
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ


# Προσεγγιστικοί Αλγόριθμοι Για Προβλήματα Επιλογής Πολλαπλών Υποσυνόλων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του


**ΜΕΛΙΣΣΙΝΟΥ ΝΙΚΟΛΑΟΥ**

**Επιβλέπων :** Αριστείδης Παγουρτζής
Αν. Καθηγητής Ε.Μ.Π.

ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΜΩΝ



Αθήνα, Ιούλιος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΜΩΝ

# Προσεγγιστικοί Αλγόριθμοι Για Προβλήματα Επιλογής Πολλαπλών Υποσυνόλων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**ΜΕΛΙΣΣΙΝΟΥ ΝΙΚΟΛΑΟΥ**

**Επιβλέπων :** Αριστείδης Παγουρτζής
Αν. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Ιουλίου 2017.

| | | |
|---|---|---|
| ..... | ..... | ..... |
| Αριστείδης Παγουρτζής | Αντώνιος Συμβώνης | Ευστάθιος Ζάχος |
| Αν. Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. |

Αθήνα, Ιούλιος 2017

.....

**ΜΕΛΙΣΣΙΝΟΣ ΝΙΚΟΛΑΟΣ**

Διπλωματούχος Σχολής Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών Ε.Μ.Π.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΜΩΝ

# Ευχαριστίες

Θα ήθελα να ευχαριστήσω πολύ τον επιβλέποντα της διπλωματικής εργασίας μου, κ. Άρη Παγουρτζή, Καθηγητή Ε.Μ.Π., για την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο αντικείμενο και για τη συνεχή και πολύτιμη καθοδήγηση και επίβλεψη που μου παρείχε κατά την πορεία εκπόνησης της εργασίας.

Εν συνεχεία, θα ήθελα να πω ευχαριστήσω στους καθηγητές μου και μέλη της επιτροπής, τον κ. Ευστάθιο Ζάχο, Καθηγητή Ε.Μ.Π. και τον κ. Αντώνιο Συμβώνη, Καθηγητή Ε.Μ.Π., οι οποίοι συνέβαλαν στο να αγαπήσω και να ασχοληθώ τελικά με την θεωρητική πληροφορική.

Σημαντική στήριξη είχα από την οικογένεια, τους φίλους και τους συμφοιτητές μου. Αναφέρω ενδεικτικά, τους συμφοιτητές μου, Θεόφιλο Τριομμάτη και Φοίβο Φιοραβάντε, οι οποίοι ήταν δίπλα μου καθ' όλη την διάρκεια της συγγραφή της παρούσας εργασίας και τον καλό μου φίλο, Αχίλλειο Κατσαβάκη, ο οποίος υπήρξε μεγάλο στήριγμα σε στιγμές που πραγματικά τον χρειαζόμουν.

# Περίληψη

Συνήθως, αν ένα πρόβλημα απόφασης είναι δύσκολο να λυθεί τότε το αντίστοιχο πρόβλημα βελτιστοποίησης είναι ακόμα πιο δύσκολο. Σε τέτοιες περιπτώσεις συχνά αναζητούμε αλγορίθμους οι οποίοι μπορούν να βρουν μια προσεγγιστική λύση αρκετά γρήγορα. Ένα πρόβλημα για το οποίο υπάρχουν τέτοιοι αλγόριθμοι είναι το Subset-sums ratio problem: δοσμένου ενός συνόλου αριθμών ψάχνουμε δύο υποσύνολα τέτοια ώστε ο λόγος των αθροισμάτων τους να είναι όσο το δυνατών πιο κοντά στο 1. Μελετήσαμε αρκετές παραλλαγές του προβλήματος αυτού, όπως το Factor-r SSR, στο οποίο ψάχνουμε δύο υποσύνολα τέτοια ώστε ο λόγος των αθροισμάτων τους είναι όσο το δυνατών πιο κοντά στο r, και το Two-Sets SSR όπου τα υποσύνολα χρησιμοποιούν τιμές από δύο διαφορετικά σύνολα εισόδου. Στην παρούσα εργασία παρουσιάζουμε πλήρως πολυωνυμικά σχήματα (FPTAS) για όλες τις παραλλαγές που μελετήσαμε. Οι βασικές τεχνικές μας περιλαμβάνουν δυναμικό προγραμματισμό και μεθόδους που μετατρέπουν ψευδοπολυωνυμικού χρόνου αλγορίθμους σε FPTAS. Επιπλέον θα αναφερθούμε σε κάποιες συνθήκες οι οποίες μας εξασφαλίζουν ύπαρξη FPTAS αλγορίθμων σε μια αρκετά γενική κατηγορία προβλημάτων επιλογής υποσυνόλων. Κατά την ενασχόληση μας αναπτύξαμε ένα νέο FPTAS, για το αρχικό Subset-Sums Ratio problem, το οποίο είναι γρηγορότερο από τα έως τώρα γνωστά FPTAS για το πρόβλημα.

## Λέξεις κλειδιά

Προσεγγιστικοί αλγόριθμοι, Προβλήματα βελτιστοποίησης, Εύρεση υποσυνόλων, Σύνολα ίσων αθροισμάτων.

