Γραμμική και Συνδυαστική Βελτιστοποίηση

Δρομολόγηση Ελικοπτέρων σε Παράκτιες Περιοχές

Ντάγκας Αλέξανδρος , 1083874

Περιεχόμενα

1 Εισαγωγή

Η παρούσα εργασία ασχολείται με το πρόβλημα καθορισμού προγράμματος πτήσεων για ελικόπτερα προς υπεράκτιες πλατφόρμες για την ανταλλαγή πληρώματος που εργάζεται σε αυτές τις πλατφόρμες. Το μοντέλο επιλύεται με χρήση ενός προβλήματος γραμμικού προγραμματισμού (LO-Model) με την τεχνική παραγωγής στηλών (Column Generation). Επειδή η τελική λύση πρέπει να έχει ακέραιες τιμές, έχουμε επιλέξει μία διαδικασία στρογγυλοποίησης για την απόκτηση ακέραιας λύσης.

2 Προβλήματα Δρομολόγησης Οχημάτων (VRP)

Το πρόβλημα δρομολόγησης ελικοπτέρων όπως και άλλα προβλήματα που σχετίζονται με τον προγραμματισμό φορτίων σε οχήματα και τη δρομολόγηση αυτών αναφέρονται ως προβλήματα δρομολόγησης οχημάτων (Vehicle Routing Problems - VRPs). Συγκεκριμένα, ένα VRP χαρακτηρίζεται από τα παρακάτω:

- Ένας αριθμός αποθηκών με γνωστές θέσεις.
- Ένας αριθμός πελατών με γνωστές θέσεις.
- Μια δεδομένη ζήτηση προϊόντων.
- Ένας αριθμός οχημάτων με δεδομένη χωρητικότητα.
- Περιορισμοί χρόνου/πόρων οδήγησης.
- Παραδόσεις που γίνονται εντός συγκεκριμένων χρονικών παραθύρων.

Στην παρούσα μελέτη, αυτά τα χαρακτηριστικά διαμορφώνονται ως εξής:

- Υπάρχει μία αποθήκη, το αεροδρόμιο.
- Οι πελάτες είναι οι πλατφόρμες.
- Η ζήτηση είναι οι απαιτούμενες ανταλλαγές πληρωμάτων.
- Τα οχήματα είναι τα ελικόπτερα.
- Ο περιορισμός χρόνου οδήγησης είναι η εμβέλεια των ελικοπτέρων, η οποία καθορίζεται από τη μέγιστη ποσότητα καυσίμου που μπορεί να μεταφερθεί.
- Τα χρονικά παράθυρα είναι οι συνηθισμένες ώρες εργασίας κατά τη διάρκεια της εβδομάδας, χωρίς τα Σαββατοκύριακα.

Τα ελικόπτερα δεν επιτρέπεται να πετούν τη νύχτα και η ποσότητα καυσίμου περιορίζει την απόσταση που μπορούν να διανύσουν. Υποθέτουμε ότι η εμβέλεια είναι αρκετά μεγάλη για να φτάσει οποιαδήποτε πλατφόρμα και να επιστρέψει στο αεροδρόμιο.

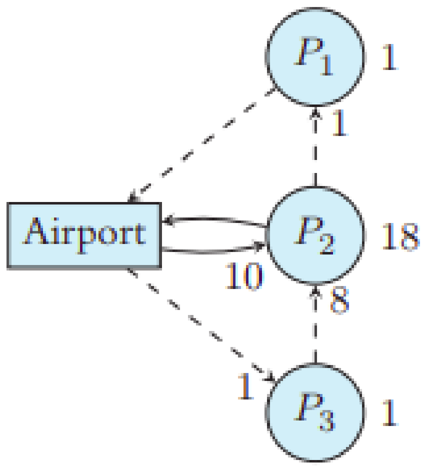
3 Διατύπωση του Προβλήματος

Έστω $P = \{P_1, \ldots, P_N\}$ το σύνολο των **πλατφορμών** και $D = \{D_1, \ldots, D_N\}$ το σύνολο των **απαιτούμενων ανταλλαγών πληρωμάτων** στις πλατφόρμες. Μπορούμε να υποθέσουμε ότι $D_i \geq 1$ για όλα τα $i = 1, \ldots, N$, διότι αν $D_i = 0$ για κάποιο i, τότε μπορούμε να διαγράψουμε την πλατφόρμα P_i από το πρόβλημα.

Οι **αποστάσεις** μεταξύ οποιουδήποτε ζεύγους πλατφορμών, καθώς και μεταξύ του αεροδρομίου και οποιασδήποτε πλατφόρμας, είναι γνωστές. Αυτές οι αποστάσεις δίνονται από $d(A, B)$ για κάθε $A, B \in P \cup \{\text{Αεροδρόμιο}\}$. Οι αποστάσεις αποτελούν και το στοιχείο του προβλήματος που θέλουμε ουσιαστικά να **ελαχιστοποιήσουμε**.

Τα ελικόπτερα έχουν **χωρητικότητα** C, η οποία είναι ο αριθμός των θέσεων (εξαιρουμένου του πιλότου) στο ελικόπτερο, και εμβέλεια R, η οποία είναι η μέγιστη απόσταση που μπορεί να διανύσει το ελικόπτερο σε μία πτήση. Υποθέτουμε ότι όλα τα ελικόπτερα έχουν την ίδια χωρητικότητα και την ίδια εμβέλεια.

Μια **πτήση** f ορίζεται ως ένα διάνυσμα $w_f = [w_{1f}, \ldots, w_{Nf}]^T$, όπου η τιμή του w_{if} υποδεικνύει τον αριθμό των ανταλλαγών πληρώματος στην πλατφόρμα P_i που θα εκτελεστεί από την πτήση f. Εάν $w_{if} = 0$, αυτό σημαίνει ότι η πτήση f δεν επισκέπτεται την πλατφόρμα P_i. Σύμφωνα με αυτόν τον ορισμό, δύο πτήσεις που επισκέπτονται το ίδιο σύνολο πλατφορμών αλλά έχουν διαφορετικά μοτίβα ανταλλαγών πληρώματος θεωρούνται διαφορετικές πτήσεις.



