπορεί να είναι και μεμονωμένοι κόμβοι). Επανασυνδέει αυτά τα k μονοπάτια με τέτοιο τρόπο ώστε να παραχθεί μία άλλη περιοδεία H', άρα τουλάχιστον δύο ακμές θα πρέπει να είναι διαφορετικές από αυτές

μπλήρωμα ανήκει στην NP. Το πρόβλημα ''Είναι ένας συγκεκριμένος γράφος μη Χαμιλτονιακός;'' ανήκει στην co-NP. Γενικά πιστεύουμε ότι NP≠co-NP.



Σχήμα 8.6: 2-εναλλαγή όπου οι πλευρές a, b αντικαθίστανται από τις πλευρές x, y.



Σχήμα 8.7: 3-εναλλαγή όπου οι πλευρές a, b, c αντικαθίστανται από τις x, y, z.

που αφαιρέθηκαν από το H. Οι τρόποι μπορεί να είναι πολλοί και θα πρέπει να τους ελέγξουμε όλους. Έτσι τα H και H' διαφέρουν σε έως και k ακμές, ενώ οι υπόλοιπες παραμένουν κοινές. Στα σχήματα 8.6 και 8.7 παρουσιάζονται οι 2- και 3- εναλλαγές αντίστοιχα. Στη συνέχεια υπολογίζουμε το ολικό βάρος $w(H')$ της περιοδείας H' και αν $w(H') < w(H)$, αντικαθιστούμε το H με το H' και επαναλαμβάνουμε την διαδικασία. Αλλιώς επιλέγουμε ένα άλλο σύνολο από k ακμές από το H για να εναλλάξουμε. Η τελική λύση, όπου δεν μπορεί να βελτιωθεί από τις εναλλαγές καμιάς από τις k ακμές της, καλείται *k-Opt λύση*. Δείτε ότι οι k-Opt λύσεις είναι και i-Opt για $i = 2, 3 \ldots, k-1$.

Maximum Cut (Max-Cut)

Ένα στιγμιότυπο αποτελείται από ένα μη κατευθυνόμενο γράφο $G = (V, E)$ με θετικά βάρη στις ακμές. Μία εφικτή λύση είναι μία διαμέριση του V σε δύο σύνολα A, B (όχι απαραίτητα ίσα). Το κόστος ορίζεται όπως και στη Διαμέριση Γράφου ότι είναι το βάρος της τομής (A, B), το οποίο θέλουμε και να μεγιστοποιήσουμε. Η εκδοχές μεγιστοποίησης και ελαχιστοποίησης εδώ δεν είναι ισοδύναμες. Στην πραγματικότητα η εκδοχή της ελαχιστοποίησης, Minimum Cut (Min-Cut), μπορεί να λυθεί σε πολυωνυμικό χρόνο,

ενώ το Max-Cut είναι NP-πλήρες. Η απλούστερη γειτονιά για αυτό το πρόβλημα είναι η *Flip*. Με την Flip δύο λύσεις (διαμερίσεις) είναι γείτονες εάν η μία μπορεί να προκύψει από την άλλη με την απλή μετακίνηση ενός κόμβου από τη μία πλευρά της διαμέρισης στην άλλη.

Maximum Satisfiability (Max-Sat)

Σε αυτό το πρόβλημα έχουμε έναν Boolean τύπο σε συζευκτική κανονική μορφή (CNF), του οποίου ο κάθε όρος (clause) έχει ένα θετικό ακέραιο βάρος. Μία λύση είναι μία ανάθεση από 0 ή 1 σε όλες τις μεταβλητές. Το κόστος, που θα πρέπει να μεγιστοποιηθεί, είναι το άθροισμα των βαρών των όρων που ικανοποιούνται από την ανάθεση. Για παράδειγμα η φόρμουλα

$$10(\bar{x} \vee y \vee \bar{z} \vee w) \wedge 15(x \vee \bar{w}) \wedge 100(\bar{y} \vee z)$$

με ανάθεση {x=1, y=1, z=0, w=1} έχει βάρος 25, αφού ικανοποιούνται μόνο οι δύο πρώτοι όροι. Ο περιορισμός των στιγμιοτύπων του Max-Sat σε αυτά που αποτελούνται από k το πολύ στοιχεία (literals) σε κάθε όρο λέγεται *Max-kSat*. Η απλούστερη γειτονιά είναι η *Flip*, στην οποία δύο λύσεις (αναθέσεις) είναι γειτονικές εάν η μία μπορεί να παραχθεί από την άλλη με την αντιστροφή της τιμής μίας μόνο μεταβλητής.

Not-All-Equal Maximum Satisfiability (NAE Max-Sat)

Ένα στιγμιότυπο του not-all-equal maximum satisfiability αποτελείται από όρους (clauses) της μορφής $\text{NAE}(\alpha_1, \ldots, \alpha_k)$, όπου κάθε α_i είναι ένα στοιχείο (δηλαδή μία μεταβλητή ή η άρνησή της) ή μία σταθερά 0 ή 1. Τέτοιοι όροι ικανοποιούνται εάν τα στοιχεία τους δεν έχουν όλα την ίδια τιμή. Σε κάθε τέτοιο όρο έχει ανατεθεί ένα θετικό ακέραιο βάρος. Μία λύση είναι και πάλι μία ανάθεση από 0 ή 1 σε όλες τις μεταβλητές και το κόστος που θα πρέπει να μεγιστοποιηθεί είναι το άθροισμα των βαρών των ικανοποιημένων όρων. Για παράδειγμα ο NAE τύπος

$$10 \cdot NAE(x, \bar{y}, z) + 20 \cdot NAE(y, w) + 120 \cdot NAE(1, \bar{w}, \bar{z})$$

με ανάθεση {x=1, y=0, z=1, w=1} έχει βάρος 140, αφού μόνο οι δύο τελευταίοι όροι ικανοποιούνται. Η γειτονιά *Flip* ορίζεται και εδώ όπως στο Max-Sat. Ο περιορισμός του

NAE Max-Sat σε στιγμιότυπα που περιέχουν το πολύ k στοιχεία σε κάθε όρο λέγεται *NAE Max-kSat*. Ο περιορισμός σε στιγμιότυπα χωρίς άρνηση, δηλαδή σε στιγμιότυπα που κανένας όρος δεν περιέχει αρνήσεις μεταβλητών, λέγεται *Pos NAE Max-kSat*. Σημειώστε ότι *το Max-Cut είναι ουσιαστικά ταυτόσημο με την ειδική περίπτωση του Pos NAE Max-2Sat όπου όλοι οι όροι έχουν μήκος δύο και δεν υπάρχουν σταθερές*.

Σε αναλογία με τη Διαμέριση Γράφου, μπορούμε επίσης να ορίσουμε μία μεταβλητού βάθους γειτονιά για τα Max-Cut, Max-Sat kai NAE Max-Sat, την οποία θα ονομάζουμε και πάλι Kernighan-Lin. Οι γείτονες μίας λύσης (διαμέρισης στο Max-Cut, ανάθεσης στα προβλήματα ικανοποιησιμότητας) είναι όλες οι λύσεις που μπορούμε να παράγουμε με μία ακολουθία από αντιστροφές. Σε κάθε βήμα της ακολουθίας επιλέγουμε την πιο επικερδή εναλλαγή ενός κόμβου ή αντιστροφή μίας μεταβλητής που δεν έχει εναλλαχθεί-αναστραφεί προηγουμένως. Και εδώ οι ισοπαλίες (ties) λύνονται με βάση κάποιον κανόνα tie-breaking. Φυσικά αυτή η γειτονιά είναι πιο ισχυρή από την Flip.

Stable configurations in neural networks (Hopfield model).
Μας δίνεται ένας μη κατευθυνόμενος γράφος $G = (V, E)$ με ένα θετικό ή αρνητικό βάρος w_e σε κάθε ακμή e και ένα κατώφλι (threshold) t_v για κάθε κόμβο v (μπορούμε να θεωρήσουμε ότι οι ακμές που λείπουν έχουν βάρος 0). Μια configuration αναθέτει σε κάθε κόμβο v μία κατάσταση s_v, η οποία είναι ή 1 ('on') ή -1 ('off'). Ένας κόμβος είναι ''χαρούμενος'' αν $s_v = 1$ και $\sum_u w_{(u,v)} s_u + t_v \geq 0$ ή $s_v = -1$ και $\sum_u w_{(u,v)} s_u + t_v \leq 0$. Μία configuration είναι stable αν όλοι οι κόμβοι είναι χαρούμενοι. Το πρόβλημα είναι να βρεθεί μία stable configuration για ένα δεδομένο δίκτυο.

Δεν είναι προφανές εκ των προτέρων ότι μία τέτοια configuration υπάρχει. Στην πραγματικότητα στην περίπτωση των κατευθυνόμενων γράφων μπορεί και να μην υπάρχει. Ο Hopfield έδειξε ότι στην περίπτωση των μη κατευθυνόμενων γράφων υπάρχει πάντα μία τέτοια configuration. Για να το αποδείξει εισήγαγε μία συνάρτηση κόστους $\sum_{(u,v)\in E} w_{(u,v)} s_u s_v + \sum_{v\in V} t_v s_v$ και υποστήριξε ότι αν ένας κόμβος είναι ''στενοχωρημένος'' τότε αλλάζοντας την κατάστασή του θα αυξηθεί το κόστος. Αυτό σημαίνει ότι οι stable configurations συμπίπτουν με τις τοπικά μέγιστες λύσεις για αυτήν τη συνάρτηση σύμφωνα με τη γειτονιά που αλλάζει την κατάσταση ενός μόνο κόμβου (Flip).

Το πρόβλημα των stable configurations για τα νευρωνικά δίκτυα όπου όλα τα βάρη των ακμών είναι αρνητικά και όλα τα thresholds είναι 0 είναι ισοδύναμο με το Max-Cut/Flip. Αν τα thresholds είναι τυχαία, τότε είναι ισοδύναμο με το s-t Max-Cut/Flip, στο οποίο υπάρχουν δύο διαφορετικοί κόμβοι s και t που πρέπει να είναι σε διαφορετικές πλευρές της διαμέρισης. Αν όλα τα βάρη είναι θετικά το πρόβλημα των stable configurations είναι ισοδύναμο με το Min-Cut/Flip ή το s-t Min-Cut/Flip, ανάλογα με το αν τα thresholds είναι 0 ή όχι. Στην περίπτωση των θετικών βαρών το πρόβλημα μπορεί να επιλυθεί σε πολυωνυμικό χρόνο, επειδή κάποιος μπορεί να βρει ακόμα και τις ολικά βέλτιστες λύσεις για τα Min-Cut και s-t Min-Cut σε πολυωνυμικό χρόνο.

8.2.3 Αναγωγές

Είμαστε έτοιμοι, λοιπόν, να συσχετίσουμε τα προβλήματα τοπικής αναζήτησης μεταξύ τους, με κατάλληλες αναγωγές, και να προσδιορίσουμε τα δυσκολότερα προβλήματα που ανήκουν στην κλάση PLS.

Ορισμός 8.2.2 (PLS-αναγωγή) *Έστω Π_1 και Π_2 δύο προβλήματα τοπικής αναζήτησης. Μία PLS-αναγωγή από το Π_1 στο Π_2 αποτελείται από δύο πολυωνυμικού χρόνου υπολογίσιμες συναρτήσεις h και g τέτοιες ώστε:*

1. *Η h απεικονίζει στιγμιότυπα x του Π_1 σε στιγμιότυπα $h(x)$ του Π_2*
2. *Η g απεικονίζει ζεύγη της μορφής (λύση του $h(x)$,x) σε λύσεις του x. Δηλαδή αντιστοιχεί τις λύσεις του $h(x)$ σε λύσεις του x και*
3. *Για όλα τα στιγμιότυπα του Π_1, αν s είναι ένα τοπικό βέλτιστο για το στιγμιότυπο $h(x)$ του Π_2 τότε το $g(s,x)$ είναι ένα τοπικό βέλτιστο του x.*

Αν υπάρχει μία τέτοια αναγωγή λέμε ότι το Π_1 PLS-ανάγεται στο Π_2.

Εύκολα διαπιστώνουμε ότι για τις PLS-αναγωγές ισχύει η μεταβατική ιδιότητα και ότι μας επιτρέπουν να συσχετίσουμε τη δυσκολία ενός προβλήματος με αυτή ενός άλλου.

Πρόταση 8.2.1 *Αν Π_1, Π_2 και Π_3 είναι προβλήματα στην PLS τέτοια ώστε το Π_1 να PLS-ανάγεται στο Π_2 και το Π_2 να PLS-ανάγεται στο Π_3, τότε το Π_1 PLS-ανάγεται στο Π_3.*

Πρόταση 8.2.2 *Αν* Π_1 *και* Π_2 *είναι προβλήματα στην PLS τέτοια ώστε το* Π_1 *να PLS-ανάγεται στο* Π_2 *και αν υπάρχει πολυωνυμικός αλγόριθμος για την εύρεση τοπικού βέλτιστου στο* Π_2*, τότε υπάρχει πολυωνυμικός αλγόριθμος για την εύρεση τοπικού βέλτιστου στο* Π_1*.*

Ορισμός 8.2.3 *Ένα πρόβλημα* Π *που ανήκει στην PLS είναι PLS-πλήρες αν κάθε πρόβλημα που ανήκει στην PLS μπορεί να PLS-αναχθεί σε αυτό.*

Το πρώτο πρόβλημα που δείχθηκε PLS-πλήρες λέγεται Circuit/Flip. Ένα στιγμιότυπο αυτού του προβλήματος είναι ένα συνδυαστικό Boolean κύκλωμα x (για την ακρίβεια μία κωδικοποίησή του) που αποτελείται από AND, OR και NOT πύλες. Έστω ότι το x έχει m εισόδους και n εξόδους. Το σύνολο των λύσεων $\mathcal{J}(x)$ αποτελείται από όλες τις δυαδικές συμβολοσειρές μήκους m, δηλαδή όλες τις πιθανές εισόδους. Η γειτονιά $\mathcal{N}(s, x)$ μιας λύσης s αποτελείται από όλες τις συμβολοσειρές μήκους m που έχουν απόσταση Hamming ίση με 1 από το s (δηλαδή η Flip). Το κόστος μιας λύσης s είναι το διάνυσμα εξόδου του κυκλώματος για είσοδο s, το οποίο εκφράζει έναν αριθμό γραμμένο σε δυαδική μορφή. Το πρόβλημα μπορεί να οριστεί είτε ως πρόβλημα μεγιστοποίησης είτε ως πρόβλημα ελαχιστοποίησης γιατί οι δύο εκδοχές είναι ισοδύναμες. Το πρόβλημα τοπικής αναζήτησης ζητάει μια είσοδο τέτοια ώστε η έξοδος να μην βελτιώνεται μέσω της Flip. *Η απόδειξη της PLS-πληρότητας αυτού του προβλήματος δείχνει ότι κάθε πρόβλημα τοπικής αναζήτησης, όποια γειτονιά και αν έχει, μπορεί να PLS-αναχθεί στο Circuit, το οποίο αφού έχει γειτονιά την απλούστατη Flip, μεταφέρει όλη την πολυπλοκότητα του αρχικού προβλήματος στη συνάρτηση κόστους του, δηλαδή μέσα στο κύκλωμα.*

Το ακόλουθο θεώρημα συνοψίζει της κυριότερες αναγωγές PLS-πληρότητας με την πορεία που ακολουθήθηκε να φαίνεται στο ς χήμα 8.8. Όλα τα προβλήματα που είδαμε στην προηγούμενη ενότητα αποδεικνύονται PLS-πλήρη με όλες τις γειτονιές που αναφέραμε. Όσες γειτονιές δεν έχουν οριστεί και αναφέρονται στο θεώρημα είναι κυρίως παραλλαγές των όσων είπαμε.