# Abstract

Usually, if a decision problem is difficult to solve the corresponding optimiza- tion problem is even more difficult. In such cases we often search for algorithms that can find approximate solutions fast enough. A problem for which such an algorithm exists is the Subset-Sums Ratio problem (SSR): given a set of integers, the goal is to find two subsets such that the ratio of their sums is as close to 1 as possible. We discuss several variations of this problem, such as the Factor-r SSR problem, in which we are searching for two subsets such that the ratio of their sums be as close to r as possible, and the Two-Sets SSR problem where the subsets are taken from two different input sets. In this work we present fully polynomial time approximation schemes (FPTAS) for all variations considered. Our main techniques involve dynamic programming and methods which trans- form pseudo-polynomial time algorithms to FPTASs. We further discuss conditions which ensure the existence of FPTASs for a generic class of subset selection pro- blems. Along the way we obtain a new FPTAS for the original Subset-Sums Ratio problem which is faster than the best currently known FTPAS for the problem.

## Keywords

Approximation algorithms, Combinatorial optimization problems, Subset se- lection, Equal sum subset.

# Περιεχόμενα


**1 Εισαγωγή** **14**
- 1.1 Βασικές Έννοιες . . . 14
- 1.2 Προβλήματα εύρεσης υποσυνόλων . . . 16
- 1.3 Σχετική δουλειά . . . 17
  - 1.3.1 Η μέθοδος των C. Bazgan, M. Santha, Z. Tuza . . . 17
  - 1.3.2 Η μέθοδος του D. Nanongkai . . . 19

**2 Υποσύνολα ίσων αθροισμάτων** **20**
- 2.1 Ψευδοπολυωνυμικός Αλγόριθμος . . . 20
- 2.2 FPTAS . . . 21
  - 2.2.1 Ο Αλγόριθμος . . . 21
  - 2.2.2 Απόδειξη Ορθότητας . . . 25

**3 Υποσύνολα λόγου r** **31**
- 3.1 Ορισμός . . . 31
- 3.2 Ψευδοπολυωνυμικός Αλγόριθμος . . . 31
- 3.3 FPTAS . . . 32
  - 3.3.1 Αλγόριθμος . . . 32
  - 3.3.2 Απόδειξη Ορθότητας . . . 35

**4 Γενίκευση αποτελεσμάτων** **44**
- 4.1 Βασικό θεώρημα . . . 44
- 4.2 Απόδειξη . . . 45

**5 Εναλλακτική Προσέγγιση** **48**
- 5.1 Η μέθοδος του μεγαλύτερου στοιχείου . . . 48
  - 5.1.1 Ψευδοπολυωνυμικός Αλγόριθμος . . . 49
- 5.2 Χρήση της μεθόδου στο SSR . . . 50
  - 5.2.1 Ψευδοπολυωνυμικός Αλγόριθμος . . . 50
  - 5.2.2 FPTAS . . . 50
  - 5.2.3 Απόδειξη Ορθότητας . . . 52
- 5.3 Alternating SSR . . . 54
  - 5.3.1 FPTAS . . . 55
  - 5.3.2 Απόδειξη Ορθότητας . . . 56

5.4 Υποσύνολα λόγου r . . . . . 63
5.4.1 FPTAS . . . . . 64
5.4.2 Απόδειξη Ορθότητας . . . . . 66
5.5 Πολυπλοκότητα . . . . . 69

**6 Γενίκευση του ESS problem** **71**
6.1 FPTAS για το TwoSets-SSR . . . . . 72
6.2 Απόδειξη ορθότητας . . . . . 76

**7 Μέθοδος διαφοράς αθροισμάτων** **81**
7.1 Αλγόριθμοι για το SSR . . . . . 81
7.1.1 Ψευδοπολυωνυμικός αλγόριθμος . . . . . 81
7.1.2 FPTAS . . . . . 83
7.1.3 Απόδειξη ορθότητας . . . . . 84
7.2 Αλγόριθμοι για το TwoSets-SSR . . . . . 89
7.2.1 Ψευδοπολυωνυμικός αλγόριθμος . . . . . 90
7.2.2 FPTAS . . . . . 93
7.2.3 Απόδειξη ορθότητας . . . . . 94
7.3 Αλγόριθμοι για το Factor-r SSR . . . . . 98
7.3.1 FPTAS . . . . . 98

**8 Ύπαρξη και κατασκευή FPTAS** **101**
8.1 Γενίκευση Προβλημάτων . . . . . 101
8.2 Ύπαρξη-Κατασκευή FPTAS . . . . . 102