Σχήμα 1: Παράδειγμα Πτήσεων

Η **συνολική διανυθείσα** απόσταση d_f κατά τη διάρκεια της πτήσης f είναι το μήκος μιας συντομότερης διαδρομής πλανόδιου πωλητή (Traveling Salesman Problem) που περιλαμβάνει το αεροδρόμιο και όλες τις πλατφόρμες P_i που ικανοποιούν $w_{if} > 0$ (δηλαδή προβλέπεται κάποια ανταλλαγή πληρώματος σε αυτές).

Μια **εφικτή πτήση** είναι μια πτήση τέτοια ώστε η εμβέλεια R να μην υπερβαίνεται, και ο αριθμός των ανταλλαγών πληρώματος κατά τη διάρκεια αυτής της πτήσης να μην υπερβαίνει τη χωρητικότητα C. Για να είμαστε ακριβείς, μια πτήση f είναι εφικτή αν:

$$d_f \leq R \quad \text{και} \quad \sum_{i=1}^{N} w_{if} \leq C.$$

Έστω $F = \{f_1, \ldots, f_F\}$ το σύνολο όλων των εφικτών πτήσεων. Προφανώς υπάρχουν αρκετές εφικτές πτήσεις.

Θεώρημα 1. *Ο αριθμός F των εφικτών πτήσεων αυξάνεται εκθετικά με τον αριθμό των πλατφορμών (N).*

Απόδειξη. Ο αριθμός εφικτών πτήσεων υπολογίζεται ως εξής

Το $\binom{N}{k}$ μετράει τον τρόπο να επιλέξει κανείς k απο N πλατφόρμες.

Επίσης ισχύει:

$$w_1 + w_2 + \ldots w_k \leq C$$

Επομένως :

$$w_1 + w_2 + \ldots w_k + z = C$$

,όπου $z \geq 0$ ο αρχισιμοποίητος χώρος. Ο αριθμος των μη αρνητικών ακέραιων λύσεων της εξίσωσης είναι ίσο με τους τρόπους να μοιράσουμε
C αντικείμενα σε $k + 1$ κουτία, δηλαδή

$$\binom{C + k}{k}$$

.

Επομένως

$$|F| = \sum_{k=0}^{N} \binom{N}{k} \binom{C + k}{k}$$

Έστω $N = 2N'$ (και με $N = 2N' + 1$ ίδια απόδειξη είναι)

Στο άθροισμα κυριαρχεί ο όρος $k = \frac{N}{2}$

$$\sum_{k=0}^{N} \binom{N}{k} \binom{C+k}{k} \sim \binom{2N'}{N'} \binom{C+N'}{N'}$$

Απο την προσέγγιση Stirling ισχύει

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

$$\binom{2N'}{N'} = \frac{(2N')!}{N'!N'!} \sim \frac{2^{2N'}}{\sqrt{\pi N'}} \sim 2^N$$

και

$$\binom{C+k}{k} \sim \frac{(C+k)^k}{k!} \sim k^a \sim N^a$$

οπότε

$$\sum_{k=0}^{N} \binom{N}{k} \binom{C+k}{k} \sim 2^N N^a \sim 2^N$$

■

Ένα εφικτό πρόγραμμα πτήσεων είναι ένα πεπερασμένο σύνολο S εφικτών πτήσεων έτσι ώστε η ζήτηση για ανταλλαγές πληρωμάτων σε όλες τις πλατφόρμες να ικανοποιείται από τις πτήσεις στο S.

$$\sum_{f_j \in S} w_{ij} = D_i \quad \text{για όλα τα}\, i = 1, \ldots, N.$$

Το ελικόπτερο έχει χωρητικότητα C και εμβέλεια R. Μια πτήση f ορίζεται ως διάνυσμα $w_f = [w_{1f}, w_{2f}, \ldots, w_{Nf}]^T$, όπου η τιμή του w_{if} υποδηλώνει τον αριθμό ανταλλαγών πληρωμάτων στην πλατφόρμα P_i κατά τη διάρκεια της πτήσης f. Η συνολική απόσταση που καλύπτει η πτήση f είναι d_f, και η πτήση θεωρείται εφικτή εάν ισχύουν τα εξής:

$$d_f \leq R \quad \text{και} \quad \sum_{i=1}^{N} w_{if} \leq C.$$

Ο στόχος είναι να βρεθεί ένα εφικτό χρονοδιάγραμμα πτήσεων που να ελαχιστοποιεί τη συνολική απόσταση πτήσεων.

Με βάση τα παραπάνω σε

$$
\begin{aligned}
N &= \text{ο αριθμός των πλατφορμών·}\\
i &= \text{ο δείκτης θέσης πλατφόρμας, με } i \in \{1, \ldots, N\};\\
F &= \text{ο αριθμός των εφικτών πτήσεων ελικοπτέρων·}\\
\mathcal{F} &= \text{το σύνολο όλων των εφικτών πτήσεων ελικοπτέρων } \{f_1, \ldots, f_F\};\\
j &= \text{ο δείκτης πτήσης, με } j \in \{1, \ldots, F\};\\
D_i &= \text{ο αριθμός των απαιτούμενων αλλαγών πληρωμάτων στην πλατφόρμα } P_i;\\
w_{ij} &= \text{ο αριθμός των αλλαγών πληρώματος στην πλατφόρμα } P_i \text{ κατά τη διάρκεια της πτήσης } f_j;\\
d_j &= \text{η συνολική απόσταση που διανύθηκε κατά τη διάρκεια της πτήσης } f_j;\\
C &= \text{η χωρητικότητα, δηλαδή ο αριθμός των διαθέσιμων θέσεων, των ελικοπτέρων·}\\
R &= \text{η εμβέλεια των ελικοπτέρων.}
\end{aligned}
$$

4 Μαθηματικό Μοντέλο

Ας θεωρήσουμε το σύνολο των εφικτών πτήσεων $F = \{f_1, f_2, \ldots, f_F\}$. Η απόσταση που διανύεται από κάθε πτήση f_j είναι d_j, και ο αριθμός των ανταλλαγών πληρωμάτων που γίνονται σε κάθε πλατφόρμα P_i κατά την πτήση f_j είναι w_{ij}.