Θεώρημα 8.2.3 *Τα ακόλουθα προβλήματα είναι PLS-πλήρη:*

1. *Διαμέριση Γράφου (GP) κάτω από τις γειτονιές (a)Kernighan-Lin, (b) Swap, (c) Fiduccia-Mattheyses, (d) FM-Swap.*

2. *Πρόβλημα Πλανόδιου Πωλητή (TSP) κάτω από την γειτονιά k-Opt για κάποια σταθερά k, καθώς και κάτω από την γειτονιά LK′ (Lin-Kernighan′).*

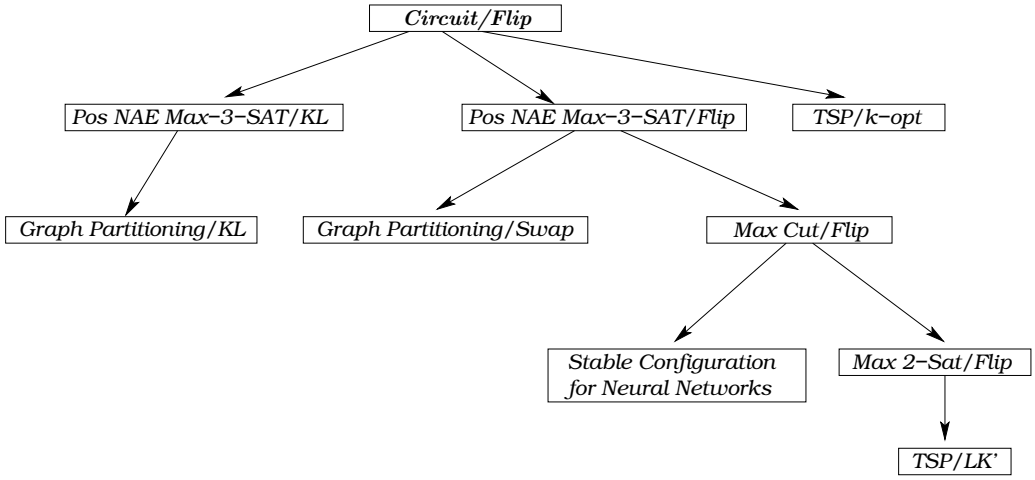
3. *Max-Cut/Flip.*

4. *Max-2Sat/Flip.*

5. *Pos NAE Max-3Sat/Flip.*

6. *Stable configurations in neural networks/flip.*

Οι γειτονιές k-Opt με τις οποίες το TSP αποδεικνύεται PLS-πλήρες έχουν $k \geq 6$. Αυτό σημαίνει ότι η πολυπλοκότητα προβλημάτων όπως των TSP/2-Opt, TSP/3-Opt αλλά και του TSP/KL παραμένει ανοιχτή. Επίσης, αφού η Flip είναι το πρώτο βήμα της KL, τα προβλήματα που έχουν δειχθεί PLS-πλήρη με την Flip θα είναι και με την KL.

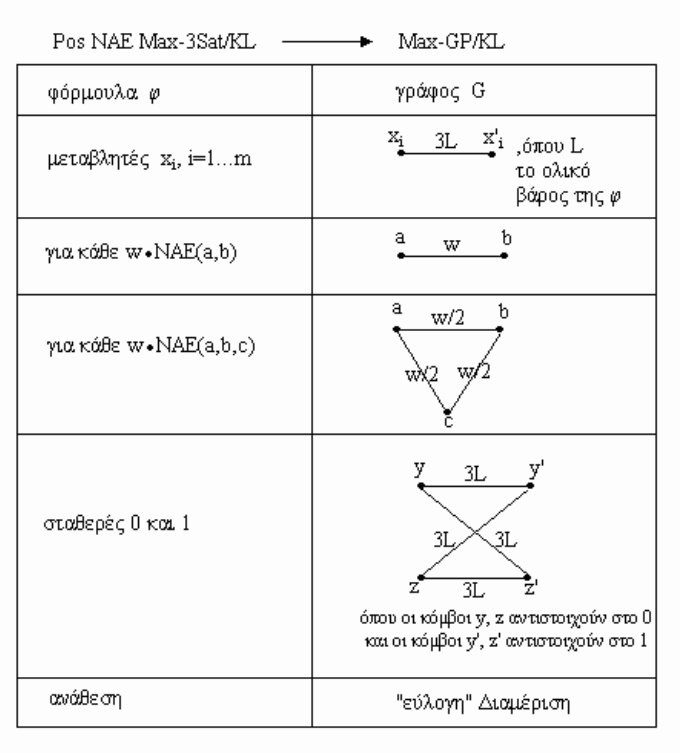


Σχήμα 8.8: Δέντρο αναγωγών. Το πρώτο PLS-πλήρες πρόβλημα είναι το Circuit/Flip, ένα συνδυαστικό Boolean κύκλωμα από AND, OR, NOT πύλες.

Στη συνέχεια θα δούμε δύο αναγωγές. Η πρώτη ανάγει το Pos-NAE Max-3Sat/KL στο GP/KL και η δεύτερη το Max-Cut/Flip στο Max-2Sat/Flip.

Λήμμα 8.2.1 (Pos NAE Max-3Sat/KL → GP/KL) *Το Pos NAE Max-3Sat με γειτονιά την Kernighan-Lin PLS-ανάγεται στη διαμέριση γράφου (graph partitioning) με γειτονιά την Kernighan-Lin.*

Απόδειξη Η αναγωγή θα γίνει στην εκδοχή μεγιστοποίησης της διαμέρισης γράφου, η οποία, όπως έχουμε πει, είναι ισοδύναμη με την εκδοχή ελαχιστοποίησης. Έστω ϕ ένας Pos NAE Max-3Sat τύπος με μεταβλητές $x_1, \ldots, x_m$ και έστω L το συνολικό βάρος όλων των όρων. Κατασκευάζουμε ένα γράφο G, ο οποίος περιέχει δύο κόμβους x_i, x_i' για κάθε μεταβλητή x_i, άλλους δύο κόμβους y, z για τη σταθερά 0 και δύο ακόμα κόμβους y', z' για τη σταθερά 1. Υπάρχει μία ακμή (x_i, x_i'), για κάθε $i = 1, \ldots, m$, καθώς επίσης και ακμές που ενώνουν και τους δύο 0 κόμβους με τους δύο 1 κόμβους, όλες βάρους $3L$. Για κάθε 2-όρο NAE(a, b) βάρους w συμπεριλαμβάνουμε μία ακμή (a, b) βάρους w, όπου για τις σταθερές 0 και 1 χρησιμοποιούμε τους κόμβους y και y'. Για έναν 3-όρο NAE(a, b, c) βάρους w, συμπεριλαμβάνουμε τις ακμές $(a, b), (b, c), (a, c)$ όλες βάρους $w/2$. Αν ένα ζευγάρι κόμβων συναντάται σε πολλούς όρους τότε προσθέτουμε τα βάρη που προκύπτουν από όλους αυτούς τους κόμβους. Συνοπτικά οι αντιστοιχίες τις κατασκευής φαίνονται στον Πίνακα 8.1.



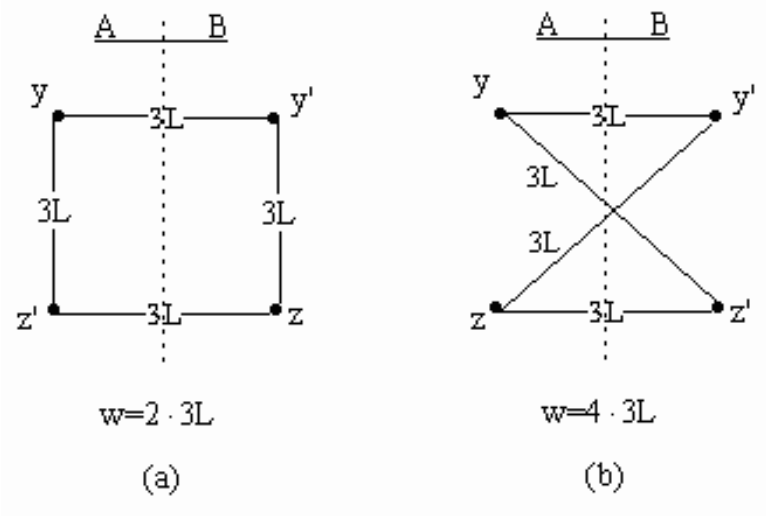
Πίνακας 8.1: Αναγωγή από το Pos NAE Max-3Sat/KL στο GP/KL

Ισχυριζόμαστε πως μία διαμέριση (A, B) των κόμβων σε δύο ίσα μέρη είναι τοπικά βέλτιστη αν και μόνο αν

(i) οι δύο 0 κόμβοι βρίσκονται στην ίδια πλευρά, οι δύο 1 κόμβοι βρίσκονται στην αντίθετη πλευρά και για κάθε i οι κόμβοι x_i, x_i' βρίσκονται σε αντίθετες πλευρές,

(ii) η ανάθεση στην οποία μία μεταβλητή x_i είναι 1 αν και μόνο αν ο κόμβος x_i βρίσκεται στην ίδια πλευρά με τους 1 κόμβους, είναι τοπικά βέλτιστη για τη ϕ.

Από τον ορισμό της PLS-αναγωγής βλέπουμε ότι μας αρκεί να αποδείξουμε μόνο το ''μόνο αν'' μέρος. Έστω ότι το (i) δεν ισχύει. Υποθέστε τότε ότι το A περιέχει ένα ζευγάρι u, u' αντίστοιχων κόμβων. Αφού το B, έχει το ίδιο μέγεθος, θα περιέχει και αυτό ένα ζευγάρι v, v'. Εναλλάσσοντας το u με το v κερδίζουμε τουλάχιστον L. Ας υποθέσουμε τώρα ότι οι αντίστοιχοι κόμβοι u, u' είναι σε αντίθετες πλευρές αλλά οι δύο 0 κόμβοι έχουν χωριστεί. Έστω το $y \in A$ και το $z \in B$, οπότε $y' \in B$ και $z' \in A$. Εναλλάσσοντας τα z και z' βελτιώνεται το βάρος, όπως παρατηρούμε στο Σχήμα 8.9.



Σχήμα 8.9: Τα y, z πάνε αντίθετα από τα y' και z'

Οπότε θεωρούμε ότι το (i) ισχύει. Έστω ότι αποκαλούμε μία τέτοια διαμέριση ''εύλογη''. Υπάρχει μία 1-1 αντιστοιχία μεταξύ εύλογων διαμερίσεων και αληθών αναθέσεων. Σημειώστε ότι αν ένας όρος $\mathrm{NAE}(a, b, c)$ ικανοποιείται τότε ακριβώς δύο

από τις τρεις αντίστοιχες ακμές βρίσκονται στη διατομή, αλλά αν δεν ικανοποιείται τότε δεν βρίσκεται καμία ακμή στη διατομή. Εύκολα συνεπάγεται, λοιπόν, ότι το βάρος μίας διαμέρισης ισούται με το βάρος της αντίστοιχης ανάθεσης συν $3L(m+4)$. Όταν ασχολούμαστε με άπλειστες εναλλαγές από μία εύλογη διαμέριση, τότε αρκεί να περιορίσουμε την προσοχή μας σε εναλλαγές που περιλαμβάνουν ένα αντίστοιχο σε μεταβλητή ζευγάρι κόμβων x_i, x'_i, επειδή οι άλλες εναλλαγές έχουν σαν αποτέλεσμα την απώλεια ενός τεράστιου κόστους. Εναλλάσσοντας τους κόμβους x_i και x'_i αντιστοιχεί με την αντιστροφή της μεταβλητής x_i. Έτσι η γειτονιά Kernighan-Lin μιας εύλογης διαμέρισης αποτελείται από μία ακολουθία εύλογων διαμερίσεων που αντιστοιχούν σε μία ακολουθία αντιστροφών. *Ο ισχυρισμός (ii), λοιπόν, συνάγεται από την 1-1 αντιστοιχία των εύλογων διαμερίσεων με τις αληθείς αναθέσεις και το γεγονός ότι η αντιστοιχία διατηρεί το βάρος μιας λύσης προσθέτοντας μόνο έναν παράγοντα (δηλαδή οι αναλογίες μεταξύ των βαρών των λύσεων διατηρούνται).* Σε αυτή την αναγωγή ο "χώρος" όπου οι λύσεις τοποθετούνται με βάση το κόστος και τους γείτονές τους παραμένει ακριβώς ο ίδιος, μόνο που ο άξονας του κόστους έχει μετατοπιστεί κατά $3L(m+4)$. □

Λήμμα 8.2.2 (Max-Cut/Flip → Max-2Sat/Flip) *Το Max-Cut με γειτονιά τη Flip PLS-ανάγεται στο Max-2Sat με γειτονιά την Flip.*

Απόδειξη Έστω G ο βεβαρημένος γράφος του Max-Cut/Flip. Κατασκευάζουμε έναν 2Sat τύπο με βάρη ϕ, ο οποίος έχει μία μεταβλητή για κάθε κόμβο του G. Αν ο G έχει μία ακμή $[x, y]$ με βάρος w, τότε συμπεριλαμβάνουμε στο ϕ τους όρους $(x \vee y)$ και $(\bar{x} \vee \bar{y})$ ο καθένας με βάρος w. Προσέξτε πως αν στην αληθή ανάθεση έχουμε $x = y$ τότε μόνο ένας από αυτούς τους δύο όρους θα ικανοποιείται. Αντίθετα αν $x \neq y$ τότε και οι δύο όροι ικανοποιούνται. Μία αληθής ανάθεση του ϕ προκαλεί μία διαμέριση του G, στην οποία περιλαμβάνονται από τη μία πλευρά οι κόμβοι που αντιστοιχούν στις αληθείς μεταβλητές και από την άλλη τους κόμβους που αντιστοιχούν στις ψευδείς μεταβλητές. Το βάρος των ικανοποιημένων όρων είναι ίσο με το βάρος της διατομής συν το συνολικό βάρος όλων των ακμών. Έτσι, αντιστρέφοντας μία μεταβλητή στο 2Sat στιγμιότυπο ϕ, αλλάζει το βάρος της ανάθεσης κατά το ίδιο ποσό που αλλάζει το βάρος της διατομής εάν μετακινήσουμε τον αντίστοιχο κόμβο στην άλλη πλευρά. Γι' αυτό το λόγο μία τοπικά βέλτιστη αληθής ανάθεση προκαλεί μία τοπικά βέλτιστη διαμέριση. □

8.2.4 Πολυπλοκότητα του κλασικού ευριστικού αλγορίθμου της τοπικής αναζήτησης

Όπως είδαμε στην προηγούμενη ενότητα, τη λύση στον καθορισμό της πολυπλοκότητας των προβλημάτων τοπικής αναζήτησης μας την έδωσε η PLS, αφού ένα πρόβλημα μπορεί να χαρακτηριστεί ''δύσκολο'' αν αποδειχθεί ότι είναι PLS-πλήρες. Τώρα θα μελετήσουμε την πολυπλοκότητα του κλασικού αλγόριθμου τοπικής αναζήτησης. Θα δούμε ότι *αυτός ο επαναληπτικός αλγόριθμος χρειάζεται εκθετικό χρόνο στη χείριστη περίπτωση για τα PLS-πλήρη προβλήματα που αναφέραμε*. Πρώτα θα δώσουμε μερικούς ορισμούς.