**9 Επίλογος** **106**
9.1 Αποτελέσματα . . . . . 106
9.2 Μελλοντική έρευνα . . . . . 107

**Βιβλιογραφία** **109**

# Κεφάλαιο 1

# Εισαγωγή

## 1.1 Βασικές Έννοιες

Όταν αναπτύσσουμε έναν αλγόριθμο για να λύσουμε κάποιο πρόβλημα μας ενδιαφέρει να είναι ”γρήγορος”. Αλγόριθμοι οι οποίοι τρέχουν ντετερμινιστικά σε πολυωνυμικό χρόνο ως προς την είσοδο του προβλήματος μπορούν να θεωρηθούν γρήγοροι αλγόριθμοι. Το πρόβλημα είναι ότι δεν μπορούμε πάντα να αναπτύξουμε τέτοιους αλγόριθμους. Υπάρχουν προβλήματα για τα οποία δεν μπορούν να αναπτυχθούν τέτοιοι αλγόριθμοι. Γενικά διαχωρίζουμε τα προβλήματα σε κλάσεις με βάση διάφορα κριτήρια ένα εκ των οποίων είναι η ταχύτητα με την οποία μπορούν να επιλυθούν. Δυο από τις βασικότερες κλάσεις προβλημάτων απόφασης είναι οι ακόλουθες:

**Ορισμός 1.1.1.** *Κλάση P ονομάζουμε την κλάση προβλημάτων απόφασης που επιλύονται σε πολυωνυμικό χρόνο από κάποιον ντετερμινιστικό αλγόριθμο.*

**Ορισμός 1.1.2.** *Κλάση NP ονομάζουμε την κλάση προβλημάτων απόφασης που επιλύονται σε πολυωνυμικό χρόνο από κάποιον μη-ντετερμινιστικό αλγόριθμο.*

Με βάση τους ορισμούς των δύο κλάσεων γίνεται προφανές ότι $P \subseteq NP$. Είναι ανοιχτό ερώτημα αν $P = NP$ αλλά η επικρατούσα άποψη αυτήν την στιγμή για την σχέση των δύο κλάσεων είναι ότι $P \neq NP$. Ιδιαίτερο ενδιαφέρον έχουν τα NP-Complete προβλήματα:

**Ορισμός 1.1.3.** *Ένα πρόβλημα απόφασης A λέμε ότι είναι NP-Complete όταν:*
*1. $A \in NP$*
*2. Κάθε $B \in NP$ μπορούμε να το αναγάγουμε στο A σε πολυωνυμικό χρόνο.*

Να παρατηρήσουμε ότι αν για κάποιο NP-Complete πρόβλημα A αποδειχθεί ότι ανήκει στην κλάση P τότε θα είχαμε αποδείξει την σχέση $P = NP$. Η δεύτερη συνθήκη του ορισμού των NP-Complete προβλημάτων από μόνη της μας δίνει τον ορισμό των NP-Hard προβλημάτων, δηλαδή:

**Ορισμός 1.1.4.** *Ένα πρόβλημα απόφασης A λέμε ότι είναι NP-hard όταν κάθε $B \in NP$ μπορούμε να το αναγάγουμε στο A σε πολυωνυμικό χρόνο.*

Γενικά για να δείξουμε ότι κάποιο πρόβλημα απόφασης είναι στην κλάση NP μπορούμε να υποθέσουμε μια λύση του και να δείξουμε ότι αυτή επαληθεύετε σε πολυωνυμικό χρόνο, ενώ για δείξουμε ότι είναι NP-Hard μπορούμε να αναγάγουμε πολυωνυμικά σε αυτό ένα NP-Complete πρόβλημα. Για μία γενική γνώση στους αλγορίθμους, σε σχετικά προβλήματα, τις έννοιες που αναφέραμε καθώς και το πως σχετίζονται μεταξύ τους μπορεί κάποιος να ανατρέξει στο [10].

Πολλές φορές δεν μας ενδιαφέρει απλά να υπολογίσουμε ή να αποφανθούμε για κάτι αλλά να βελτιστοποιήσουμε κάποιες μεταβλητές ενός προβλήματος. Για παράδειγμα θα μπορούσαμε να θέλουμε να ελαχιστοποιήσουμε το κόστος παραγωγής ενός προϊόντος ή να μεγιστοποιήσουμε την απόδοση μιας μηχανής. Ακριβέστερα τα προβλήματα που θα ασχοληθούμε είναι προβλήματα συνδυαστικής βελτιστοποίησης (Combinatorial optimization problems).

**Ορισμός 1.1.5.** *Ένα πρόβλημα συνδυαστικής βελτιστοποίησης A ορίζεται ως μια τετράδα (I,f,m,g), όπου:*
• *I είναι ένα σύνολο των στιγμιοτύπων*
• *Αν $x \in I$, τότε f(x) είναι το σύνολο των πιθανών λύσεων για το στιγμιότυπο αυτό*
• *Αν $x \in I$ και $y \in f(x)$, τότε το m(x,y) είναι η μέτρηση που αποδίδετε στο y για το στιγμιότυπο x*
•*Η g είναι συνάρτηση μεγίστου η ελαχίστου*
*και ο σκοπός μας είναι για κάποιο στιγμιότυπο $x \in I$ να βρούμε το $y \in f(x)$ για το οποίο ισχύει ότι $m(x, y) = g(m(x, y')|y' \in f(x))$.*

Στα προβλήματα βελτιστοποίησης ενδιαφέρον έχει η κλάση NPO η οποία ορίζεται ως εξής:

**Ορισμός 1.1.6.** *Ένα πρόβλημα A ανήκει στην κλάση NPO αν είναι προβλήματα συνδυαστικής βελτιστοποίησης και επιπλέον:*
• *Το σύνολο I των στιγμιοτύπων είναι αναγνωρίσιμο σε πολυωνυμικό χρόνο.*
• *Υπάρχει πολυώνυμο q τέτοιο ώστε, δοθέντος ενός στιγμιότυπου $x \in I$, για κάθε πιθανή λύση $y \in f(x)$ ισχύει $|y| \leq q(|x|)$ και αντίστροφα για κάθε y τέτοιο ώστε $|y| \leq q(|x|)$ μπορούμε να αποφανθούμε σε πολυωνυμικό χρόνο αν $y \in f(x)$.*
• *Η συνάρτηση m(x,y) είναι υπολογίσιμη σε πολυωνιμικό χρόνο.*

Στην εργασία μας θα αναπτύξουμε προσεγγιστικούς αλγορίθμους. Δύο κλάσεις που σχετίζονται με τα προβλήματα μας είναι οι PTAS και FPTAS.

**Ορισμός 1.1.7** (PTAS (polynomial time aprpoximation scheme))**.** *Ένα πρόβλημα $A \in NPO$ ανήκει στην κλάση PTAS αν μπορεί να αναπτυχθεί αλγόριθμος που δέχεται σαν είσοδο στιγμιότυπα του A και μία παράμετρο $\varepsilon$ και επιστρέφει λύση του A παράγοντα $1 + \varepsilon$ από την βέλτιστη (ή $1 - \varepsilon$ αν είναι πρόβλημα μεγιστοποίησης) και επιπλέον για οποιοδήποτε σταθερό $\varepsilon$ είναι πολυωνυμικά φραγμένος ως προς την είσοδο.*

**Ορισμός 1.1.8** (FPTAS (fully polynomial time aprpoximation scheme))**.** *Ένα πρόβλημα $A \in NPO$ ανήκει στην κλάση FPTAS αν μπορεί να αναπτυχθεί αλγόριθμος*

*που δέχεται σαν είσοδο στιγμιότυπα του A και μία παράμετρο $\varepsilon > 0$ και επιστρέφει λύση του A παράγοντα $1+\varepsilon$ από την βέλτιστη (ή $1-\varepsilon$ αν είναι πρόβλημα μεγιστοποίησης) και επιπλέον είναι πολυωνυμικά φραγμένος ως προς την είσοδο και την τιμή $\frac{1}{\varepsilon}$.*

Θα μπορούσε κάποιος να ανατρέξει στα [1] και [21] για να μελετήσει πιο αναλυτικά τεχνικές προσεγγιστικών αλγορίθμων σε NP-Hard προβλήματα, καθώς και να δει αναγωγές και άλλα θεωρητικά αποτελέσματα σχετικά με αυτά.

## 1.2 Προβλήματα εύρεσης υποσυνόλων

Τα προβλήματα εύρεσης υποσυνόλων αποτελούν μια αρκετά γενική κατηγορία προβλημάτων στα οποία δοθέντος ενός συνόλου ακεραίων αναζητούμε ένα η περισσότερα υποσύνολα αυτού τα οποία πληρούν κάποιες επιθυμητές προϋποθέσεις. Μια από τις πιο γνωστές κλάσεις τέτοιων προβλημάτων είναι τα προβλήματα Knapsack τα οποία αναλύονται αρκετά στα [15] και [17] και φαίνεται να υπάρχει ακόμα ερευνητικό ενδιαφέρων με εργασίες να εμφανίζονται και πρόσφατα όπως η [4], [11] και [12] . Άλλα προβλήματα τα οποία σχετίζονται είναι αυτά που ζητάνε διαμερίσεις του συνόλου που δίνεται αρχικά (partition problems), και τα οποία θα μπορούσαμε να φανταστούμε ως προβλήματα ενός υποσυνόλου $S$ το οποίο ορίζει την διαμέριση (πληροφορίες σχετικά με προβλήματα διαμερίσεων μπορεί κάποιος να αναζητήσει στο [16]) και τα προβλήματα Bin Packing (για τα οποία μπορεί κάποιος να ανατρέξει στα [14] και [13]). Στην παρούσα εργασία δεν θα ασχοληθούμε με προβλήματα εύρεσης ενός συνόλου ενώ για τα περισσότερα από τα προβλήματα με τα οποία θα καταπιαστούμε ισχύει ότι είναι NP-hard. Το πρώτο από τα προβλήματα που θα ασχοληθούμε είναι το ESS problem (equal sum subsets problem):

**Πρόβλημα 1** (Υποσύνολα ίσων αθροισμάτων ή ESS)**.** *Δοθέντος ενός συνόλου από n θετικούς ακεραίους $A = \{a_1, a_2, ..., a_n\}$, υπάρχουν δύο μη κενά και ξένα σύνολα $S_1, S_2 \subseteq \{1, 2, ..., n\}$ τέτοια ώστε $\sum_{i \in S_1} a_i = \sum_{j \in S_2} a_j$;*