Ορίζουμε τις μεταβλητές απόφασης x_j, όπου x_j είναι ο αριθμός των φορές που εκτελείται η πτήση f_j. Το μοντέλο βελτιστοποίησης διατυπώνεται ως εξής:

$$
\begin{aligned}
&\min \sum_{j=1}^{F} d_j x_j \\
\text{υπο τους περιορισμούς: } &\sum_{j=1}^{F} w_{ij} x_j = D_i, \quad \forall i = 1, 2, \ldots, N, \\
&x_j \geq 0, \quad x_j \text{ ακέραιος } \forall j = 1, 2, \ldots, F.
\end{aligned}
\tag{FF}
$$

Το πρόβλημα περιλαμβάνει την εύρεση του ελάχιστου συνολικού μήκους των πτήσεων, ώστε να ισχύουν οι περιορισμοί χωρητικότητας και απόστασης.

Παρόλο που το πρόβλημα FF μοιάζει εύκολο να επιλυθεί υπάρχει το βασικό πρόβλημα ότι ο υπολογισμός του συνόλου των εφικτών λύσεων είναι ανέφικτος ή πολύ κοστοβόρος υπολογιστικά λόγο του *Θεωρήματος 1* , γιαυτό καταφεύγουμε στον παρακάτω αλγόριθμο.

5 Παραγωγή Στηλών (Column Generation)

Μια αποτελεσματική μέθοδος επίλυσης μοντέλων γραμμικής βελτιστοποίησης με μεγάλο αριθμό στηλών είναι η διαδικασία παραγωγής στηλών. Αυτή η μέθοδος είναι επαναληπτική και λειτουργεί παρόμοια με τον αλγόριθμο simplex. Η βασική διαφορά είναι η διαδικασία που καθορίζει τη στήλη που εισάγεται στη βάση σε κάθε επανάληψη.

Η διαδικασία αυτή χρησιμοποιείται για την επίλυση του χαλαρού/χαλαρωμένου μοντέλου (RFF), στο οποίο οι περιορισμοί των ακέραιων μεταβλητών χαλαρώνουν. Το χαλαρό μοντέλο έχει τη μορφή:

$$\begin{aligned} & \min \sum_{j=1}^{F} d_j x_j \\ \text{υπο τους περιορισμούς:} \quad & \sum_{j=1}^{F} w_{ij} x_j = D_i, \quad \forall i = 1, 2, \ldots, N, \\ & x_j \geq 0. \end{aligned} \tag{RFF}$$

Πιο συνεπτηγμένα γράφουμε το πρωτεύον και το δυικό σύστημα:

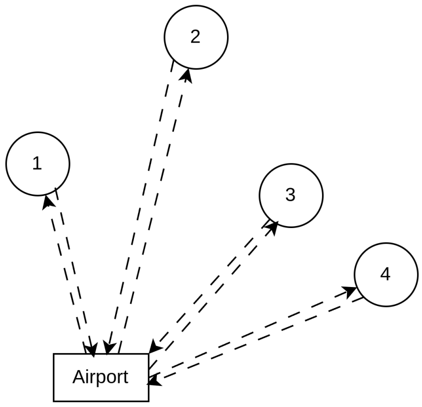
$$\begin{array}{lcl} \min d^T x & & \max D^T y \\ \text{s.t.} Wx = D & \xrightarrow{\text{Δυικό Πρόβλημα}} & \text{s.t.} W^T y \leq d \\ x_j \geq 0 & & y \in \mathbb{R}^N \end{array}$$

Στο παραπάνω πρόβλημα προφανώς δεν έχουμε τον πλήρη πίνακα W και έτσι θα χρησιμοποιήσουμε την μεθοδολογία παραγωγής στηλών για να επαυξάνουμε επαναληπτικά τον πίνακα W και το πλήθος μεταβλητών απόφασης.

5.1 Αλγόριθμος Παραγωγής Στηλών

Αρχικά πρέπει να βρούμε μία προφανή εφικτή πτήση για να αρχικοποιήσουμε τον πίνακα $W^{(1)}$.

Πράγματι, μια αρχική εφικτή βασική λύση απαιτεί N βασικές μεταβλητές· για $i = 1, \ldots, N$, παίρνουμε απλώς την πτήση f_{ji} που εκτελεί $\min\{C, D_i\}$ ανταλλαγές πληρωμάτων στην πλατφόρμα P_i και αμέσως επιστρέφει στο αεροδρόμιο. Ο πίνακας $W^{(1)}$ είναι διαγώνιος με $\min\{C, D_i\}$ (για $i = 1, \ldots, N$) ως τα διαγώνια στοιχεία και επομένως είναι αντιστρέψιμος.



Σχήμα 2: Διαδρομή για αρχικοποίηση

Στη συνέχεια υπολογίζουμε την λύση του δυικού συστήματος για αυτό το βήμα επανάληψης και χρησιμοποιούμε την εξής παρατήρηση για να εξάγουμε την στήλη που θα προσθέσουμε στον πίνακα W.