Ορισμός 8.2.4 *Έστω* Π *ένα πρόβλημα τοπικής αναζήτησης και έστω* x *ένα στιγμιότυπο του* Π. *Ο* **γράφος γειτονιάς** $NG_\Pi(x)$ *του στιγμιοτύπου* x *είναι ένας κατευθυνόμενος γράφος με ένα κόμβο για κάθε εφικτή λύση του* x *και με ένα τόξο* $s \to t$ *όταν το* $t \in \mathcal{N}_\Pi(s, x)$. *Ο* **γράφος μετάβασης** $TG_\Pi(x)$ *είναι ο υπογράφος που περιέχει εκείνα τα τόξα για τα οποία το κόστος* $f_\Pi(t, x)$ *είναι αυστηρά καλύτερο από το* $f_\Pi(s, x)$ *(μεγαλύτερο αν το* Π *είναι πρόβλημα μεγιστοποίησης ή μικρότερο αν το* Π *είναι πρόβλημα ελαχιστοποίησης). Το* **ύψος ενός κόμβου** v *είναι το μήκος του συντομότερου μονοπατιού στον* $TG_\Pi(x)$ *από το* v *σε ένα πηγάδι (sink), δηλαδή σε ένα κόμβο χωρίς εξερχόμενα τόξα. Το* **ύψος του γράφου μετάβασης** $TG_\Pi(x)$ *είναι το μεγαλύτερο από τα ύψη των κόμβων.*

Θα ασχοληθούμε κυρίως με τον γράφο μετάβασης. Παρατηρήστε ότι ο $TG_\Pi(x)$ είναι ένας ακυκλικός γράφος. Επίσης δείτε ότι το κόστος προκαλεί μία τοπολογική διάταξη των κόμβων: τα τόξα κατευθύνονται από χειρότερους σε καλύτερους κόμβους. Έτσι, τα τοπικά βέλτιστα είναι τα πηγάδια του γράφου. Ο $TG_\Pi(x)$ αναπαριστά τις πιθανές νόμιμες κινήσεις για έναν αλγόριθμο τοπικής αναζήτησης στο στιγμιότυπο x. Αρχίζοντας από κάποιο κόμβο (λύση) v, ο κλασικός αλγόριθμος τοπικής αναζήτησης κινείται από κόμβο σε κόμβο προς ένα πηγάδι, διαγράφοντας ένα μονοπάτι. Το μήκος του μονοπατιού είναι το πλήθος των επαναλήψεων του αλγόριθμου, το οποίο καθορίζει και το χρόνο εκτέλεσής του. Το ακριβές μονοπάτι που ακολουθείται (οπότε και η πολυπλοκότητα) καθορίζεται από τον κανόνα μετάβασης (pivoting rule) που έχουμε επιλέξει. Σε κάθε κόμβο που δεν είναι πηγάδι ο κανόνας μετάβασης επιλέγει ποιο από τα εξερχόμενα τόξα θα ακολουθηθεί. Το ύψος ενός κόμβου v είναι το χαμηλότερο

φράγμα του πλήθους των επαναλήψεων που απαιτούνται από τον κλασικό αλγόριθμο ακόμα και αν χρησιμοποιεί σε κάθε βήμα τον καλύτερο δυνατό κανόνα μετάβασης.

Η ανάλυση της πολυπλοκότητας ενός συγκεκριμένου αλγορίθμου τοπικής αναζήτησης δεν είναι γενικά μια εύκολη διαδικασία. Αυτό φαίνεται καθαρά στο παράδειγμα του γραμμικού προγραμματισμού με τον αλγόριθμο Simplex ως τον αντίστοιχο αλγόριθμο τοπικής αναζήτησης. Η πολυπλοκότητα εξαρτάται από τον κανόνα μετάβασης. Πολλές μελέτες έχουν εξετάσει την πολυπλοκότητα της Simplex και έχουν κατασκευάσει αντιπαραδείγματα για πολλούς κανόνες μετάβασης, όπως ο πρώτα-ο-καλύτερος (steepest descent) και άλλοι, δείχνοντας ότι οδηγούν σε ένα εκθετικό πλήθος επαναλήψεων. Ωστόσο ορισμένοι κανόνες μετάβασης δεν έχουν κατανοηθεί ακόμα αρκετά καλά. Για παράδειγμα δεν γνωρίζουμε τον χρόνο εκτέλεσης του τυχαίου κανόνα μετάβασης ο οποίος σε κάθε υποβέλτιστη λύση απλά επιλέγει έναν καλύτερο γείτονα τυχαία, και εάν πράγματι μπορεί να εγγυηθεί ένα πολυωνυμικό πλήθος επαναλήψεων. Το μόνο, έως τώρα γνωστό, κάτω φράγμα είναι το τετραγωνικό. Γενικά παραμένει ακόμα ανοιχτό το πρόβλημα του εάν υπάρχει ένας κανόνας που κάνει την Simplex έναν πολυωνυμικού χρόνου αλγόριθμο.

Τώρα, όπως προαναφέραμε, αν ο γράφος μετάβασης έχει εκθετικό ύψος, ο κλασικός αλγόριθμος τοπικής αναζήτησης θα χρειαστεί εκθετικό χρόνο στην χείριστη περίπτωση, ανεξαρτήτως του πώς διαλέγει καλύτερους γείτονες. Αυτό προκύπτει ότι συμβαίνει με όλα τα προβλήματα που έχουν δειχθεί ότι είναι PLS-πλήρη. Η έννοια της PLS-αναγωγής που ορίσαμε δεν είναι αρκετή για να αποδείξει κάτι τέτοιο, αλλά μπορεί να ενδυναμωθεί με έναν κατάλληλο τρόπο.

Ορισμός 8.2.5 (tight PLS-αναγωγή) *Έστω P και Q δύο προβλήματα τοπικής αναζήτησης και έστω (h, g) μία PLS-αναγωγή από το P στο Q. Λέμε ότι η αναγωγή είναι* **αυστηρή (tight)** *αν για κάθε στιγμιότυπο x του P μπορούμε να επιλέξουμε ένα υποσύνολο R από τις εφικτές λύσεις της απεικόνισης $y = h(x)$ του x στο Q, έτσι ώστε να ικανοποιούνται οι ακόλουθες ιδιότητες:*

1. *Το R περιέχει όλα τα τοπικά βέλτιστα του y*

2. *Για κάθε λύση p του x μπορούμε να κατασκευάσουμε σε πολυωνυμικό χρόνο μία λύση $q \in R$ του y έτσι ώστε $g(q, x) = p$*

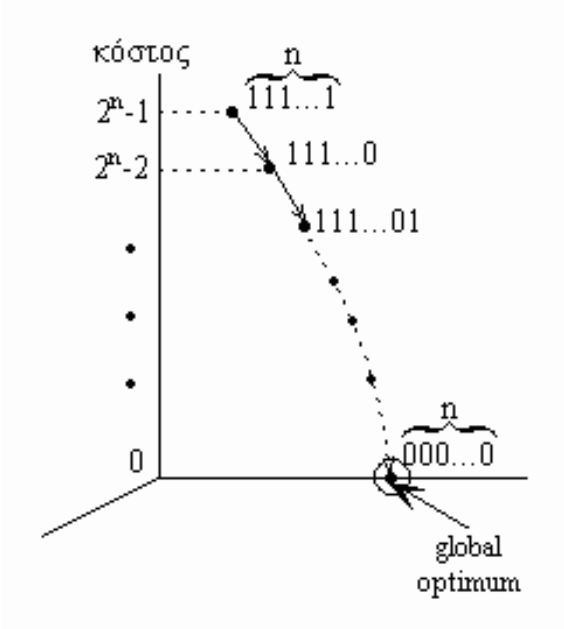
3. *Υποθέστε ότι ο γράφος μετάβασης του y, $TG_Q(y)$ περιέχει ένα κατευθυνόμενο μονοπάτι από το $q \in R$ στο $q' \in R$, τέτοιο ώστε όλοι οι εσωτερικοί κόμβοι του μονοπατιού να είναι εκτός του R και έστω $p = g(q, x)$ και $p' = g(q', x)$ οι αντίστοιχες λύσεις του x. Τότε ή θα ισχύει $p = p'$ ή ο $TG_P(x)$ θα περιέχει ένα τόξο από το p στο p'.*

Όλες οι αναγωγές που έχουμε αναφέρει είναι αυστηρές. Τις αυστηρές αναγωγές μπορούμε να τις συνθέσουμε. Επίσης, θα αναφερόμαστε συνήθως στις λύσεις που είναι στο R ως εύλογες (reasonable) λύσεις του στιγμιοτύπου y. *Οι αυστηρές αναγωγές μας επιτρέπουν να μεταφέρουμε κάτω φράγματα στο χρόνο εκτέλεσης του αλγόριθμου τοπικής αναζήτησης από ένα πρόβλημα σε κάποιο άλλο.*

Λήμμα 8.2.3 *Υποθέστε ότι τα P και Q είναι προβλήματα στην PLS και ότι οι h, g ορίζουν μία αυστηρή PLS-αναγωγή από το πρόβλημα P στο πρόβλημα Q. Αν x είναι στιγμιότυπο του P και $y = h(x)$ η απεικόνισή του στο Q, τότε το ύψος του $TG_Q(y)$ είναι τουλάχιστον τόσο όσο το ύψος του $TG_P(x)$. Έτσι αν ο κλασικός αλγόριθμος τοπικής αναζήτησης του P θέλει εκθετικό χρόνο στη χείριστη περίπτωση, τότε το ίδιο θα θέλει και ο κλασικός αλγόριθμος τοπικής αναζήτησης του Q.*

Απόδειξη Έστω x ένα στιγμιότυπο του P και $TG_P(x)$ ο γράφος μετάβασής του και έστω p μια λύση (κόμβος) της οποίας το ύψος είναι ίσο με το ύψος του $TG_P(x)$. Έστω $y = h(x)$ και $q \in R$ μία λύση του y τέτοια ώστε $g(q, x) = p$. Ισχυριζόμαστε ότι το ύψος της q στο $TG_Q(y)$ είναι τουλάχιστον τόσο όσο το ύψος του p στο $TG_P(x)$. Για να φανεί αυτό, θεωρείστε το συντομότερο μονοπάτι από το q σε ένα πηγάδι του $TG_Q(y)$ και έστω οι κόμβοι του R που εμφανίζονται σε αυτό το μονοπάτι είναι οι $q, q1, \ldots, qk$. Έστω $p, p1, \ldots, pk$ οι απεικονίσεις υπό την g αυτών των λύσεων, δηλαδή $p_i = g(q_i, x)$. Από τον ορισμό της αυστηρής αναγωγής ξέρουμε ότι το q_k είναι τοπικό βέλτιστο του y και έτσι και το p_k είναι τοπικό βέλτιστο του x. Επίσης για κάθε i είτε $p_i = p_{i+1}$ ή υπάρχει ένα τόξο στο $TG_P(x)$ από το p_i στο p_{i+1}. Επομένως υπάρχει ένα μονοπάτι μήκους το πολύ k από μια κορυφή p σε ένα πηγάδι του $TG_P(x)$. □

Για να αποδείξουμε ότι στη χείριστη περίπτωση ο χρόνος εκτέλεσης του κλασικού αλγόριθμου τοπικής αναζήτησης για τα αυστηρά PLS-πλήρη προβλήματα είναι εκθετικός, αρκεί να δείξουμε ότι ένα πρόβλημα της PLS έχει αυτή την ιδιότητα.



Σχήμα 8.10: Γράφος Μετάβασης εκθετικού PLS προβλήματος

Λήμμα 8.2.4 *Υπάρχει ένα πρόβλημα τοπικής αναζήτησης στην PLS του οποίου ο κλασικός αλγόριθμος απαιτεί εκθετικό χρόνο.*

Απόδειξη Θεωρείστε το ακόλουθο τεχνητό πρόβλημα ελαχιστοποίησης. Για κάθε στιγμιότυπο x μεγέθους n, το σύνολο των λύσεων αποτελείται από όλους τους, μήκους n-bit, ακέραιους $0, ..., 2^n - 1$. Για κάθε λύση i, το κόστος της είναι i και αν $i > 0$, τότε έχει ένα γείτονα, τον $i - 1$. Έτσι, υπάρχει ένα μοναδικό τοπικό και ολικό βέλτιστο, το 0, και ο γράφος μετάβασης είναι ένα μονοπάτι από το $2^n - 1$ προς το 0 (Σχήμα 8.10). Ο αλγόριθμος τοπικής αναζήτησης αρχίζοντας από το $2^n - 1$ θα ακολουθήσει αυτό το μονοπάτι και θα τερματίσει μετά από εκθετικό πλήθος επαναλήψεων. □

Θεώρημα 8.2.4 (Πολυπλοκότητα Κλασικού Αλγόριθμου) *Στα παρακάτω προβλήματα ο κλασικός αλγόριθμος τοπικής αναζήτησης απαιτεί εκθετικό χρόνο στη χείριστη περίπτωση, ανεξαρτήτως ποιοι κανόνες μετάβασης και tie-breaking χρησιμοποιούνται:*

1. *To Graph Partitioning με τις γειτονιές (a) Kernighan-Lin, (b) Swap, (c) Fiduccia-Mattheyses, (d) FM-Swap.*

2. *Το Travelling Salesman Problem με τις γειτονιές k-Opt για κάποια σταθερά k και LK΄ (Lin-Kernighan΄).*

3. *Τα Max-Cut, Max-2Sat και Pos NAE Max-3Sat με τη γειτονιά Flip.*

4. *Το πρόβλημα των stable configurations για τα νευρωνικά δίκτυα*

Σημειώστε ότι τα εκθετικά φράγματα ισχύουν για κάθε κανόνα μετάβασης, συμπεριλαμβανομένων και των τυχαιοποιημένων κανόνων.