Μερικές παραλλαγές του ESS περιγράφονται παρακάτω:

**Πρόβλημα 2** (Υποσύνολα λόγου r ή Factor-r Sum Subsets problem)**.** *Δοθέντος ενός συνόλου από n θετικούς ακεραίους $A = \{a_1, a_2, ..., a_n\}$ και ενός θετικού αριθμού r, υπάρχουν δύο μη κενά και ξένα σύνολα $S_1, S_2 \subseteq \{1, 2, ..., n\}$ τέτοια ώστε $\sum_{i \in S_1} a_i = r \cdot \sum_{j \in S_2} a_j$;*

**Πρόβλημα 3** (Alternating ESS problem)**.** *Δοθέντος ενός συνόλου από n ζεύγη θετικών ακεραίων $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$, υπάρχουν δύο μη κενά και ξένα σύνολα $S_1, S_2 \subseteq \{1, 2, ..., n\}$ τέτοια ώστε $\sum_{i \in S_1} a_i + \sum_{j \in S_2} b_j = \sum_{i \in S_1} b_i + \sum_{j \in S_2} a_j$;*

**Πρόβλημα 4** (k ESS problem)**.** *Δοθέντος ενός συνόλου από n θετικούς ακεραίους $A = \{a_1, a_2, ..., a_n\}$, υπάρχουν k μη κενά και ξένα σύνολα $S_1, S_2, ...S_k \subseteq \{1, 2, ..., n\}$ τέτοια ώστε $\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i = ... = \sum_{i \in S_k} a_i$;*

**Πρόβλημα 5** (Υποσύνολα ίσων αθροισμάτων με ίσες πληθικότιτες)**.** *Δοθέντος ενός συνόλου από n θετικούς ακεραίους* $A = \{a_1, a_2, ..., a_n\}$*, υπάρχουν δύο μη κενά και ξένα σύνολα* $S_1, S_2 \subseteq \{1, 2, ..., n\}$ *τέτοια ώστε* $\sum_{i \in S_1} a_i = \sum_{j \in S_2} a_j$ *και* $|S_1| = |S_2|$*;*

Περισσότερες λεπτομέρειες σχετικά με τα παραπάνω προβλήματα, με την πολυπλοκότητα τους καθώς και το πως θα μπορούσαν να λυθούν ακριβώς μπορεί κάποιος να δει στα [7] και [6]. Η δικιά μας δουλειά αφορά κυρίως ανάπτυξη προσεγγιστικών αλγορίθμων στα αντίστοιχα προβλήματα βελτιστοποίησης. Ακριβέστερα θα λύσουμε προβλήματα όπως το Subset-sums ratio problem το οποίο εμφανίζεται στο [2] και στο [18] και αποτελεί πρόβλημα ελαχιστοποίησης. Ο ορισμός του Subset-sums ratio problem είναι ο ακόλουθος:

**Πρόβλημα 6** (Subset-sums ratio problem ή SSR)**.** *Δοθέντος ενός συνόλου από n θετικούς ακεραίους* $A = \{a_1, a_2, ..., a_n\}$*, ψάχνουμε δύο μη κενά και ξένα σύνολα* $S_1, S_2 \subseteq \{1, 2, ..., n\}$ *τέτοια ώστε* $\sum_{i \in S_1} a_i \geq \sum_{j \in S_2} a_j$ *και να ελαχιστοποιείται ο λόγος* $\sum_{i \in S_1} a_i / \sum_{j \in S_2} a_j$*.*

Στην εργασία αυτή θα αναπτύξουμε αλγόριθμους και ελαχιστοποίησης αλλά και μεγιστοποίησης για κάποια από τα παραπάνω προβλήματα. Στην συνέχεια, όπου χρειάζεται, θα υπενθυμίζουμε ή θα δίνουμε καινούριους ορισμούς .

## 1.3 Σχετική δουλειά

Για το Subset sums ratio problem(SSR) υπάρχουν δύο διαφορετικοί FPTAS αλγόριθμοι στα [2] και [18] όπου ο ένας έχει πολυπλοκότητα τουλάχιστον $O(\frac{n^5}{\varepsilon^2})$ ενώ ο δεύτερος έχει χειρότερη πολυπλοκότητα αλλά ενδιαφέρουσα μέθοδο. Επιπλέον υπάρχει προσεγγιστικός αλγόριθμος ακρίβειας 1.324 στο [22]. Μέρος της παρούσας εργασίας είναι η παρουσίαση ενός FPTAS αλγορίθμου, ο οποίος επιλύει γρηγορότερα το SSR, και στην γενίκευσή του σε διάφορα άλλα προβλήματα (παραλλαγές του ESS-SSR). Εδώ θα δούμε, όχι ιδιαίτερα αναλυτικά, τα FPTAS που υπάρχουν στα [2] και [18].