Θεώρημα 2. *Για μία μη βασική μεταβλητή x_j ο συντελεστής της στην αντικειμενική συνάρτηση είναι $d_j - \sum_{i=1}^{N} w_{ij} y_i$;*

Απόδειξη. Οι συντελεστές στην αντικειμενική συνάρτηση είναι:

$$\begin{bmatrix} 0^T & c_N^T - c_B^T B^{-1} N \end{bmatrix} = \begin{bmatrix} c_B^T - c_B^T B^{-1} B & c_N^T - c_B^T B^{-1} N \end{bmatrix} = \begin{bmatrix} c_B^T & c_N^T \end{bmatrix} - c_B^T B^{-1} \begin{bmatrix} B & N \end{bmatrix}$$

$$= \begin{bmatrix} c^T & 0^T \end{bmatrix} - c_B^T B^{-1} \begin{bmatrix} A & I_m \end{bmatrix}$$

$$= \begin{bmatrix} c^T & 0^T \end{bmatrix} - y^T \begin{bmatrix} A & I_m \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A^T \\ I_m \end{bmatrix} y$$

Θέτοντας $c \equiv d, W \equiv A$ έχουμε το ζητούμενο.

■

Στον αλγόριθμο simplex η μεταβλητή που μπαίνει στην βάση B πρέπει να έχει αρνητικό συντελεστή στην αντικειμενή συνάρτηση.

Αν μπορέσουμε να υπολογίσουμε την μεταβλητή με τον μικρότερο αντικειμενό συντελεστή έστω c^* τότε ξέρουμε ποια μεταβλητή θα μπεί στην νέα βάση.

Αν $c^* \geq 0$ σημαίνει ότι είμαστε σε βέλτιστη βασική λύση ενώ αν $c < 0$ μπορούμε να προσθέσουμε μια στήλη στον πίνακα $W^{(k)}$.

Για να βρούμε το c^* λύνουμε ενα ακόμα πρόβλημα βελτιστοποίησης.

$$\begin{aligned} c^* = \min d_{S(w)} - \sum_{i=1}^{N} y_i w_i & \\ \text{υπο τους περιορισμούς:} \quad \sum_{i=1}^{N} w_i \leq C & \\ w_i \leq D_i \quad \text{για}\, i = 1, \ldots, N & \\ d_S(w) \leq R, & \\ w_i \geq 0 \quad w_i \in Z \quad \text{για}\, i = 1, \ldots, N, & \end{aligned} \tag{CG}$$

όπου $S(w) = \{i \mid w_i > 0\} \subset P$ και $d_S(w)$ είναι η συντομότερη διαδρομή που περνά απο τις πλατφόρμες P_i στο $S(w)$, και καταλήγει στο αεροδρόμιο.

Μεταβλητές απόφασης είναι τα w_i. Ο όρος $d_{S(w)}$ καθιστά το πρόβλημα μη γραμμικό.Παρόλα αυτά μπορεί να υπολογιστεί ως η βέλτιστη τιμή ενός προβλήματος πλανόδιου πωλητή.

Για να λύσουμε το σύστημα CG μπορούμε να το σκεφτούμε σαν ένα πρόβλημα διακλαδωμένο πρόβλημα βελτιστοποίησης (bi-level optimization)

Για καθε υποσύνολο $S \subset P$ βρίσκω τη βέλτιστη λύση $c(S)$ του παρακάτω:

$$\begin{aligned} c(S) = d_s - \max \sum_{i=1}^{N} y_i w_i & \\ \text{s.t.} \sum_{i=1}^{N} w_i \leq C & \\ w_i \leq D_i & \\ w_i = 0 \ \text{για} \ P_i \notin S & \\ w_i \geq 0 \ \text{και ακέραιος για} \ i = 1, \ldots, N & \end{aligned} \tag{CGs}$$

Το σύστημα CG είναι ισοδύναμο με το :

$$\begin{aligned} c^* = \min_{S \subseteq \mathcal{P} \setminus \{\emptyset\}} c(S) & \\ \text{s.t.} \quad d_S \leq R & \end{aligned} \tag{CG^*}$$

δηλαδή κοιτάμε όλα τα δυνατά υποσύνολα των πλατφορμών και ανάμεσα σε αυτά που ικανοποιούν την σχέση $d_s \leq R$ επιστρέφουμε αυτό με το ελάχιστο $c(S)$.
Στην επόμενη υποενότητα εξετάζουμε πως θα λύσουμε το πρόβλημα CGs χωρίζοντας το σε ένα πρόβλημα πλανόδιου πωλητή και ένα πρόβλημα σακιδίου.

5.2 Πρόβλημα Πλανόδιου Πωλητή (Traveling Salesman Problem)

Ουσιαστικά θελω να υπολογίσω την ελάχιστη απόσταση για να περάσω απο όλες τις πλατφόρμες P_i με $i \in S$.
Χρησιμοποιώ τον brute force αλγόριθμο δηλαδή εξετάζω όλες τις αναδιατάξεις του S και βρίσκω την ελάχιστη. Στην περίπτωση μας πειραματικά βλέπουμε ότι το S έχει μέγεθος 4-8 στοιχεία οπότε είναι πολύ μικρή η πολυπλοκότητα του αλγορίθμου.

Έγιναν μερικές **βελτιστοποιήσεις** στον κώδικα όπως:

- Caching: Δημιουργώ ενα λεξικό {S:d_s} έτσι αν ξαναδώ το S έχω χρόνο πολυπλοκότητας $\mathcal{O}(1)$
- Επειδη έχουμε συμετρικό πρόβλημα πρέπει να εξετάσω τις μισές αναδιατάξεις. Για παράδειγμα, για N=3, η διαδρομή (0,1,2,3,0) και (0,3,2,1,0) δίνουν το ίδιο αποτέλεσμα.