8.3 Προσέγγιση Ολικού Βέλτιστου

Όπως έχουμε πει, τη μέθοδο της Τοπικής Αναζήτησης τη χρησιμοποιούμε, συνήθως, για να αντιμετωπίσουμε δύσκολα προβλήματα βελτιστοποίησης. Αναθέτοντας μια δομή γειτονιάς στις λύσεις ενός προβλήματος και αναζητώντας ένα τοπικό, μόνο, βέλτιστο κατορθώνουμε να μειώσουμε την πολυπλοκότητα του προβλήματος. Ο μοναδικός περιορισμός που βάζουμε στις γειτονιές είναι ότι αυτές θα πρέπει να μπορούν να αναζητηθούν αποδοτικά, δηλαδή σε πολυωνυμικό χρόνο. Το ιδανικό, για κάποιο πρόβλημα βελτιστοποίησης, θα ήταν να βρίσκαμε μία ακριβή γειτονιά , μια γειτονιά, δηλαδή, στην οποία τα τοπικά και τα ολικά βέλτιστα συμπίπτουν. Δυστυχώς όμως, κάτι τέτοιο δεν συμβαίνει και όπως διαισθητικά αντιλαμβάνεται κάποιος, αυτή η μείωση της πολυπλοκότητας του προβλήματος έχει αντίκτυπο στην ποιότητα των βέλτιστων που μπορούμε να εγγυηθούμε ότι θα καταλήξουμε. Πιο τυπικά μπορούμε να πούμε τα εξής:

Θυμηθείτε καταρχήν ότι ένα πρόβλημα λέγεται *ισχυρά (strongly) NP-δύσκολο* εάν παραμένει NP-δύσκολο ακόμα και όταν τα βάρη (κόστη) των στιγμιοτύπων του είναι πολυωνυμικά φραγμένα. Επίσης σημειώστε ότι την ποιότητα μιας λύσης s τη μετράμε με το λόγο του κόστους της c_s προς το κόστος του ολικού βέλτιστου c_{s^*}

$$\lambda = \frac{c_s}{c_{s^*}}$$

σε προβλήματα ελαχιστοποίησης και αντίστροφα σε προβλήματα μεγιστοποίησης. Λέμε ότι μία δομή γειτονιάς εγγυάται λόγο ε ή ότι είναι *ε-προσσεγγιστική (ε-approximation)*, εάν για κάθε στιγμιότυπο x και τοπικά βέλτιστη λύση $\hat{s}$, το κόστος του $\hat{s}$ απέχει το

πολύ κατά ένα παράγοντα ε του βέλτιστου κόστους, δηλαδή $\lambda \leq \varepsilon$ με $\varepsilon \geq 1$.[6]

Θεώρημα 8.3.1 *Έστω* Π *ένα πρόβλημα βελτιστοποίησης και* $\mathcal{N}$ *μία συνάρτηση γειτονιάς τέτοια ώστε το πρόβλημα τοπικής αναζήτησης* $\Pi/\mathcal{N}$ *να ανήκει στην PLS.*

1. *Αν* Π *είναι ισχυρά NP-δύσκολο (αντίστοιχα, NP-δύσκολο), τότε η* $\mathcal{N}$ *δεν μπορεί να είναι ακριβής εκτός εάν P=NP, (αντίστοιχα NP=co-NP).*

2. *Αν η προσέγγιση του* Π *σε λόγο* ϵ *είναι ισχυρά NP-δύσκολη (αντίστοιχα, NP-δύσκολη), τότε η* $\mathcal{N}$ *δεν εγγυάται λόγο r εκτός εάν P=NP, (αντίστοιχα NP=co-NP).*

Απόδειξη Έστω ότι το Π είναι ισχυρά NP-δύσκολο και ας θεωρήσουμε ένα στιγμιότυπο με πολυωνυμικά φραγμένα βάρη. Τότε ο κλασικός αλγόριθμος τοπικής αναζήτησης θα συγκλίνει σε πολυωνυμικό χρόνο. Αν, επιπλέον, η γειτονιά είναι ακριβής, τότε η λύση που θα υπολογισθεί θα είναι και ολικά βέλτιστη. Άρα θα έχουμε σε πολυωνυμικό χρόνο τη βέλτιστη λύση ενός NP-πλήρους προβλήματος, δηλαδή P=NP.

Γενικά υποθέστε ότι το Π είναι ένα NP-δύσκολο (πιθανά, όχι ισχυρά) πρόβλημα βελτιστοποίησης. Έστω ότι είναι πρόβλημα ελαχιστοποίησης. Τυπικά, το ακόλουθο πρόβλημα απόφασης είναι NP-πλήρες: Δεδομένου ενός στιγμιοτύπου x και μιας τιμής v, υπάρχει λύση με κόστος το πολύ v; Αν $\mathcal{N}$ είναι μία ακριβής γειτονιά, τότε μπορούμε να λύσουμε το συμπληρωματικό πρόβλημα απόφασης σε μη ντετερμινιστικό πολυωνυμικό χρόνο ως εξής: Δεδομένου του x και του v, μάντεψε μια λύση $\hat{s}$ και επαλήθευσε ότι η $\hat{s}$ είναι τοπικά (άρα και ολικά) βέλτιστη και ότι το κόστος της ικανοποιεί τη σχέση $f(\hat{s}) > v$. □

Τα αντίστοιχα θεωρήματα για τον προσεγγιστικό λόγο αποδεικνύονται με ανάλογο τρόπο.

Πρώτοι οι Παπαδημητρίου και Steiglitz έδωσαν τέτοιου είδους αποτελέσματα για το πρόβλημα του πλανόδιου πωλητή (TSP). Απέδειξαν ότι υποθέτοντας πως $P \neq NP$, εάν μπορούμε να ψάξουμε μία γειτονιά σε πολυωνυμικό χρόνο τότε αυτή δεν μπορεί να είναι ακριβής. Επιπλέον, με απουσία της ιδιότητας της τριγωνικής ανισότητας στα βάρη των ακμών, έδειξαν ότι υπάρχουν τοπικά βέλτιστα τα οποία είναι αυθαίρετα χειρότερα από το ολικό βέλτιστο. Στην πραγματικότητα έχει επίσης δειχθεί πως το να βρεις οποιονδήποτε

[6]Μερικές φορές χρησιμοποιείται το $\lambda = \frac{c_s}{c_{s^*}}$ και για τα προβλήματα μεγιστοποίησης οπότε $\varepsilon \leq 1$ και $\lambda \geq \varepsilon$.

πολυωνυμικού χρόνου προσεγγιστικό αλγόριθμο για το TSP που θα εγγυάται ένα άνω φράγμα για τον λόγο λ είναι NP-δύσκολο. Δηλαδή το ίδιο δύσκολο με το να λύσουμε ακριβώς το TSP κι έτσι ικανοποιείται και η δεύτερη υπόθεση του θεωρήματος, (με παρουσία της τριγωνικής ανισότητας, υπάρχει ο 3/2-προσεγγιστικός αλγόριθμος του Χριστοφίδη).

Παρά τα αρνητικά αποτελέσματα που είδαμε και αφορούν τις γενικές περιπτώσεις προβλημάτων, υπάρχουν πολλά αποτελέσματα ε-προσεγγιστικών αλγόριθμων τοπικής αναζήτησης που αφορούν υποπεριπτώσεις των γνωστών προβλημάτων. Αυτές οι υποπεριπτώσεις είναι είτε μη βεβαρημένες ή με πολυωνυμικά φραγμένα βάρη εκδοχές των προβλημάτων. Επομένως αφού ο κλασικός αλγόριθμος είναι ψευδοπολυωνυμικός θα τερματίζει σε πολυωνυμικό χρόνο για αυτά τα προβλήματα. Στη συνέχεια θα δούμε τρία τέτοια αποτελέσματα.

8.3.1 Max-Cut χωρίς βάρη

Για το πρόβλημα Max-Cut χωρίς βάρη στις ακμές του και γειτονιά την Flip, αποδεικνύεται εύκολα ότι ο κλασικός αλγόριθμος τοπικής αναζήτησης προσεγγίζει την ολικά βέλτιστη λύση με παράγοντα 2.

Θεώρημα 8.3.2 *Δεδομένου ενός στιγμιοτύπου x (ένας γράφος $G = (V, E)$) του προβλήματος Max-Cut χωρίς βάρη στις ακμές του, έστω $A = (V_1, V_2)$ μία τοπικά βέλτιστη διαμέριση κάτω από τη γειτονιά Flip και έστω ότι με $m_A(x)$ μετράμε το κόστος μιας τοπικά βέλτιστης διαμέρισης A του x κάτω από την Flip. Τότε*

$$m^*(x)/m_A(x) \leq 2$$

για κάθε A, όπου $m^*(x)$ η βέλτιστη διαμέριση.

Απόδειξη Έστω m το πλήθος των ακμών του γράφου. Αφού $m^*(x) \leq m$ αρκεί να δείξουμε ότι $m_A(x) \geq m/2$. Επίσης, έστω (V_{1k}, V_{2k}) η γειτονική λύση της (V_1, V_2) που προκύπτει από την αλλαγή πλευράς του κόμβου v_k.

Συμβολίζουμε με m_1 και m_2 το πλήθος των ακμών που ενώνουν κόμβους μέσα στο V_1 και στο V_2 αντίστοιχα. Έχουμε

$$m = m_1 + m_2 + m_A(x). \tag{8.1}$$

Δεδομένου οποιουδήποτε κόμβου v_i, ορίζουμε

$$m_{1i} = \{v | v \in V_1 \text{ και } (v, v_i) \in E\}$$

και

$$m_{2i} = \{v | v \in V_2 \text{ και } (v, v_i) \in E\}.$$

Αν $A = (V_1, V_2)$ είναι μία τοπικά βέλτιστη διαμέριση, για κάθε κόμβο v_k η λύση που προκύπτει από το (V_{1k}, V_{2k}) έχει τιμή το πολύ $m_A x$. Αυτό σημαίνει ότι για κάθε κόμβο $v_i \in V_1$,

$$|m_{1i}| - |m_{2i}| \leq 0$$

και για κάθε κόμβο $v_j \in V_2$,

$$|m_{2j}| - |m_{1j}| \leq 0.$$

Αθροίζοντας όλες τις κορυφές στα V_1 και V_2 παίρνουμε

$$\sum_{v_i \in V_1} (|m_{1i}| - |m_{2i}|) = 2m_1 - m_A(x) \leq 0$$

και

$$\sum_{v_j \in V_2} (|m_{2j}| - |m_{1j}|) = 2m_2 - m_A(x) \leq 0.$$

Επομένως $m_1 + m_2 - m_A(x) \leq 0$. Από αυτήν την ανισότητα και την εξίσωση (8.1) συνεπάγεται ότι $m_A(x) \geq m/2$. □

8.3.2 Max-k-Sat χωρίς βάρη

Το πρόβλημα Max-Sat το ορίσαμε στην ενότητα 8.2.2. Εδώ θα δούμε ότι στην μη βεβαρημένη εκδοχή του, με k ακριβώς στοιχεία σε κάθε όρο, μπορούμε, μέσω της τοπικής αναζήτησης, να προσεγγίσουμε το ολικό βέλτιστο κατά ένα παράγοντα $\frac{k}{k+1}$. Η γειτονιά που χρησιμοποιούμε είναι η Flip.

Θεώρημα 8.3.3 *Έστω m και m_{loc} το πλήθος των ικανοποιημένων όρων σε ένα ολικό και σε ένα τοπικό βέλτιστο ενός στιγμιοτύπου του Max-k-Sat προβλήματος χωρίς βάρη, αντίστοιχα. Τότε ισχύει $m_{loc} \geq \frac{k}{k+1} m$*

Απόδειξη Χωρίς απώλεια της γενικότητας μπορούμε να υποθέσουμε ότι στο τοπικό βέλτιστο κάθε μεταβλητή έχει την τιμή αληθής. Αν δεν συμβαίνει αυτό, θέτοντας $x'_i = \bar{x}_i$ όπου $x_i \leftarrow$ ψευδές, και $x'_i = x_i$ όπου $x_i \leftarrow$ αληθές στο τοπικό βέλτιστο, παίρνουμε ένα ισοδύναμο στιγμιότυπο για το οποίο ισχύει η υπόθεση.

Έστω δ_i η διαφορά του πλήθους των όρων που ικανοποιούνται όταν η μεταβλητή x_i αλλάζει τιμή. Αφού η ανάθεση είναι τοπικό βέλτιστο, αλλάζοντας τιμή σε οποιαδήποτε μεταβλητή θα μειώνεται το πλήθος των ικανοποιημένων όρων, δηλαδή $\delta_i \leq 0$, για $1 \leq i \leq n$.

Έστω cov_s το υποσύνολο των όρων που έχουν ακριβώς s στοιχεία αληθή με την παρούσα ανάθεση (δηλαδή, οι αληθείς θετικές μεταβλητές συν τις ψευδείς αρνητικές είναι s) και $cov_s(l)$ το πλήθος των όρων της cov_s που περιέχουν το στοιχείο l.

Έχουμε $\delta_i = -cov_1(x_i) + cov_0(\bar{x}_i)$. Πράγματι, όταν το x_i αλλάζει τιμή από αληθές σε ψευδές χάνονται οι όροι που περιέχουν το x_i σαν το μόνο αληθές στοιχείο, δηλ. οι $cov_1(x_i)$ και κερδίζονται οι όροι που δεν έχουν αληθές στοιχείο αλλά περιέχουν το $\bar{x}_i$, δηλ. οι $cov_0(\bar{x}_i)$.

Αθροίζοντας πάνω σε όλες τις μεταβλητές παίρνουμε

$$\sum_{i=1}^{n} \delta_i \leq 0,$$

οπότε

$$\sum_{i=1}^{n} cov_0(\bar{x}_i) \leq \sum_{i=1}^{n} cov_1(x_i).$$

Χρησιμοποιώντας τις ισότητες (8.1) και (8.2)

$$\sum_{i=1}^{n} cov_1(x_i) = |cov_1| \tag{8.1}$$

$$\sum_{i=1}^{n} cov_0(\bar{x}_i) = k|cov_0|, \tag{8.2}$$

οι οποίες μπορούν εύκολα να επαληθευτούν παίρνουμε

$$k|cov_0| \leq |cov_1| \leq m_{loc}.$$

Οπότε

$$m = m_{loc} + |cov_0| \leq (1 + \frac{1}{k})m_{loc} = \frac{k+1}{k}m_{loc}.$$

□

8.3.3 Πρόβλημα Περιοδεύοντος Πωλητή (TSP)

Το πρόβλημα του περιοδεύοντος πωλητή το συναντήσαμε και αυτό στην ενότητα 8.2.2. Είναι από τα πιο ''δύσκολα'' προβλήματα συνδυαστικής βελτιστοποίησης. Ακόμα και στην εκδοχή του όπου όλα τα βάρη στις ακμές είναι ή 1 ή 2 παραμένει NP-δύσκολο. Εδώ θα δείξουμε ότι ο αλγόριθμος τοπικής αναζήτησης με γειτονιά την 2-Opt είναι 3/2-προσεγγιστικός.

Θεώρημα 8.3.4 *Ο αλγόριθμος τοπικής αναζήτησης με γειτονιά την 2-Opt είναι 3/2 προσεγγιστικός για το TSP(1,2).*

Απόδειξη Έστω $C = v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}, v_{\pi_1}$ η τοπικά βέλτιστη λύση με γειτονιά την 2-Opt. Έστω O μία οποιαδήποτε βέλτιστη λύση. Για κάθε μοναδιαίου βάρους ακμή e της O αντιστοιχούμε μία μοναδιαίου βάρους ακμή e' στη C ως εξής. Έστω $e = (v_{\pi_i}, v_{\pi_j})$ με $i < j$. Αν $j = i + 1$ τότε $e' = e$, αλλιώς η e' είναι μία μοναδιαίου κόστους ακμή ανάμεσα στις $e_1 = (v_{\pi_i}, v_{\pi_{i+1}})$ και $e_2 = (v_{\pi_j}, v_{\pi_{j+1}})$. Πράγματι, είτε η e_1 ή η e_2 πρέπει να είναι μοναδιαίου κόστους. Αν όχι τότε η λύση C' που προκύπτει από την C με την αφαίρεση των e_1 και e_2 και την προσθήκη των e και $f = (v_{\pi_{i+1}}, v_{\pi_{j+1}})$ έχει κόστος τουλάχιστον 1 λιγότερο από της C οπότε η C δεν θα ήταν τοπικά βέλτιστη λύση.