### 1.3.1 Η μέθοδος των C. Bazgan, M. Santha, Z. Tuza

Αρχίζοντας από το [2] να πούμε ότι μας περιγράφει μια διαδικασία που λύνει το SSR. Η διαδικασία αυτή είναι ψευδοπολυωνυμικού χρόνου. Ακριβέστερα, για την είσοδο $A = \{a_1, ..., a_n\}$ (θα θεωρούμε το A ταξινομημένο), κατασκευάζει δύο πίνακες $t$ και $c$ οι οποίοι έχουν ίδιες διαστάσεις $n \times B$ (όπου $B$ το άθροισμα όλων των στοιχείων $a_i$). Το σκεπτικό με το οποίο γεμίζει τους πίνακες είναι ότι αν υπάρχει σύνολο $S \subseteq \{1, ..., n\}$ με $max\{S\} = n_0$ και $\sum_{i \in S} a_i = k$ τότε να έχουμε $t(n_0, k) = 1$ και $c(n_0, k) = S$ αλλιώς τα κελιά του $t$ έχουν τιμή 0 και του $c$ είναι κενά. Τέλος για το μικρότερο $k$ τέτοιο ώστε να υπάρχουν τουλάχιστον δύο κελιά ώστε $t(i, k) = t(j, k) = 1$ ($i \neq j$) επιστρέφει τα σύνολα $c(i, k) = S_1$, $c(j, k) = S_2$ ενώ αν δεν υπάρχει τέτοιο $k$ για όλα τα ζεύγη $k_1$, $k_2$ τέτοια ώστε να

υπάρχουν $c(i,k_1)=S_1$, $c(j,k_2)=S_2$ μη κενά και ξένα επιστρέφει αυτά που δίνουν μικρότερο λόγο $\frac{k_1}{k_2}>1$. Η ορθότητα τις παραπάνω διαδικασία είναι προφανής ενώ στην χειρότερη περίπτωση θέλει χρόνο $O(n\cdot B^2)$.

Με βάση τον παραπάνω αλγόριθμο και με την συνειδητοποίηση ότι αν η βέλτιστη λύση με είσοδο το $A=\{a_1,...,a_n\}$ έχει μέγιστο στοιχείο το $m$ τότε είναι βέλτιστη και για είσοδο $A_m=\{a_1,...,a_m\}$ κατασκευάστηκε ο ακόλουθος προσεγγιστικός αλγόριθμος. Για δοθέν $\varepsilon$ και στιγμιότυπο $A_m=\{a_1,...,a_m\}$ $(m=2,...,n)$ υπολογίζει την τιμή $k(m)=\varepsilon\cdot a_m/(2\cdot m)$. Ο αλγόριθμος βρίσκει το μέγιστο $n_0\leq n$ τέτοιο ώστε $k(n_0)<1$. Έπειτα επιστρέφει μια λύση για κάθε στιγμιότυπο $A_m$.
• Αν $m\leq n_0$ τρέχει τον παραπάνω ψευδοπολυωνυμικού χρόνου αλγόριθμο για στιγμιότυπο αυτό (το $B$ είναι της τάξης του $O(n^2)$ σε αυτή την περίπτωση).
• Αν $n_0<m\leq n$ κατασκευάζει $a'_i=\lfloor a_i/k(m)\rfloor$ και το $A'_m$ που θα περιέχει όσα $a'_i\geq m/\varepsilon$ (το $A'_m$ είναι μη κενό καθώς $a'_m\geq m/\varepsilon$). Έπειτα ο αλγόριθμος αποφασίζει τι θα κάνει με βάση το πλήθος των στοιχείων του $A'_m$ ακολουθεί διαφορετική διαδικασία.

**Περίπτωση 1** (Αν το $A'_m$ έχει μόνο ένα στοιχείο)**.** *Τότε ο αλγόριθμος βρίσκει το μικρότερο $j$ ώστε $a_{j+1}+...+a_{m-1}<a_m$. Αν $j=0$ τότε η λύση που επιστρέφει είναι $S_1=\{m\}$ και $S_2=\{1,...,m-1\}$. Αλλιώς η λύση που επιστρέφει είναι $S_1=\{j,j+1,...,m-1\}$ και $S_2=\{m\}$.*

**Περίπτωση 2** (Αν $|A'_m|=t\neq 1$)**.** *Τότε ο αλγόριθμος καλεί τον προηγούμενο ψευδοπολυωνυμικό για το στιγμιότυπο $A'_m$ (για το οποίο εκτελείτε σε πολυωνυμικό χρόνο) και έπειτα εξετάζει τις ακόλουθες περιπτώσεις:*