Συνολική πολυπλοκότητα $\mathcal{O}(|S|!) * (T-k) + k * \mathcal{O}(1)$, οπου T, k είναι οι φορες που καλείται η τσπ_σολε και οι περιπτώσεις που βρίσκουμε την αναδιατάξη στο λεξικό αντίστοιχα.

Σε μεγάλα προβλήματα όπως αυτό με τις 51 πλατφόρμες $\frac{k}{T} = 96.8\%$

5.3 Πρόβλημα Σακιδίου (Knapsack Problem)

Επόμενως στο πρόβλημα CGs το d_S είναι ¨σταθερά' με βάση τα προηγούμενα.
Μένει να βρεθεί η βέλτιστη λύση στο.

$$\begin{aligned} \max \sum_{i \in S} y_i w_i & \\ \text{s.t.} \sum_{i \in S} w_i \leq C & \\ w_i \leq D_i & \\ w_i \geq 0 \text{ και ακέραιος για } i = 1, \ldots, N & \end{aligned} \tag{CGs}$$

Εφόσον οι μεταβλητές απόφασης είναι ακέραιες υπάγεται στο εξής πρόβλημα σακιδίου.

Μοντελο Σακιδίου 7.2.1

$$\max c_1 x_1 + \cdots + c_n x_n$$

υπό τους περιορισμούς:

$$a_1 x_1 + \cdots + a_n x_n \leq b$$

και

$$x_1, \ldots, x_n \in \{0, 1\},$$

όπου:
- n είναι ο αριθμός των αντικειμένων,
- $c_i \geq 0$ είναι η αξία του αντικειμένου i,
- $b \geq 0$ η χωρητικότητα του σακιδίου,
- $a_i \geq 0$ είναι το βάρος του αντικειμένου i ,
- $i = 1, \ldots, n$.

Θεώρημα 3. *Το μοντέλου 7.2.1 έχει τη βέλτιστη λύση:*

$$x_1^* = \cdots = x_r^* = 1, \quad x_{r+1}^* = \frac{1}{a_{r+1}} \left(b - a_1 - \cdots - a_r\right), \quad x_{r+2}^* = \cdots = x_n^* = 0,$$

όπου r τέτοιο ώστε:

$$a_1 + \cdots + a_r \leq b \quad \textit{και} \quad a_1 + \cdots + a_r + a_{r+1} > b,$$

υπό την προϋπόθεση ότι:

$$\frac{c_1}{a_1} \geq \cdots \geq \frac{c_n}{a_n}.$$

Επομένως προσπαθούμε να χωρέσουμε στο σακίδιο όσο περισσότερα μπορούμε απο τα αντικείμενα με την μεγαλύτερη αξία και οτι περισεύει απο τα υπόλοιπα.

Μπορούμε να το γενικεύσουμε στην περίπτωση του Bounded Knapsack Problem ως εξης :

5.4 Φραγμενο Πρόβλημα Σακιδίου Bounded Knapsack Problem

Ο στόχος είναι να μεγιστοποιήσουμε τη συνολική αξία:

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

υπό τον περιορισμό για το συνολικό βάρος:

$$\sum_{i=1}^{n} a_i x_i \leq b, \quad x_i \in \{0, 1, \ldots, d_i\}, \quad i = 1, \ldots, n$$

όπου:
- $c_i \geq 0$ είναι η αξία του αντικειμένου i,
- $a_i \geq 0$ είναι το βάρος κάθε αντικειμένου i,
- $b \geq 0$ είναι η χωρητικότητα του σακιδίου.

Γενικευμένη Λύση

Για να γενικεύσουμε το Θεώρημα 7.2.1, προχωράμε ως εξής:

1. Ταξινομήστε τα αντικείμενα κατά φθίνουσα σειρά του λόγου αξίας προς βάρος:

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \cdots \geq \frac{c_n}{a_n}$$

2. Επιλέξτε όσο το δυνατόν περισσότερα από τα αντικείμενα με τον υψηλότερο λόγο αξίας προς βάρος:

$$x_1^* = d_1, \quad x_2^* = d_2, \quad \ldots, \quad x_r^* = d_r$$

τέτοια ώστε:

$$a_1 x_1^* + a_2 x_2^* + \cdots + a_r x_r^* \leq b$$

3. Για το αντικείμενο $r+1$,παίρνουμε οσα περισσότερα γίνεται χωρίς να υπερβούμε τη χωρητικότητα:

$$x_{r+1}^* = \left\lfloor \frac{b - (a_1 x_1^* + \cdots + a_r x_r^*)}{a_{r+1}} \right\rfloor$$

4. Για τα υπόλοιπα αντικείμενα $r+2 \ldots n$, δεν επιλέγουμε κανένα, οπότε:

$$x_{r+2}^* = \cdots = x_n^* = 0$$

Έτσι, η γενικευμένη βέλτιστη λύση για το πρόβλημα του σακιδίου είναι:

$$x_1^* = x_2^* = \cdots = x_r^* = d_r, \quad x_{r+1}^* = \left\lfloor \frac{b - (a_1 x_1^* + \cdots + a_r x_r^*)}{a_{r+1}} \right\rfloor, \quad x_{r+2}^* = \cdots = x_n^* = 0$$

Στο δικό μας πρόβλημα έχουμε $d_i = D_i, a_i = 1, c_i = y_i, b = C$

Οπότε η βελτιστη λύση είναι

$$x_1^* = x_2^* = \cdots = x_r^* = D_r, \quad x_{r+1}^* = C - (x_1^* + \cdots + x_r^*), \quad x_{r+2}^* = \cdots = x_n^* = 0$$

ώστε $x_{r+1}^* \geq 0$

Σημείωση: Θα μπορούσαμε να το επιλύσουμε με τη μέθοδο B&B που μάθαμε στο μαθημα αλλα αφού βρήκαμε την αναλυτική λύση δεν υπάρχει λόγος για επιπλέον πολυπλοκότητα.