Έστω ότι το U_O αναπαριστά το σύνολο των ακμών μοναδιαίου κόστους της O και το U_C το σύνολο των ακμών μοναδιαίου κόστους της C που προέκυψαν από το U_O χρησιμοποιόντας την παραπάνω απεικόνιση. Αφού μία ακμή $e' = (v_{\pi_i}, v_{\pi_{i+1}})$ στο U_C μπορεί να είναι μόνο η εικόνα των μοναδιαίου κόστους ακμών που προσπίπτουν στο v_{π_i} της O και αφού η O είναι απλός κύκλος, θα υπάρχουν το πολύ δύο ακμές στο U_O οι οποίες θα απεικονίζονται στην e'. Έτσι

$$|U_C| \geq |U_O|/2$$

και προκύπτει

$$\frac{cost(C)}{cost(O)} \leq \frac{|U_O|/2 + 2(n - |U_O|/2)}{|U_O| + 2(n - |U_O|)} \leq \frac{3}{2}.$$

□

Αξίζει να αναφέρουμε όμως και το ακόλουθο θεώρημα για το μετρικό TSP, (στο μετρικό TSP ισχύει η τριγωνική ανισότητα στα βάρη των ακμών).

Θεώρημα 8.3.5 *Ένας αλγόριθμος τοπικής αναζήτησης με γειτονιά την 2-Opt είναι $4\sqrt{n}$-προσεγγιστικός για το μετρικό TSP, όπου n το πλήθος των κόμβων του προβλήματος.*

8.4 Επεκτάσεις της Τοπικής Αναζήτησης και Συσχετισμοί της με Άλλα Αλγοριθμικά Πεδία

Σε αυτήν την τελευταία ενότητα θα παρουσιάσουμε, εν συντομία, κάποιες επεκτάσεις της τοπικής αναζήτησης, όπως τα ε-τοπικά βέλτιστα, η μη-επιλήσμων (non-oblivious) τοπική αναζήτηση και οι μετα-ευριστικοί αλγόριθμοι (metaheuristics). Επίσης θα αναφερθούμε στη σχέση της τοπικής αναζήτησης με τις θεωρίες τοπίων (landscape theory) και παιγνίων (game theory).

8.4.1 ε-Τοπικά βέλτιστα

Η έννοια του *ε-τοπικού βέλτιστου* ορίζεται ως η λύση $\bar{S}$, για της οποίας το κόστος ισχύει $cost(\bar{S}) \leq (1+\varepsilon)cost(S)$, για κάθε $S \in N(\bar{S})$. Για την εύρεση τέτοιων λύσεων υπάρχει ένα FPTAS το οποίο απαιτεί χρόνο $O(n^2\varepsilon^{-1}logn)$ και θα το αναπτύξουμε στη συνέχεια.

Πριν από αυτό, όμως, θα χρειαστεί να ορίσουμε *τα προβλήματα συνδυαστικής βελτιστοποίησης σαν μια ισοδύναμη κλάση προβλημάτων με αυτή των προβλημάτων 0/1-ακέραιου γραμμικού προγραμματισμού.* Έτσι έχουμε ότι ένα πρόβλημα συνδυαστικής βελτιστοποίησης Π αποτελείται από μία συλλογή στιγμιοτύπων $(\mathcal{F}, c)$, όπου το σύνολο $\mathcal{F}$ των εφικτών λύσεων είναι μια οικογένεια υποσυνόλων ενός πεπερασμένου βασικού συνόλου $E = \{1, \ldots, n\}$. Κάθε $e \in E$ έχει κόστος c_e. Η αντικειμενική συνάρτηση $c : E \to \mathbb{Q}_+$ αναθέτει ένα μη αρνητικό κόστος σε κάθε εφικτή λύση $S \in \mathcal{F}$ έτσι ώστε $c(S) := \sum_{e \in S} c_e$. Σκοπός είναι φυσικά να βρεθεί μία εφικτή λύση S^* τέτοια ώστε

$c(S^*) \leq c(S)$ για κάθε $S \in \mathcal{F}$. Υποθέτουμε ότι το Π είναι κλειστό στην κλιμάκωση των όρων της αντικειμένικής συνάρτησης, δηλαδή αν $(\mathcal{F}, c) \in \Pi$, τότε $(\mathcal{F}, c') \in \Pi$ για κάθε $c' \in \mathbb{Q}_+$. Για τεχνικούς λόγους υποθέτουμε, εδώ, επίσης ότι $c(S) \neq 0$ για $S \in \mathcal{F}$. Οι ακόλουθοι ορισμοί και τα αποτελέσματα αναφέρονται σε προβλήματα ελαχιστοποίησης αλλά μπορούν εύκολα να επεκταθούν και σε προβλήματα μεγιστοποίησης.

Ορίζοντας ένα ε-τοπικό βέλτιστο ως μία εφικτή λύση S^ε για την οποία ισχύει

$$\frac{c(S^\varepsilon) - c(S)}{c(S)} \leq \varepsilon \text{ για κάθε } S \in N(S^\varepsilon),$$

για κάποιο $\varepsilon > 0$, εννοούμε ότι η S^ε είναι ''σχεδόν'' ένα τοπικό βέλτιστο, δηλ. ''σχεδόν'' μικρότερη από τους γείτονές της. Παρατηρήστε ότι ένα ε-τοπικό βέλτιστο μπορεί να είναι αυθαίρετα μακριά από ένα τοπικό βέλτιστο. Μία οικογένεια αλγόριθμων $(A_\varepsilon)_{\varepsilon>0}$ για το Π είναι ένα *σχήμα ε-τοπικής βελτιστοποίησης* αν το A_ε παράγει ένα ε-τοπικό βέλτιστο. Αν ο χρόνος εκτέλεσης του αλγόριθμου A_ε είναι πολυωνυμικό ως προς το μέγεθος της εισόδου και του $1/\varepsilon$ τότε καλείται *σχήμα ε-τοπικής βελτιστοποίησης πλήρους πολυωνυμικού χρόνου.*

Ο αλγόριθμος στο Σχήμα 8.11 αρχίζει με μία εφικτή λύση S^0 και στη συνέχεια αλλάζει τα κόστη των στοιχείων c_e, για $e \in E$, σύμφωνα με έναν καθορισμένο κανόνα κλιμάκωσης, για να δημιουργήσει ένα τροποποιημένο στιγμιότυπο (Βήματα 1 και 2). Σημειώστε ότι η συνάρτηση $IMPROVE_N(\cdot, \cdot)$ είναι ο αλγόριθμος C_Π που είδαμε στον ορισμό 8.2.1 της PLS και ότι η πράξη $c'_e := \left\lceil \frac{c_e}{q} \right\rceil q$ στογγυλοποιεί τα c_e στο κοντινότερο ακέραιο πολλαπλάσιο του q. Χρησιμοποιώντας την τοπική αναζήτηση στο τροποποιημένο πρόβλημα, στο Βήμα 4 ο αλγόριθμος ψάχνει για ένα τοπικό βέλτιστο σύμφωνα με την ίδια γειτονιά και *τη νέα συνάρτηση κόστους* c'. Τα τοπικά βέλτιστα του τροποποιημένου στιγμιοτύπου είναι ε-τοπικά βέλτιστα του αρχικού στιγμιοτύπου, άρα ο αλγόριθμος τερματίζει όταν βρεθεί ένα τέτοιο βέλτιστο. Αν κατά τη διάρκεια αυτής της αναζήτησης το κόστος της τρέχουσας λύσης γίνει μικρότερο από $c(S^0)/2$ τότε ο αλγόριθμος θέτει ως αρχική λύση την τρέχουσα, υπολογίζει μία νέα στογγυλοποίηση των c_e σύμφωνα με αυτήν τη λύση και επαναλαμβάνει την διαδικασία της τοπικής αναζήτησης για το νέο στιγμιότυπο κ.ο.κ. (Βήμα 5).

Βήμα 1: $i := 0$;
Βήμα 2: $K := c(S^i); q := \frac{K\varepsilon}{2n(1+\varepsilon)}$ και $c'_e := \left\lceil \frac{c_e}{q} \right\rceil q$ για κάθε $e \in E$;
Βήμα 3: $k := 0$ και $S^{i,k} := S^i$;
Βήμα 4: **επανέλαβε**
Κάλεσε την $IMPROVE_N(S^{i,k}, c')$;
Αν η απάντηση είναι ''όχι'' **τότε**
$S^{i,k+1} \in N(S^{i,k})$ έτσι ώστε $c'(S^{i,k+1}) < c'(S^{i,k})$;
$k := k + 1$;
αλλιώς $S^\varepsilon := S^{i,k}$; **τερμάτισε.**
μέχρις ότου $c(S^{i,k}) \leq K/2$
Βήμα 5: $S^{i+1} := S^{i,k}; i := i + 1;$ **πήγαινε στο** Βήμα 2.

Σχήμα 8.11: Αλγόριθμος ε-Τοπικής Αναζήτησης

Αυτός ο αλγόριθμος παράγει ένα ε-τοπικό βέλτιστο. Δείτε ότι

$$c(S^\varepsilon) = \sum_{e \in S^\varepsilon} c_e \leq \sum_{e \in S^\varepsilon} \left\lceil \frac{c_e}{q} \right\rceil q \leq$$

$$\sum_{e \in S} \left\lceil \frac{c_e}{q} \right\rceil q \leq \sum_{e \in S} q \left(\frac{c_e}{q} + 1 \right) \leq \sum_{e \in S} c_e + nq = c(S) + nq,$$

όπου $n = |E|$. Σε συνδυασμό με το ότι $c(S^\varepsilon) \geq K/2$, συνεπάγεται

$$\frac{c(S^\varepsilon) - c(S)}{c(S)} \leq \frac{nq}{c(S)} \leq \frac{nq}{c(S^\varepsilon) - nq} \leq \frac{2nq}{K - 2nq} = \varepsilon.$$

Ο αλγόριθμος καλεί τη συνάρτηση $IMPROVE$, ανάμεσα σε δύο συνεχόμενες επαναλήψεις του Βήματος 2, $O(n/\varepsilon)$ φορές, αφού σε κάθε κίνηση βελτίωσης η τιμή της αντικειμενικής συνάρτησης μειώνεται τουλάχιστον κατά q μονάδες. Το Βήμα 2 εκτελείται το πολύ $logc(S^0)$ φορές, έτσι $O(n\varepsilon^{-1}logc(S^0))$ γείτονες ερευνώνται. Ωστόσο μπορεί κανείς να δείξει ότι το Βήμα 2 εκτελείται το πολύ $O(nlogn)$ φορές. Επομένως, αν υποθέσουμε ότι οι γείτονες αναζητούνται αποδοτικά και ότι η αρχική λύση μπορεί να παραχθεί σε πολυωνυμικό χρόνο, πράγματα που ισχύουν για όλα τα προβλήματα της PLS, ο χρόνος εκτέλεσης του αλγόριθμου είναι $O(n^2\varepsilon^{-1}logn)$.

8.4.2 Μη-επιλήσμων (non-oblivious) τοπική αναζήτηση

Η μη-επιλήσμων τοπική αναζήτηση είναι μία επέκταση της τοπικής αναζήτησης, αφού σε αυτήν επιτρέπεται η συνάρτηση κόστους, που χρησιμοποιείται για την εύρεση ενός τοπικού βέλτιστου, να είναι διαφορετική από αυτήν του αρχικού προβλήματος. Έτσι η αναζήτηση μπορεί να κατευθυνθεί προς λύσεις καλύτερης ποιότητας.

Τα δύσκολα προβλήματα της κλάσης MAX-SNP δεν έχουν PTAS, εκτός αν P-NP. Αποδεικνύεται, ωστόσο, ότι κάθε MAX-SNP πρόβλημα μπορεί να προσεγγιστεί κατά ένα σταθερό παράγοντα με αυτή τη μέθοδο.

Τα προβλήματα συνδυαστικής βελτιστοποίησης που ανοίκουν στην κλάση GLO (guaranteed local optima) έχουν την ιδιότητα να τα τοπικά τους βέλτιστα να προσεγγίζουν τα ολικά κατά ένα σταθερό παράγοντα. Τα προβλήματα που είδαμε στην προηγούμενη ενότητα ανοίκουν σε αυτήν την κλάση. Η Κομβική Επικάλυψη Vertex Covering, όμως δεν ανοίκει στην GLO ενώ ανοίκει στη MAX-SNP, έτσι έχουμε

$$\text{GLO} \subset \text{non-oblivious GLO}.$$

Άρα, η μη-επιλήσμων τοπική αναζήτηση είναι πιο ισχυρή από την κλασική τοπική αναζήτηση.

8.4.3 Μεταευριστικοί αλγόριθμοι (Metaheuristics)

Οι μεταευριστικοί αλγόριθμοι προσπαθούν να εξερευνήσουν το χώρο των λύσεων με έναν πιο αποδοτικό τρόπο αποφεύγοντας τα τοπικά βέλτιστα και στοχεύοντας σε περιοχές με καλές λύσεις. Μπορούν να χωριστούν σε μεθόδους τροχιάς, οι οποίες περνούν κάθε φορά από τη μια λύση στην άλλη, και σε μεθόδους βασιζόμενες σε πληθυσμό, οι οποίες έχουν να κάνουν με πολλές λύσεις σε κάθε επανάληψη. Η απόδοση τέτοιων μεθόδων είναι καλή γι΄ αυτό χρησιμοποιούνται ευρέως. Δυστυχώς μπορούν να μελετηθούν μόνο εμπειρικά και θα ήταν μάλλον δύσκολο να έχουμε κάποια έστω φτωχά θεωρητικά συμπεράσματα σχετικά με αυτές. Οι ακόλουθες μέθοδοι είναι οι πιο συχνά χρησιμοποιούμενες στα προβλήματα συνδυαστικής βελτιστοποίησης και όλες χρησιμοποιούν την τοπική αναζήτηση.

Η *Προσομοιωμένη Ανώπτωση (Simulated Annealing)* ξεκινάει με μία αρχική εφικτή λύση, διαλέγει μία τυχαία γειτονική της λύση και αν αυτή είναι καλύτερη τότε πηγαί-

νει σε αυτή και επαναλαμβάνει με τρέχουσα λύση τη νέα. Αν η γειτονική λύση είναι χειρότερη από την τρέχουσα λύση τότε, με βάση μια πιθανότητα, δέχεται να πάει σε αυτή ή διαλέγει μια άλλη γειτονική λύση και επαναλαμβάνει. Η πιθανότητα γενικώς υπολογίζεται ακολουθώντας την κατανομή Boltzmann, $exp(\frac{f(s)-f(s')}{T})$, όπου $s' \in N(s)$ και το T, που ονομάζεται ''θερμοκρασία'', μειώνεται κατά τη διάρκεια της αναζήτησης. Η ιδέα της μεθόδου αυτής είναι να αρχίσει σαν ένας τυχαίος περίπατος με σκοπό να ξεφύγει από κακά τοπικά βέλτιστα και σταδιακά να γίνει ο κλασικός αλγόριθμος τοπικής αναζήτησης, ελπίζοντας ότι αυτός θα συγκλίνει σε ένα καλό λογικό βέλτιστο.