**Υπο-περίπτωση 2.1** (Η βέλτιστη λύση του $A'_m$ είναι το 1)**.** *Τότε επιστρέφει αυτήν την λύση.*

**Υπο-περίπτωση 2.2** (Η βέλτιστη λύση του $A'_m$ είναι μεγαλύτερη του 1)**.** *Τότε το πλήθος όλων των ζευγών $P_1$, $P_2$, ξένων υποσυνόλων του $\{m-t+1,...,m\}$, με $m\in P_1$ είναι $3^{t-1}$. Για κάθε ζεύγος $P_1$, $P_2$ ορίζουμε $S'_1=P_1$ αν $\sum_{i\in P_1}a_i>\sum_{i\in P_2}a_i$ ή $S'_1=P_1$ αλλιώς. Ορίζουμε $S'_2$ το άλλο σύνολο. Για κάθε ζεύγος $S'_1$, $S'_2$ βρίσκουμε το μικρότερο $j$ ώστε:*

$$\sum_{i\in S'_2}a_i+\sum_{i=j+1}^{m-t}a_i<\sum_{i\in S'_1}a_i$$

*Αν $j=0$ θέτει $S_1=S'_1$ και $S_2=S'_2\cup\{1,...,m-t\}$. Αλλιώς, αν $m\in S'_1$ τότε $S_1=S'_2\cup\{j,...,m-t\}$ και $S_2=S'_1$. Ενώ αν $m\in S'_2$ τότε $S_1=S'_1$ και $S_2=S'_2\cup\{j+1,...,m-t\}$. Τέλος επιστρέφει τα $S_1$, $S_2$ που δίνουν τον καλύτερο λόγο.*

Να σχολιαστεί ότι όταν η βέλτιστη λύση του $A'_m$ είναι μεγαλύτερη του 1 το πλήθος $3^{t-1}$ ζευγών που εξετάζει ο αλγόριθμος αποδεικνύεται πολυωνυμικό. Για παραπάνω λεπτομέρειες αλλά και για την απόδειξη ότι ο αλγόριθμος είναι FPTAS μπορεί κάποιος να ανατρέξει στο [2].

### 1.3.2 Η μέθοδος του D. Nanongkai

Στο [18] αναπτύχθηκε ένα απλούστερο στην περιγραφή αλλά πιο αργό FPTAS για το RSS. Εκεί ορίζεται ένα πρόβλημα παρόμοιο με το ESS (και το SSR αντίστοιχα) στο οποίο θέλουμε επιπλέον συγκεκριμένα μέγιστα για τα σύνολα. Η ιδέα που παρουσιάστηκε εκεί είναι ότι λύνοντας το καινούριο πρόβλημα για όλα τα δυνατά ζεύγη μεγίστων μπορώ να βρω την βέλτιστη λύση του SSR. Ακριβέστερα για είσοδο ένα σύνολο $A = \{a_1, ..., a_n\}$ και κάθε ζεύγος μεγίστων (έστω $1 \leq p < qleqn$) λύνει ακριβώς το SSR με χρήση των $p$ και $q$ κατασκευάζοντας ένα πίνακα $T_i[x, y]$ μεγέθους $n \times (n \cdot a_q)^2$ στον οποίο $T_0[a_q, a_p] = (\{q\}, \{p\})$ και αν υπάρχουν ξένα σύνολα $S_1$, $S_2$ ($\subseteq \{1, 2, ..., q\}$) τέτοια ώστε $max\{S_1\} = q$, $max\{S_2\} = p$, $(S_1 \cup S_2) \subseteq \{1, 2, ..., i\}$ $\sum_{i \in S_1} a_i = x$ και $\sum_{i \in S_2} a_i = y$ τότε το κελί $T_i[x, y]$ είναι μη κενό (ακριβέστερα περιέχει ένα τέτοιο ζεύγος συνόλων). Τέλος από όλα τα κελιά με δείκτη $q - 1$ ($T_{q-1}[x, y]$) είναι μη κενά επιστρέφει αυτό με ελάχιστο λόγο $x/y \geq 1$. Από τις επιστροφές για όλα τα ζεύγη μεγίστων κρατά την μικρότερη τιμή η οποία αποτελεί και την λύση για το SSR.

Το FPTAS είναι παρόμοιου σκεπτικού. Αν θεωρήσουμε είσοδο $A = \{a_1, ..., a_n\}$ και $\varepsilon \in (0, 1)$ τότε για κάθε ζεύγος μεγίστων $1 \leq p < q \leq n$ έχουμε:

**Περίπτωση 1** (Αν $n \cdot a_p < a_q$)**.** *Τότε επιστρέφει τα σύνολα $S_1 = \{q\}$, $S_2 = \{1, 2, ..., p\}$ τα οποία δίνουν το βέλτιστο σε αυτήν την περίπτωση.*

**Περίπτωση 2** (Αλλιώς)**.** *Κατασκευάζει στιγμιότυπο $A' = \{a'_1, ..., a'_q\}$ (με $a'_i = \lfloor a_i/\delta \rfloor$, $\delta = \varepsilon \cdot a_p/(3 \cdot n)$) και καλεί τον προηγούμενο, ακριβή, αλγόριθμο για το $A'$.*

Τέλος από τις $n^2$ πιθανές λύσεις που προκύπτουν κρατά αυτή με τον μικρότερο λόγο με τιμές από το $A = \{a_1, ..., a_n\}$.

Για παραπάνω λεπτομέρειες αλλά και για την απόδειξη ότι ο αλγόριθμος είναι FPTAS μπορεί κάποιος να ανατρέξει στο [18].