5.5 Αλγόριθμος

Η παραγωγή στηλών μας λέει οτι πρεπει να επαυξάνεται σε κάθε βήμα επανάληψη ο πίνακας $W^{(k)}$ καθώς και το διάνυσμα d και έτσι να προσθέτουμε κάθε φορά μια εφικτή πτήση.

Αλγόριθμος 1 Αλγόριθμος Παραγωγής Στηλών

```
Αρχικοποίηση με τον πίνακα W^(1) = diag{min(D_i, C)}
επανάληψη
    Επίλυση του RFF μοντέλου.
    Υπολογισμός των δυϊκών τιμών y_i
    για S ⊂ P
        Επίλυση του CGs με την χρήση του πλανόδιου πωλητή και του σακιδιου.
        Έυρεση του ελάχιστου c* = c(S)
        Ευρεση του αντίστοιχου d_s (TSP) και του βελτιστου w* (Knapsack)
    τελος για
    αν c* ≤ 0 τοτε
        W^(k+1) = [W^(k)  w*]
        d^(k+1) = [d^k  d_s]
    αλλιως
        Βρίκαμε την βελτιστη λύση
    τελος αν
μεχρι βρεθεί η βέλτιστη λύση.
```

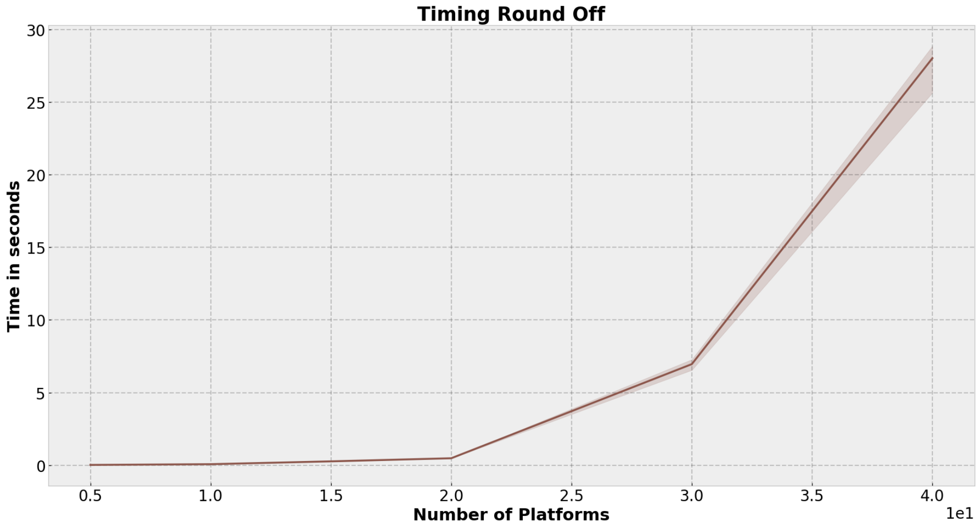
Παρακάτω φαίνεται το αποτέλεσμα για $N = 51$ και είναι ταυτόσημο με τον πίνακα 17.3 του [1].

x_j	Platforms	Crew Exchanges	Distance
1.000	2	20	72.111
1.261	9	23	127.906
1.348	19	23	151.921
1.304	21	23	159.399
1.139	40	23	83.762
1.739	41	23	88.204
1.739	43	23	137.463
0.298	12, 14, 16	7, 8, 8	182.389
0.298	13, 14, 16	7, 8, 8	182.152
0.404	14, 15, 16	8, 7, 8	182.033
0.217	44, 45	5, 18	167.805
0.435	49, 51	19, 4	165.516
0.739	12, 13, 15	8, 8, 7	175.034
1.000	22, 23, 37, 39	3, 2, 5, 13	184.442
0.783	11, 44, 45	8, 5, 10	168.380
0.435	45, 50	19, 4	167.328
0.565	49, 50, 51	15, 4, 4	168.648
0.250	28, 31, 32	4, 5, 14	141.355
0.739	46, 47, 48	14, 5, 4	156.627
0.261	46, 48	19, 4	156.280
0.261	46, 47	18, 5	150.999
0.217	11, 49	8, 15	162.138
0.478	33, 34, 36	12, 6, 5	152.756
0.217	33, 34	12, 11	151.024
1.000	18, 19, 20	10, 1, 12	162.182
1.000	17, 19	10, 13	152.101
1.000	25, 26, 27	12, 5, 6	145.194
0.272	30, 31	7, 16	134.765
1.000	9, 10, 35	1, 8, 14	138.846
0.522	33, 34, 36	7, 11, 5	152.756
0.750	29, 32	9, 14	142.441
0.550	26, 28, 29	4, 4, 15	140.102
0.200	24, 26, 28	10, 9, 4	138.105
0.800	24, 31	10, 13	136.041
0.182	8, 30	4, 19	132.858
1.000	4, 5	9, 14	110.776
0.250	4, 42	12, 11	96.862
0.828	3, 40, 42	3, 4, 16	89.111
0.909	8, 30	8, 15	132.858
0.172	3, 6, 7, 40	3, 4, 8, 8	104.628
0.828	6, 7, 40	4, 8, 11	104.145

Objective value: 3903.6217381000797

Σχήμα 3: Αποτέλεσμα

Η χρονική πολυπλοκότητα του αλγορίθμου φαίνεται παρακάτω για $N = 5, 10, 15, 20, 30, 40$



Σχήμα 4: Χρονος Εκτέλεσης Column Generation

Είναι πράγματι εκθετική κάτι που δέν μας εκπλήσσει αφού έχουμε αποδείξει οτι $F \sim 2^N$.