Η *Αναζήτηση Ταμπού(Tabu Search)* είναι ένας αλγόριθμος τοπικής αναζήτησης, πρώτα-ο-καλύτερος (δηλ. διαλέγει να μεταβεί πάντα στην καλύτερη από τις γειτονικές λύσεις), ο οποίος χρησιμοποιεί μία λίστα Ταμπού, όπως λέγεται, με λύσεις που δεν λαμβάνονται υπ' όψη κατά τη διάρκεια της βελτίωσης. Η λίστα ταμπού αρχικά είναι κενή και ανάλογα με το μέγεθός της οι λύσεις που έχουν ήδη επισκεφτεί προσθέτονται σε αυτή σε μορφή ουράς (FIFO). Αυτό επιτρέπει στην διαδικασία της αναζήτησης να πάει σε χειρότερες λύσεις αν οι καλύτερες λύσεις είναι στη λίστα ταμπού ή αν δεν υπάρχουν καλύτερες λύσεις. Ο αλγόριθμος τερματίζει όταν ικανοποιηθεί κάποια συνθήκη. Αυτό το είδος βραχύχρονης μνήμης κατά τη διάρκεια της αναζήτησης βοηθάει να αποφευχθούν τοπικά ελάχιστα αλλά και ατέρμονες κύκλοι. Σημειώστε ότι αν το μέγεθος της λίστας είναι μικρό τότε η αναζήτηση επικεντρώνεται σε μικρές περιοχές του πεδίου αναζήτησης, ενώ μια μεγάλη λίστα ωθεί τη διαδικασία της αναζήτησης στο να εξερευνά μεγαλύτερες περιοχές. Συνήθως, αντί για λεπτομερείς λύσεις, η λίστα ταμπού μπορεί να καταχωρεί χαρακτηριστικά λύσεων, με σκοπό να είναι πιο αποδοτική.

Η *Διαδικασία Άπληστης Στοχαστικής Προσαρμόσιμης Αναζήτησης* (Greedy Randomized Adaptive Search Procedure - *GRASP*) χρησιμοποιεί μία άπληστη στοχαστική μέθοδο για να κατασκευάσει, βήμα βήμα, αρχικές λύσεις από υποσχόμενες περιοχές του χώρου αναζήτησης. Μετά εφαρμόζει έναν κλασικό αλγόριθμο τοπικής αναζήτησης σε κάθε λύση με σκοπό να τις βελτιώσει και τελικά επιστρέφει την καλύτερη λύση από αυτές που βρήκε.

Στην *Αναζήτηση Μεταβλητής Γειτονιάς (Variable Neighborhood Search)* επιλέγεται ένα σύνολο από δομές γειτονιάς $\mathcal{N}_k, k = 1, \ldots, k_{max}$. Συνήθως αυτές οι γειτονιές είναι αυξανόμενης πληθικότητας. Ξεκινώντας από μία αρχική λύση, επιλέγεται τυχαία ένας γείτονας από μία από αυτές τις γειτονιές. Στη συνέχεια ξεκινάει μια τοπική αναζήτηση με αυτόν το γείτονα σαν αρχική λύση. Η τοπική αναζήτηση μπορεί να χρησιμοποιήσει

οποιαδήποτε γειτονιά και όχι μόνο την αρχική. Η λύση που θα βρεθεί συγκρίνεται με την αρχική και αν είναι καλύτερη τότε ο αλγόριθμος επαναλαμβάνεται με σημείο εκκίνησης τη νέα λύση. Διαφορετικά ένας νέος γείτονας επιλέγεται τυχαία με βάση κάποια άλλη συνάρτηση γειτονιάς κ.ο.κ., μέχρις ότου ικανοποιηθεί κάποια συνθήκη τερματισμού.

Η *Κατευθυνόμενη Τοπική Αναζήτηση (Guided Local Search)*, αντί να αλλάζει τη δομή γειτονιάς ή τις λύσεις που θεωρεί κάθε φορά, δυναμικά αλλάζει τη συνάρτηση κόστους έτσι ώστε να κάνει το τρέχον τοπικό βέλτιστο ''λιγότερο επιθυμητό''. Αυτό επιτυγχάνεται με την προσθήκη ενός όρου, ο οποίος εξαρτάται από τα χαρακτηριστικά των λύσεων. Όπως είναι για παράδειγμα οι ακμές του TSP πολλαπλασιασμένες, η κάθε μία, με έναν παράγοντα ποινής. Όσο υψηλότερης σημασίας είναι ένα χαρακτηριστικό τόσο μεγαλύτερο είναι το κόστος της μη ύπαρξής του στη λύση. Οι παράγοντες ποινής αλλάζουν κάθε φορά που αρχίζει μία νέα τοπική αναζήτηση.

Η *Επαναλαμβανόμενη Τοπική Αναζήτηση (Iterated Local Search)* είναι μία γενική μέθοδος που μπορεί να χρησιμοποιηθεί σαν πλαίσιο για άλλες μεθόδους. Ξεκινώντας με μία αρχική λύση, βρίσκει ένα τοπικό βέλτιστο, το διαταράσσει και ξαναρχίζει την τοπική αναζήτηση από εκείνο το σημείο. Το βήμα της διατάραξης είναι πολύ σημαντικό αφού είναι υπεύθυνο για την αποτελεσματικότητα της μεθόδου. Θα θέλαμε να αποδράσουμε από την λεκάνη έλξης του τοπικού ελάχιστου που είμαστε αλλά όχι και να κάνουμε απλά πολλές τυχαίες τοπικές αναζητήσεις. Έτσι ένα καλό κριτήριο είναι να συγκρίνουμε το νέο τοπικό βέλτιστο που πήραμε με το παλιό έτσι ώστε να δούμε αν είμαστε σε μια περιοχή του χώρου αναζήτησης με καλύτερες λύσεις.

Τέλος, μία μέθοδος βασισμένη σε πληθυσμό, που καλείται *Εξελικτικός Υπολογισμός (Evolutionary Computation) ή Γενετικοί Αλγόριθμοι (Genetic Algorithms)*, εμπνεύστηκε από την ικανότητα της φύσης να εξελίσσει τα έμψυχα όντα που είναι καλά προσαρμοσμένα στο περιβάλλον τους. Έτσι, ξεκινώντας από έναν αρχικό πληθυσμό λύσεων και χρησιμοποιώντας πάνω τους μερικούς τελεστές, όπως *ανασυνδυασμός*, *μετάλλαξη* και *επιλογή* κάποιων να συνεχίσουν να ζουν με βάση την *υγεία* τους, αυτοί αλγόριθμοι συγκλίνουν σε καλύτερους πληθυσμούς λύσεων.

8.4.4 Θεωρία Τοπίων (Landscape Theory)

Για να αιτιολογήσουμε θεωρητικά το γιατί μία γειτονιά είναι καλύτερη στην πράξη από μία άλλη για ένα δεδομένο πρόβλημα βελτιστοποίησης χρησιμοποιούμε τη θεωρία

τοπίων. Ένα τοπίο σχηματίζεται από τη συνάρτηση κόστους και τη γειτονιά. Προφανώς, ένα τραχύ τοπίο έχει πολλά τοπικά βέλτιστα και γι΄ αυτό δεν είναι τόσο κατάλληλο για την τοπική αναζήτηση όσο είναι ένα επίπεδο τοπίο. Οι μεταευριστικοί που βασίζονται στην τοπική αναζήτηση προσπαθούν να ξεπεράσουν το πρόβλημα της παγίδευσης σε κακά τοπικά βέλτιστα χρησιμοποιώντας διάφορες τεχνικές, αλλά εξαρτάται κυρίως από το τοπίο του προβλήματος το πόσο καλά θα τα καταφέρουν. Έτσι, μετριέται η τραχύτητα του τοπίου και μπορεί να προκύψει μία ιεραρχία των προβλημάτων συνδυαστικής βελτιστοποίησης σε σχέση με αυτή.

Πιο συγκεκριμένα, για αυτό το σκοπό χρησιμοποιήται ο *συντελεστής αυτοσυσχετισμού*. Ο ορισμός της *συνάρτησης αυτοσυσχετισμού* τοπίου είναι

$$\rho(d) = 1 - \frac{\langle (C(x) - C(y))^2 \rangle_{d(x,y)=d}}{\langle (C(x) - C(y))^2 \rangle}$$

όπου $C(\cdot)$ η συνάρτηση κόστους και $\langle (C(x) - C(y))^2 \rangle$ η μέση τιμή του $(C(x) - C(y))^2$ όλων των ζευγαριών $\{x, y\}$ τα οποία βρίσκονται σε απόσταση d. Αυτή η ποσότητα δείχνει το βαθμό συσχετισμού ανάμεσα σε οποιεσδήποτε δύο λύσεις οι οποίες βρίσκονται σε απόσταση d η μία από την άλλη. Η πιο σημαντική τιμή που θέλουμε να ξέρουμε είναι η $\rho(1)$, επειδή η σύνδεση δύο γειτονικών λύσεων είναι η πλέον σημαντική για κάθε αλγόριθμο τοπικής αναζήτησης. Μία τιμή κοντά στο 1 για το $\rho(1)$ υποδηλώνει ότι τα κόστη οποιονδήποτε δύο γειτονικών λύσεων είναι κατά μέσω όρο πολύ κοντά και μπορούμε να θεωρήσουμε ότι το τοπίο είναι επίπεδο. Αντίθετα, μία τιμή κοντά στο 0 υποδηλώνει ότι τα κόστη οποιονδήποτε δύο γειτονικών λύσεων είναι σχεδόν ανεξάρτητα και μπορούμε να θεωρήσουμε ότι ένα τέτοιο τοπίο είναι απόκρημνο. Ο συντελεστής αυτοσυσχέτισης ξ ορίζεται ως $\xi = 1/(1 - \rho(1))$ και μπορεί να υπολογιστεί από ένα θεώρημα που βασίζεται στην ανάλυση του τοπίου κατά Fourier.

8.4.5 Τοπική Αναζήτηση και Θεωρία Παιγνίων

Η πολυπλοκότητα της εύρεσης αιτιοκρατικών Nash σημείων ισορροπίας (pure Nash equilibria) στα παίγνια αναδεικνύει τη σχέση της θεωρίας της PLS με τη θεωρία παιγνίων. Δύο είναι τα βασικά ερωτήματα. Η πρώτη ερώτηση είναι ποιου είδους παίγνια έχουν (pure Nash equilibria) και η δεύτερη πόσο δύσκολο είναι να τα βρούμε.

Ας δώσουμε πρώτα μερικούς χρήσιμους ορισμούς πριν δούμε κάποιες απαντήσεις στα

παραπάνω ερωτήματα. Ένα *παιχνίδι* με $n \geq 2$ *παίχτες* αποτελείται από ένα πεπερασμένο σύνολο από *ενέργειες (στρατηγικές)* S_i, για κάθε παίχτη i, και μία *συνάρτηση ωφέλειας* $u_i : S_1 \times \cdots \times S_n \longrightarrow \mathbb{Z}$ η οποία απεικονίζει το καρτεσιανό γινόμενο των ενεργειών στους ακεραίους, επίσης για κάθε παίχτη i. Τα στοιχεία $s = (s_1, \ldots, s_n)$ του συνόλου $S_1 \times \cdots \times S_n$, όπου $s_i \in S_i$ για κάθε $i = 1, \ldots, n$, καλούνται *συνδυασμοί ενεργειών ή καταστάσεις*. Κάθε παίχτης επιλέγει να παίξει εκείνη τη στρατηγική, η οποία μεγιστοποιεί το όφελός του, δεδομένης μιας κατάστασης. Σε αυτήν την περίπτωση, οι παίχτες καλούνται *εγωιστές*, αφού ο καθένας δρα ανεξάρτητα, με βάση μόνο το προσωπικό του όφελος. Ένα *(αιτιοκρατικό) Nash σημείο ισορροπίας* είναι μία κατάσταση $s = (s_1, \ldots, s_n)$ τέτοια ώστε, για κάθε παίχτη i, να ισχύει $u_i(s_1, \ldots, s_i, \ldots, s_n) \geq u_i(s_1, \ldots, s'_i, \ldots, s_n)$, για κάθε $s'_i \in S_i$. Δηλαδή μία κατάσταση στην οποία κανένας παίχτης δεν ωφελείται με το να αλλάξει μεμονωμένα την στρατηγική του.

Ας δούμε τώρα την ισοδυναμία του Γράφου Μετάβασης του ορισμού 8.2.4, όταν οι γειτονικές λύσεις διαφέρουν στην τιμή ενός μόνο συστατικού της λύσης (μεταβλητής, κόμβου) με τον ακόλουθο γράφο. Για κάθε κατάσταση ενός παίγνιου υπάρχει ένας κόμβος και αν δύο καταστάσεις διαφέρουν μόνο στη στρατηγική ενός παίχτη τότε υπάρχει ένα τόξο ανάμεσά τους. Η κατεύθυνση του τόξου είναι από τον κόμβο με το μικρότερο όφελος, για τον παίχτη που αλλάζει στρατηγική, προς τον άλλο κόμβο. Αν αυτός ο γράφος είναι ακυκλικός τότε υπάρχουν (pure Nash equilibria) τα οποία είναι τα πηγάδια του γράφου, δηλ. τα τοπικά του βέλτιστα.

Υπάρχει μία κλάση παιγνίων, τα *παίγνια συμφόρησης* για τα οποία αποδεικνύεται ότι έχουν (pure Nash equilibria). Αυτά τα παίγνια αναφέρονται σε παίκτες, οι οποίοι χρησιμοποιούν ένα σύνολο από πόρους που η ταυτόχρονη χρήση τους αυξάνει το κόστος τους. Η απόδειξη γίνεται με τη βοήθεια του επιχειρήματος της συνάρτησης δυναμικού. *Η συνάρτηση δυναμικού (potential function)* είναι μία συνάρτηση, η οποία εάν μεταβάλλεται το όφελος οποιουδήποτε παίχτη, μεταβάλλεται και η τιμή αυτής κατά το ίδιο ποσό. Οπότε τον ίδιο ρόλο που παίζει η συνάρτηση κόστους για τις λύσεις των προβλημάτων, παίζει αυτή για τις καταστάσεις των παιγνίων. Αποδεικνύεται λοιπόν ότι υπάρχει μια τέτοια συνάρτηση δυναμικού για αυτά τα παίγνια και επομένως ο γράφος με τις καταστάσεις έχει πηγάδια.

Το πρόβλημα της εύρεσης αιτιοκρατικών Nash σημείων ισορροπίας αποδεικνύεται ότι είναι πολυωνυμικό για μία απλή περίπτωση των παιγνίων συμφόρησης ενώ οι γενικότερες περιπτώσεις είναι PLS-πλήρεις.

Τέλος, σχετικά με την ύπαρξη τέτοιων σημείων ισορροπίας υπάρχει άλλο ένα αποτέλεσμα. Αν επεκτείνουμε την έννοια της συνάρτησης δυναμικού έτσι ώστε μια *γενική συνάρτηση δυναμικού (general potential function)* να είναι μία συνάρτηση η οποία μεταβάλλεται σύμφωνα με τις μεταβολές των οφελών των παικτών μόνο κατά το ίδιο πρόσημο και όχι κατ΄ ανάγκη και κατά την ίδια ποσότητα, τότε το επιχείρημα που είπαμε προηγουμένως εξακολουθεί να ισχύει. Τα παίγνια που έχουν τέτοια συνάρτηση λέγονται *παίγνια γενικού δυναμικού (general potential games)*. Αποδεικνύεται ότι έχουν αντιστοιχία 1-1 με τα προβλήματα της PLS, έτσι που τα pure Nash equilibria των μεν να ταυτίζονται με τα τοπικά βέλτιστα των δε.