6 Λεξικογραφική Ταξινόμηση

Στον παραπάνω αλγόριθμο η πιο χρονοβόρα διαδικασία είναι ενδεχομένως η εκτατικη αναζήτηση σε όλα τα υποσύνολα του P για να βρούμε το ελάχιστο συντελεστή. Αυτό που μπορούμε να κάνουμε είναι να τα ταξινομήσουμε με τέτοιο τρόπο ώστε να μην χρειαστεί να εξετάσουμε κάποια απο αυτά ποτέ.

Αρχικά ταξινομούμε κατά φθίνουσα τιμή τα y_i, μετονομάζοντάς τα έτσι ώστε $y_1 \geq y_2 \geq \cdots \geq y_N$. Τα υποσύνολα S των πλατφορμών δημιουργούνται **λεξικογραφικά** στην αναζήτηση, ξεκινώντας από το $S = \{P_1\}$ μέχρι το $S = \{P_N\}$. Για παράδειγμα, αν $N = 3$ και $P = \{P_1, P_2, P_3\}$, τα υποσύνολα του P ταξινομούνται ως:

$$\{P_1\}, \{P_1, P_2\}, \{P_1, P_2, P_3\}, \{P_1, P_3\}, \{P_2\}, \{P_2, P_3\}, \{P_3\}.$$

Τα υποσύνολα που έχουν υψηλότερες δυαδικές τιμές να δημιουργούνται **πρώτα**.

Ένα υποσύνολο S_2 ονομάζεται **λεξικογραφικά υπερσύνολο** του S_1, εάν $S_1 \subseteq S_2$ και $S_1 \neq S_2$, και το S_2 εμφανίζεται αργότερα στη λεξικογραφική σειρά από το S_1. Για παράδειγμα, το $\{P_1, P_2, P_3\}$ είναι λεξικογραφικά υπερσύνολο του $\{P_1\}$, αλλά όχι του $\{P_2\}$.

Με βάση τους παραπάνω ορισμούς μπορούμε να αφαιρούμε λεξικογραφικά υπερσύνολα στις παρακάτω περιπτώσεις:

(Β1) Υπέρβαση της εμβέλειας. Εάν ένα υποσύνολο πλατφόρμας S ικανοποιεί $d_S > R$, τότε οποιοδήποτε $S' \supseteq S$ που ικανοποιεί $d_{S'} > R$ (λόγω της ανισότητας του τριγώνου) μπορεί να αποκλειστεί.

(Β2) Υπέρβαση της χωρητικότητας. Εάν το τρέχον υποσύνολο S δεν περιέχει όλες τις πλατφόρμες (δηλαδή, $S \neq P$) και αν $\sum_{i \in S} D_i \geq C$, τότε όλα τα λεξικογραφικά υπερσύνολα του S μπορούν να εξαιρεθούν.

7 Στρογγυλοποίηση Λύσης (Round-Off)

Οι τιμές του διανύσματος της δεξιάς πλευράς $\mathbf{m} = [D_1 \ldots D_N]^T$ του (RFFm) ενημερώνονται σε κάθε επανάληψη. Ας θεωρήσουμε ότι το (RFFm) αναφέρεται στο μοντέλο με τις τιμές της δεξιάς πλευράς που δίνονται από το $\mathbf{m}$.

Αλγοριτημ 2 Round-Off

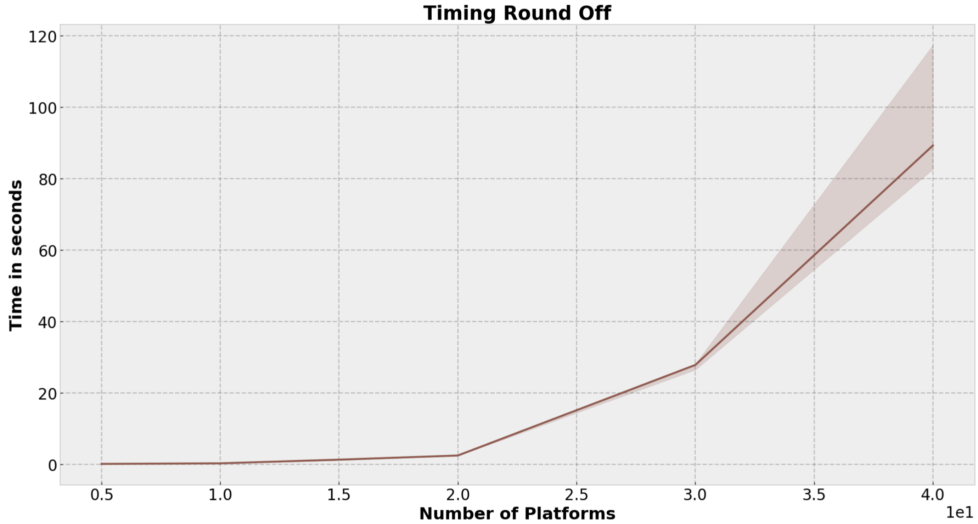
Βήμα 1: Επίλυση του μοντέλου (RFFm) με χρήση του *αλγορίθμου simplex μϵ παραγωγή στηλών*. Εάν η λύση του (RFFm) είναι ακέραια, τότε σταματα. Διαφορετικά, συνεχίστε στο Βήμα 2.

Βήμα 2: Επιλέξτε αυθαίρετα μια μεταβλητή του (RFFm) με θετική βέλτιστη τιμή, έστω $x_j > 0$.