Κεφάλαιο 9

Αξιολόγηση Ευριστικών Αλγορίθμων

Πολλές φορές βλέπουμε ότι η εύρεση της βέλτιστων λύσεων προβλημάτων ακέραιου γραμμικού προγραμματισμού είναι μια χρονοβόρα διαδικασία (εκθετική πολυπλοκότητα). Υπάρχουν γρήγορες μέθοδοι επίλυσης (πολυωνυμική πολυπλοκότητα) που δεν μας εγγυούνται όμως ότι η λύση θα είναι βέλτιστη. Σε αυτές τις περιπτώσεις, προσπαθούμε να αξιολογήσουμε αυτούς τους προσεγγιστικούς αλγορίθμους σύμφωνα με το πόσο απέχει η λύση που επιστρέφουν από τη βέλτιστη.

Θεωρούμε έναν προσεγγιστικό αλγόριθμο A και ένα στιγμιότυπο του προβλήματος I. Ορίζουμε τις παρακάτω ποσότητες:

$f_A(I)$: η λύση που επιστρέφει ο A

$f_*(I)$: η βέλτιστη λύση

$\epsilon^\alpha = |f_*(I) - f_A(I)|$: απόλυτο σφάλμα

$\epsilon^r = \left|\frac{f_*(I)-f_A(I)}{f_*(I)}\right|$: απόλυτο σχετικό σφάλμα

$\rho = \frac{f_A(I)}{f_*(I)} = 1 + \epsilon^r$: προσεγγιστικός λόγος (approximation ratio) για προβλήματα ελαχιστοποίησης

$\rho = \frac{f_*(I)}{f_I(I)} = \frac{1}{1-\epsilon^r}$: προσεγγιστικός λόγος (approximation ratio) για προβλήματα μεγιστοποίησης

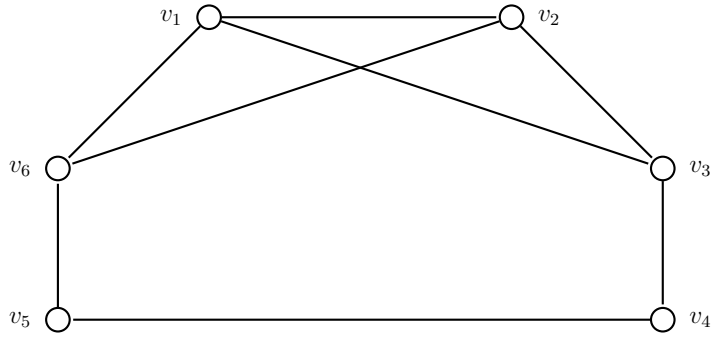
$\epsilon^d = \frac{|f_{worst}(I)-f_A(I)|}{|f_{worst}(I)-f_*(I)|} = 1 + \epsilon^r$: διαφορικός λόγος (differential ratio)

9.1 Αξιολόγηση 2 αλγορίθμων ελάχιστης κομβικής επικάλυψης

Ας προσπαθήσουμε να κάνουμε αξιολόγηση 2 ευριστικών αλγορίθμων που λύνουν το πρόβλημα της ελάχιστης κομβικής επικάλυψης (vertex cover). Στο πρόβλημα αυτό έχουμε έναν γράφο G(V,E) και ψάχνουμε σύνολο $V' \subseteq V$ τέτοιο ώστε $\forall [v_i, v_j] \in E$ να ισχύει $v_i \in V'$ ή $v_j \in V'$. Επιπλέον, θέλουμε η ποσότητα $|V'|$ (αριθμός των κόμβων) να ελαχιστοποιηθεί.

Παράδειγμα 1

Ας πάρουμε τον ακόλουθο γράφο σαν παράδειγμα:

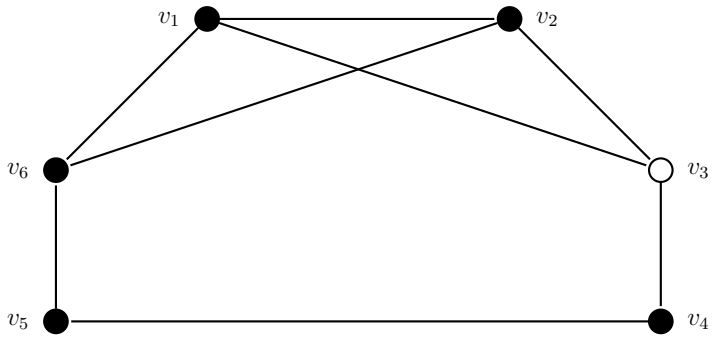


$|V| = 6, |E| = 8$

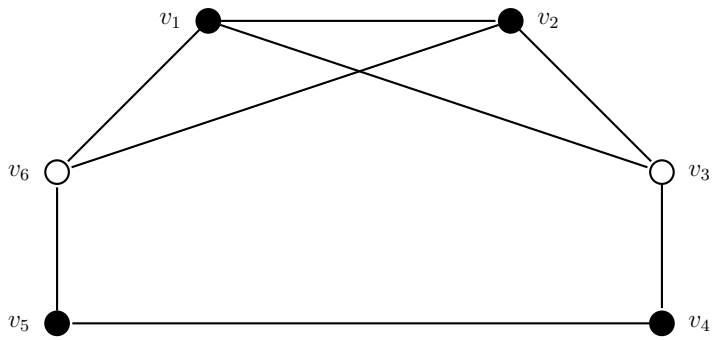
Έστω ο παρακάτω ευριστικός αλγόριθμος:

Ευριστικός αλγόριθμος 1

Input : Γράφος $G(V, E)$
Output : Κάλυψη $V' \subseteq V$
$V' \leftarrow \emptyset$
while $E \neq \emptyset$ **do**
  Επέλεξε $v \in V$ με μέγιστο βαθμό
  $V' \leftarrow V' \cup \{v\}$
  Διέγραψε όλες τις προσκείμενες στον κόμβο v ακμές από το σύνολο E
end while



$|V'| = \{v_1, v_2, v_4, v_5, v_6\}$, $|V^*| = 5$

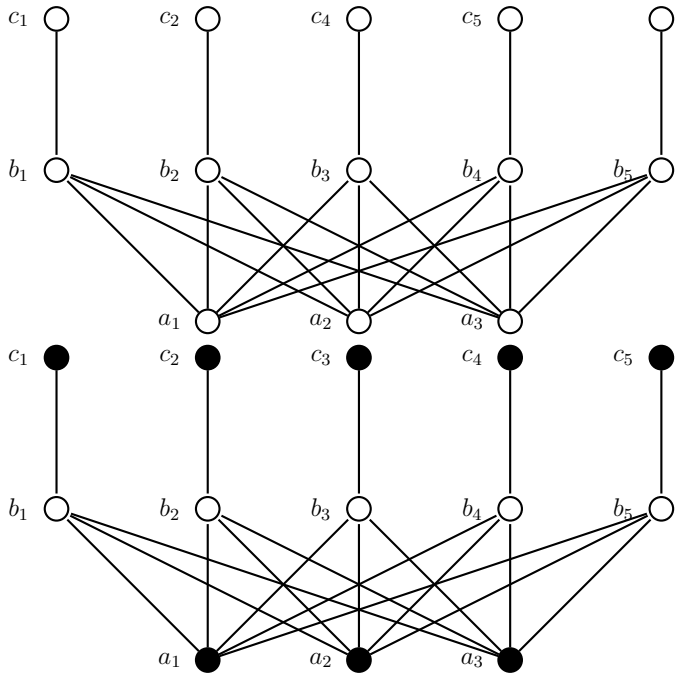


$|V^*| = \{v_1, v_2, v_4, v_5\}$, $|V^*| = 4$, βέλτιστη λύση

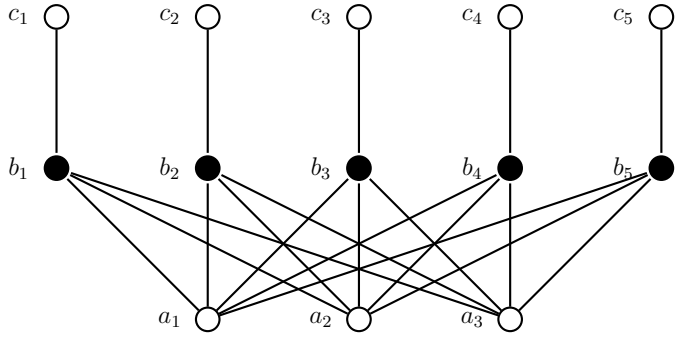
Στα δύο παραπάνω σχήματα βλέπουμε το αποτέλεσμα που δίνει ο ευρεστικός αλγόριθμος σε σχέση με το βέλτιστο. Παρατηρούμε ότι οι λύσεις δεν διαφέρουν και τόσο.

Παράδειγμα 2

Ας θεωρήσουμε τώρα ένα άλλο παράδειγμα γράφου, ο οποίος θα αποτελείται από 3 ομάδες κόμβων. Θα υπάρχουν $n+2$ κόμβοι τύπου c, $n+2$ κόμβοι τύπου b και n κόμβοι τύπου a. Κάθε κόμβος c_i συνδέεται μόνο με τον αντίστοιχο b_i. Κάθε κόμβος b_i είναι συνδεδεμένος με όλους του κόμβους της ομάδας a. Σχηματικά δηλαδή, για $n = 3$ ο γράφος θα έχει την ακόλουθη μορφή:



Η λύση του ευρεστικού αλγορίθμου 1



Η βέλτιστη λύση

Παρατηρούμε ότι για μια αυθαίρετη τιμή του n θα είναι:

- $d(a_i) = 5, i = 1, 2, ..., n$
- $d(b_i) = 4, i = 1, 2, ..., n$
- $d(c_i) = 2, i = 1, 2, ..., n$

Από τα παραπάνω προκύπτει ότι γενικά η βέλτιστη λύση θα δίνει $|V*| = 5(n + 2)$, ενώ ο ευριστικός αλγόριθμος 1 θα δίνει $|V| = 8(n+n+2)$. Επομένως, ο προσεγγιστικός

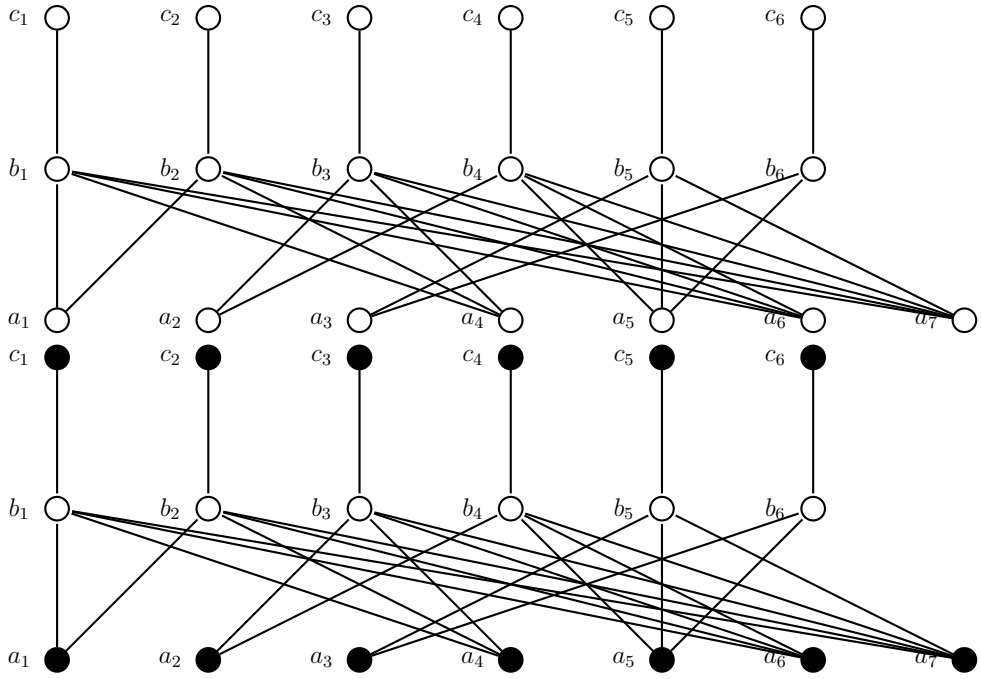
λόγος θα είναι $\rho_1 = \frac{H_1}{Opt} = \frac{2n+2}{n+2} = 2$, το οποίο μεταφράζεται σε σφάλμα $\epsilon^r = 100\%$. Όμως, το σφάλμα μπορεί να γίνει ακόμα μεγαλύτερο!

Παράδειγμα 3

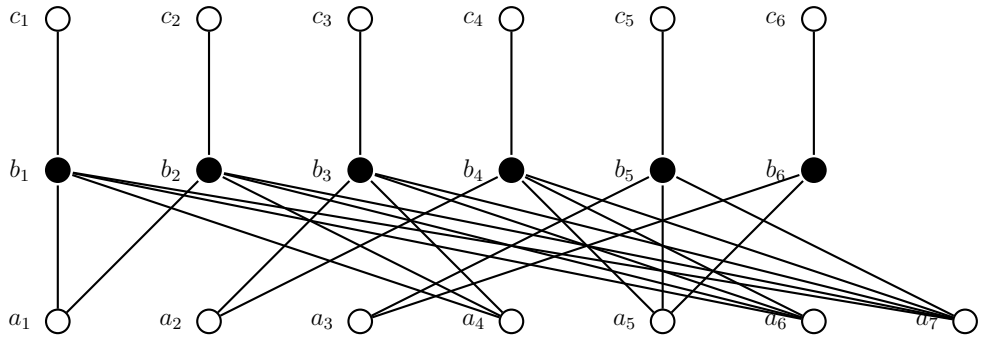
Ας επεκτείνουμε λίγο το παραπάνω παράδειγμα. Στην ομάδα a αντί για $n+2$ κόμβους προσθέτουμε έναν κόμβο για κάθε σύνολο σε κάθε διαμέριση. Αυτό σημαίνει ότι για $n = 4$ θα έχουμε:

- 3 ζευγάρια
- 2 τριάδες
- 1 τετράδα
- 1 πεντάδα.

Σε αυτή την περίπτωση, ο γράφος θα φαίνεται σχηματικά ως εξής:



Η λύση του ευρεστικού αλγορίθμου 1



Η βέλτιστη λύση

Με την ίδια λογική που αναλύσαμε τους βαθμούς των κόμβων στο προηγούμενο παράδειγμα, προκύπτει ότι η βέλτιστη λύση θα δίνει $|V^*| = 6(n+2)$, ενώ ο ευριστικός αλγόριθμος 1 θα δίνει $|V_1| = 10(n+n+2)$.

Ας εξηγήσουμε όμως γιατί συνέβη αυτή η αύξηση σφάλματος. Ο τελευταίος κόμβος της ομάδας a πάντοτε θα έχει τον μεγαλύτερο βαθμό (εκ κατασκευής του γράφου). Αυτό σημαίνει ότι ο αλγόριθμος θα επιλέγει συνολικά τους κόμβους με τους μεγαλύτερους βαθμούς από τις ομάδες a και c. Όπως φαίνεται όμως και από το σχήμα της βέλτιστης λύσης, θα έπρεπε να επιλέγονται μόνο οι κόμβοι της ομάδας b.

Αναλύοντας περαιτέρω, θα είναι:

- Λύση ευριστικού 1: $|\{\text{κόμβοι}-a\}| + n = L(n) + n$,

$$L(n) = \sum_{j=2}^{n} \left\lfloor \frac{n}{j} \right\rfloor$$

- Βέλτιστη λύση: $|\{\text{κόμβοι}-b\}| = n$
- Προσεγγιστικός λόγος: $\rho = \frac{n+L(n)}{n}$
- Απόλυτο σχετικό σφάλμα: $\epsilon^r = \frac{L(n)}{n}$

Ας δούμε στον ακόλουθο πίνακα πόσο γρήγορα μπορεί να αυξηθεί το σφάλμα ποσοστιαία, αν αυξάνουμε την τιμή του n λογαριθμικά:

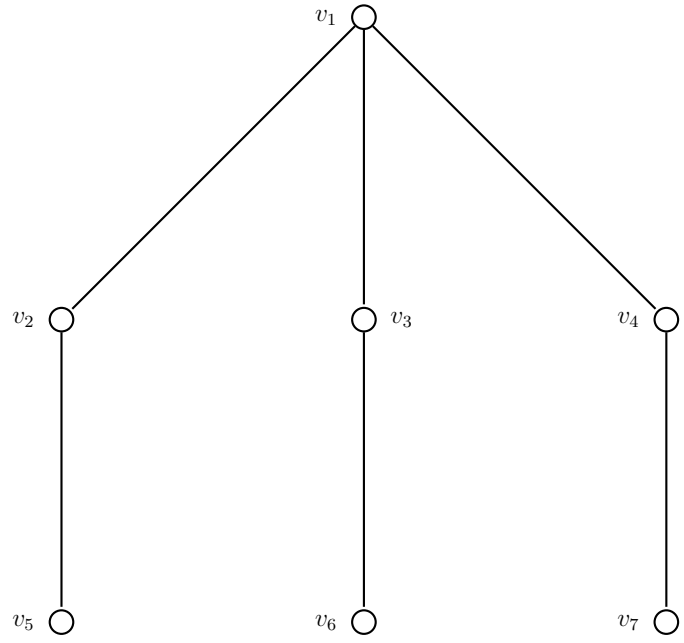
V	6	10	30	100	1000
(%) $\frac{L(n)}{n}$	117	160	267	380	600

Ας μελετήσουμε τώρα έναν δεύτερο ευριστικό αλγόριθμο, όπως αυτός φαίνεται παρακάτω:

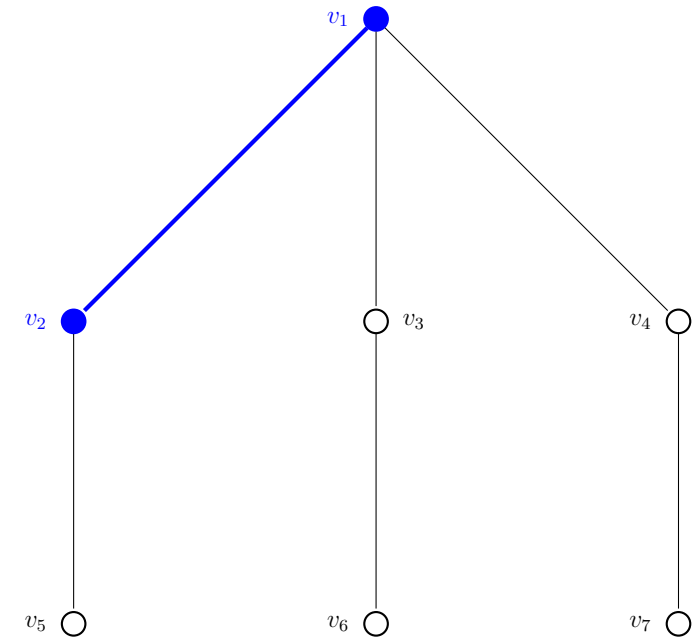
Ευριστικός αλγόριθμος 2

Input : Γράφος $G(V, E)$
Output : Κάλυψη $V' \subseteq V$
$V' \leftarrow \emptyset$
while $E \neq \emptyset$ **do**
  Επέλεξε μια ακμή $[v, u] \in E$ τυχαία
  $V' \leftarrow V' \cup \{v, u\}$
  Διέγραψε τούς κόμβους v, u από τον γράφο G
end while

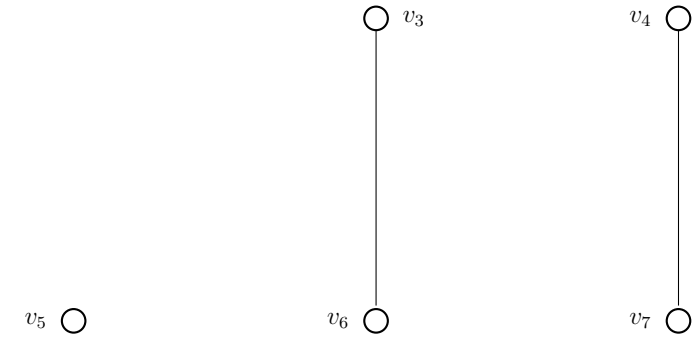
Ας δοκιμάσουμε να τρέξουμε τον ευριστικό αλγόριθμο 2 στον παρακάτω γράφο:



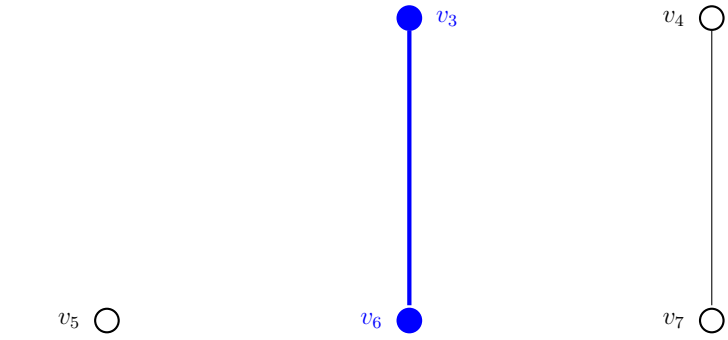
$V' = \emptyset$



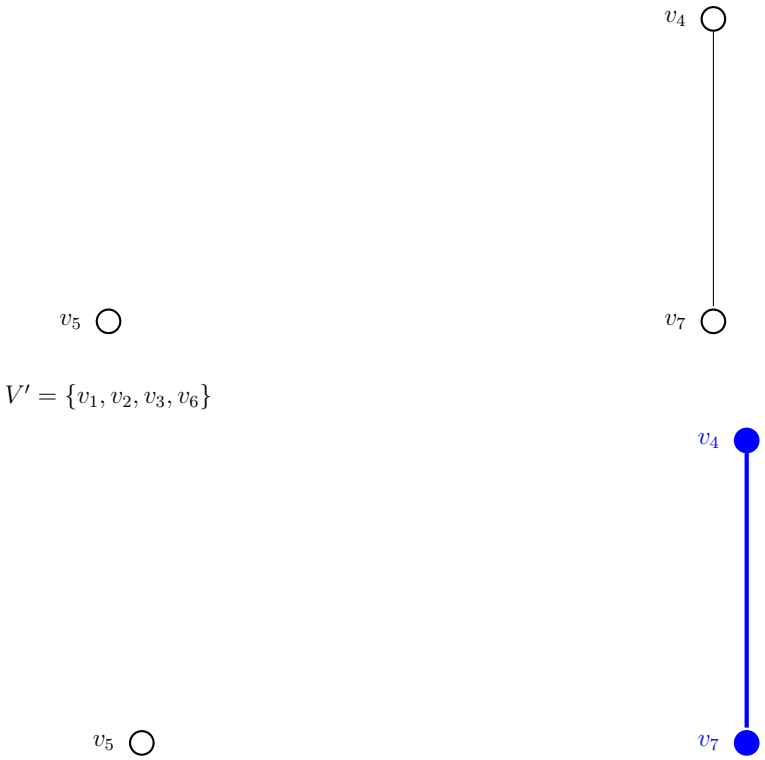
$V' = \{v_1, v_2\}$



$V' = \{v_1, v_2\}$



$V' = \{v_1, v_2, v_3, v_6\}$



Εξετάζοντας προσεκτικά την παραπάνω εκτέλεση, μπορούμε να εξάγουμε κάποια συμπεράσματα για τον ευριστικό αλγόριθμο 2:

- Κάθε επικάλυψη έχει τουλάχιστον έναν κόμβο από κάθε πλευρά μιας οποιασδήποτε ακμής (άρα ο αλγόριθμος αυτός σε κάποιες ειδικές περιπτώσεις γράφων θα μπορούσε ακόμα και να επιστρέφει το *ταίριασμα* ενός διμερούς γράφου!)

- Η λύση που επιστρέφεται από τον ευριστικό αλγόριθμο 2 είναι το *πολύ* δύο φορές μεγαλύτερη της βέλτιστης.

  Έστω V_2 η κομβική επικάλυψη του ευριστικού 2 και V^* η βέλτιστη (ελάχιστη) κομβική επικάλυψη. Θα ισχύει:

$$V^* \geq \frac{1}{2} V_2$$

  Επιπλέον, ο προσεγγιστικός λόγος θα είναι

$$\rho = \frac{V_2}{V^*} = 2.$$

Υπάρχει περίπτωση να βρούμε στιγμιότυπο γράφου τέτοιο ώστε η λύση να είναι 2 φορές μεγαλύτερη;

9.2 Weighted Vertex Cover μέσω Γραμμικού Προγραμματισμού

Έχοντας εξοικειωθεί με τις μετρικές αξιολόγησης ευριστικών αλγορίθμων, ας μελετήσουμε τώρα το πρόβλημα της *Ελάχιστης Βεβαρημένης Κομβικής Επικάλυψης*. Το γραμμικό πρόγραμμα που επιλύει το πρόβλημα για τον δεδομένο γράφο $G(V, E, w)$ είναι:

$$minw = \sum_{i=1}^{n} w_i x_i$$

$$s.t. x_i + x_j \geq 1, \forall [v_i, v_j] \in E x_i \in \{0, 1\}, \forall v_i \in V.$$

Ας προσπαθήσουμε να χαλαρώσουμε το πρόβλημα παίρνοντας μια ακέραια λύση με τη βοήθεια μιας πραγματικής, μέσω του παρακάτω ευριστικού αλγορίθμου:

Ευριστικός αλγόριθμος 3

Input : Βεβαρημένος Γράφος $G(V, E, w)$
Output : Κάλυψη $V' \subseteq V$
$V' \leftarrow \emptyset$
Βρες τη βέλτιστη συνεχή λύση (έστω x^*) του αντίστοιχου γραμμικού προβλήματος
for $i = 1$ to n **do**
  if $x_i^* \geq \frac{1}{2}$ **then**
    $V' \leftarrow V' \cup \{i\}$
  end if
end for
return V'

Αν x^* η βέλτιστη συνεχής λύση του προβλήματος, τότε θα ισχύει

$$\sum_{i=1}^{n} w_i x_i^* \geq \sum_{i=1|x_i \geq \frac{1}{2}}^{n} w_i x_i^* \geq \sum_{i=1|x_i \geq \frac{1}{2}}^{n} w_i \frac{1}{2} = \frac{1}{2} \sum_{i=1|x_i \geq \frac{1}{2}}^{n} w_i.$$

Ο τελευταίος όρος που υπολογίστηκε είναι και το κόστος της επικάλυψης που επιστρέφει ο προσεγγιστικός αλγόριθμος που παρουσιάσαμε. Γνωρίζουμε εξάλλου ότι σε προβλήματα ελαχιστοποίησης η βέλτιστη συνεχής λύση είναι πάντοτε μικρότερη από τη βέλτιστη διακριτή και επομένως θα είναι

$$W(V') \leq \frac{1}{2} \sum_{i=1}^{n} w_i x_i^* \leq \frac{1}{2} W(V^*).$$

Είναι εμφανές ότι ο προσεγγιστικός λόγος είναι

$$\rho = \frac{W(V')}{W(V^*)} = 2.$$

Σε αυτό το σημείο αξίζει να αναφέρουμε ότι αν ο γράφος που εξετάζουμε έχει συγκεκριμένη δομή, τότε η ελάχιστη κομβική επικάλυψη είναι δυνατό να βρεθεί και σε πολυωνυμικό χρόνο. Μερικές κατηγορίες γράφων που παρουσιάζουν τέτοια ειδική δομή είναι:

- Δέντρα
- Δάση
- Χορδικοί γράφοι
- Διμερείς γράφοι
- Interval γράφοι.

Βιβλιογραφία

[1] E. AARTS and J. KORST, Simulated Annealing and Boltzmann Machines, John Wiley & Sons, 1989.

[2] E. AARTS and J. K. LENSTRA, LOCAL SEARCH IN COMBINATORIAL OPTIMIZATION, John Wiley & Sons, 1997.

[3] G. AUSIELLO, P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI-SPACCAMELA, and M. PROTASI, Complexity and Approximation, Springer-Verlag, Heidelberg, 1999.

[4] C. BERGE, Graphs, Gauthier-Villars, 1983. (English and French versions).

[5] C. BERGE, Graphs and Hypergraphs North Holland, Amsterdam, 1973, (English and French versions).

[6] TH. H. CORMEN, CH. E. LEISERSON, R.L. RIVEST and C.STEIN Introduction to algorithms, 1st edition: MIT-PRESS, 1990, 2nd edition: MIT Press, 2001 and 3rd edition: MIT Press, 2009.

[7] S. EVEN, Graph Algorithms, Computer Science Press, Potomac, New York, 1979.

[8] M.R. GAREY & D.S. JOHNSON, Computers and Intractability: A guide to the theory of NP-completeness, W.H. Freeman and company, 1979.

[9] R. GARFINKEL and G. NEMHAUSER, Integer programming, New York, John Wiley, 1972.

[10] A.M. GIBBONS, Algorithmic Graph Theory, Cambridge University Press, 1985.

[11] GODRAN et M. MINOUX, Graphes et algorithmes, Eurolles, 1979.

[12] F.S. HILLIER and G.J LIEBERMAN, Introduction to Mathematical Programming, 6th ed., McGraw-Hill, New York.

[13] F.S. HILLIER and G.J LIEBERMAN, Introduction to Operations Research, Holden Day, Inc., San Francisco, 1980.

[14] H.H. HOOS and T.STUTZLE, Stochastic Local Search: Foundations and Applications, Elsevier, 2005.

[15] E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, and D.B. SHMOYS, The Travelling Salesman Computational Solutions for TSP Applications, Willey, Chichester, 1985.

[16] L. LOVÁSZ, Combinatorial Problems and Exercises, 2nd Ed., Budapest, 1993.

[17] R.K. MARTIN, Large scale linear and integer optimization, Kluwer Academic Publishers, 1999.

[18] C. PAPADIMITRIOU, K. STEIGLITZ, Combinatorial Optimization, Prentice-Hall, 1982.

[19] C. PRINS, Algorithmes de graphes, Eurolles, 1994.

[20] M. SAKAROVITCH, Optimisation combinatoire, (2 tomes), Hermann, 1984.

[21] M. SYSLO, N. DEO, J.S. KOWALIK, Discrete Optimization Algorithms with Pascal Programs, Prentice-Hall, 1983.

[22] V. V. VAZIRANI. Approximation Algorithms. Springer-Verlag, 2001.