- Αν $x_j < 1$, τότε $x_j := \lfloor x_j \rfloor = 1$· αν $x_j > 1$, τότε $x_j := \lfloor x_j \rfloor$.
- Η πτήση f_j εκτελείται είτε $\lfloor x_j \rfloor$ είτε $\lceil x_j \rceil$ φορές, και οι τιμές του D_i ενημερώνονται αφαιρώντας τον αριθμό των ανταλλαγών που πραγματοποιούνται από την πτήση f_j ,δηλαδή $D_i' = D_i - \sum w_j x_j$
- D_i' είναι όλες μηδέν, τότε τερματισε.
  Διαφορετικά, **αφαιρούμε** όλες τις στήλες από το W που δεν αντιστοιχούν πλέον σε **εφικτές** πτήσεις και τα αντίστοιχα στοιχεία απο το d ,δηλαδή τους αντικειμενικούς συντελεστές. Επίσης, **προσθέτουμε** στο W τον διαγώνιο πίνακα $A_W = diag\{\min(D_i', C)\}$και τις αντίστοιχες αποστάσεις στο d , δηλαδή απο το αεροδρόμιο στην πλατφόρμα i και πίσω.

Επιστρέψτε στο Βήμα 1.

Η χρονική πολυπλοκότητα του αλγορίθμου φαίνεται παρακάτω για $N = 5, 10, 15, 20, 30, 40$



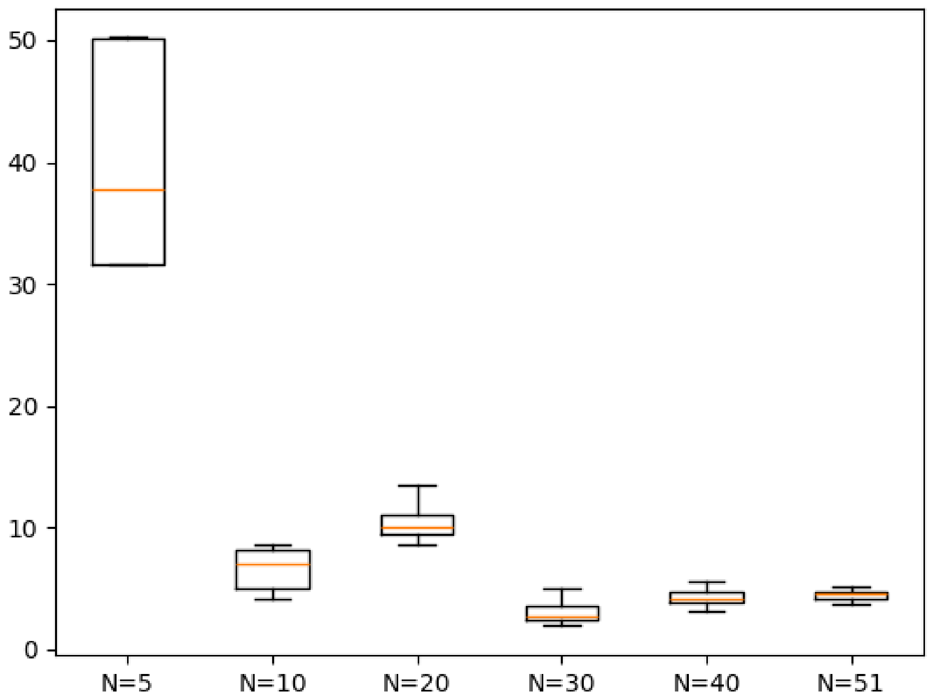
Σχήμα 5: Χρονος Εκτέλεσης Round Off

Ο παραπάνω αλγόριθμος επιστρέφει σίγουρα **ακέραιες** λύσεις, παρόλα αυτά δ**εν υπάρχει εγγύηση** οτι θα είναι και βέλτιστες. Αυτό συμβαίνει κυρίως απο την αυθαίρετη επιλογή $x_j > 0$ στο βήμα 2. Δηλαδή κάθε φορά και ανάλογα την επιλογή θα πάρουμε διαφορετική λύση, πάντα κοντά στην βέλτιστη λύση αλλά ποτέ με σιγουριά τη βέλτιστη.

Για να βρούμε **ακριβώς τη βέλτιστη λύση** πρέπει να εξετάσουμε όλους τους πιθανούς συνδιασμούς $x_j > 0$ κάτι που είναι αρκετά δύσκολο αν όχι αδύνατο αφού κάθε φορά το RFFm αλλάζει και δεν έιμαστε σίγουροι πόσοι είναι αυτόι οι συνδιασμοί.

Αλλίως μπορούμε να βρούμε μια πολύ καλή **υποβέλτιστη** λύση κάνοντας πολλές φορές την διαδικασία στρογγυλοποίησης με μια διαφορετική **γεννήτρια** τυχαίων αριθμών και να κρατήσουμε την καλύτερη.

Παρακάτω μπορούμε να δούμε πόσο θα απέχει ποσοστιαία η λύση που βρίσκουμε με την στρογγυλοποίηση απο την λύση με την παραγωγή στηλών. Προφανώς ποτέ δεν θα είναι ίδιες εκτός αν βρούμε εξαρχής ακέραια λύση. Βλέπουμε επίσης οτι ανάλογα το N δηλαδή τη δομή του προβλήματος και τις τοποθεσίες των πλατφορμών μπορεί να είναι είτε μικρή είτε μεγάλη η απόκλιση.



Σχήμα 6: Αποκλιση απο την βέλτιστη λύση του Column Generation με βαση 20 πειράματα

8 Βιβλιογραφία

Αναφορές

[1] Sierksma, G., & Zwols, Y. (2015). *Linear and Integer Optimization: Theory and Practice, Third Edition.* CRC Press, Taylor & Francis Group. ISBN: 978-1-4987-4312-9.

[2] Zeeshan-Ul-Hassan, *Introduction to Programming in Python.* (2024). Indiana State University. Available at:`http://cs.indstate.edu/~zeeshan/aman.pdf`.

[3] *Stirling's Approximation.* (2024). In Wikipedia. Available at: `https://en.wikipedia.org/wiki/Stirling%27s_approximation`.