# Κεφάλαιο 2

# Υποσύνολα ίσων αθροισμάτων

## 2.1 Ψευδοπολυωνυμικός Αλγόριθμος

Γενικά δεν θα λύνουμε το ακριβές πρόβλημα ($\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} = 1$) αλλά θα βελτιστοποιούμε τον λόγο (για ακρίβεια ο Algorithm 2.1 θα βρίσκει το μέγιστο λόγο $\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \leq 1$). Πιο συγκεκριμένα, αυτό το πρόβλημα βελτιστοποίησης αναφέρεται από πολλούς ως Subset-Sums Ratio Problem και έχουν αναπτυχθεί ψευδοπολυωνυμικοί αλγόριθμοι στα [2] και [18] με βασική διαφορά ότι εκεί ψάχνουν τον ελάχιστο λόγο $\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \geq 1$. Η ιδέα που θα χρησιμοποιήσουμε για να λύσουμε το πρόβλημα είναι παρόμοια με τις ήδη υπάρχοντες. Θα κατασκευάσουμε ένα πίνακα $F_k[x, y]$ (με $1 \leq k \leq n$ και $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a_i\}$) στα κελιά του οποίου θα κρατάμε δύο σύνολα $S_1, S_2$ (αν υπάρχουν) τέτοια ώστε $\sum_{i\in S_1} a_i = x$, $\sum_{j\in S_2} a_j = y$ και $k = max\{S_1 \cup S_2\}$. Ο τρόπος με τον οποίο γεμίζουμε τον πίνακα είναι παρόμοιος με αυτόν που βλέπουμε στο [7] για την επίλυση του Factor-r Sum Subsets problem με βασική διαφορά ότι στα κελιά του κρατάμε σύνολα και όχι μόνο true ή false. Η πολυπλοκότητα του αλγορίθμου δεν είναι καλύτερη από αυτή του [2] αλλά ο ψευδοπολυωνυμικός αλγόριθμος είναι αρκετά απλός. Για διευκόλυνση έχουμε σπάσει τον αλγόριθμο σε δύο κομμάτια:

**Sub-Algorithm 2.1** Compute table F for SSR

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$

1: **make** table $F_k[x, y]$ with size $k = n$ and $x = y = \sum_{i=1}^{n} a_i$
2: **for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a_i\}$ **do**
3: $\quad F_k[x, y] \leftarrow \vec{\emptyset}$
4: **end for**
5: **for** $i \leftarrow 1$ to $n$ **do**
6: $\quad F_i[a_i, 0] \leftarrow (\{i\}, \emptyset)$ and $F_i[0, a_i] \leftarrow (\emptyset, \{i\})$
7: **end for**

8: **for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a_i\}$ **do**
9: **for all** $0 < l < k$ **do**
10: **if** $F_l[x - a_k, y] \neq \vec{\emptyset}$ **then**
11: $(S_1, S_2) \leftarrow F_l[x - a_k, y]$
12: $F_k[x, y] \leftarrow (S_1 \cup \{k\}, S_2)$
13: **else if** $F_l[x, y - a_k] \neq \vec{\emptyset}$ **then**
14: $(S_1, S_2) \leftarrow F_l[x, y - a_k]$
15: $F_k[x, y] \leftarrow (S_1, S_2 \cup \{k\})$
16: **end if**
17: **end for**
18: **end for**
19: **return** $F$

**Algorithm 2.1** Solve SSR

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$
1: **sort** $A$
2: **run** Sub-Algorithm 2.1 with **input** $A$ and **output** $F$
3: $k_0 \leftarrow 0$, $x_0 \leftarrow 0$, $y_0 \leftarrow 0$
4: $Opt \leftarrow 0$
5: **for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{1, 2, ..., \sum_{i=1}^{n} a_i\}$ **do**
6: **if** $F_k[x, y] \neq \vec{\emptyset}$ **and** $Opt \leq \frac{x}{y} \leq 1$ **then**
7: $k_0 \leftarrow k$, $x_0 \leftarrow x$, $y_0 \leftarrow y$
8: $Opt \leftarrow \frac{x}{y}$
9: **end if**
10: **end for**
11: **return** $F_{k_0}[x_0, y_0]$ **and** $Opt$

Καλό είναι να επισημάνουμε ότι ο αλγόριθμος θα κατασκευάσει όλους τους δυνατούς συνδυασμούς αθροισμάτων (συμπεριλαμβανομένου και αυτού του βέλτιστου) και άρα θα επιστρέφει πάντα το βέλτιστο λόγο και δύο σύνολα που τον παράγουν.

## 2.2 FPTAS

### 2.2.1 Ο Αλγόριθμος

Κατά την ενασχόληση μας με το SSR καταφέραμε να αναπτύξουμε διάφορα FPTAS πέρα από τα ήδη υπάρχοντα (στα [2] και [18]). Καλό είναι να κατανοήσουμε πλήρως τον ακόλουθο FPTAS αλγόριθμο γιατί αυτός είναι η βάση για τους αλγορίθμους των υπολοίπων προβλημάτων. Εδώ θα παρουσιάσουμε την πρώτη προσπάθεια προσέγγισης που κάναμε.

Ο αλγόριθμος που ακολουθεί θα δέχονται σαν είσοδο ένα σύνολο $A = \{a_1, a_2, ..., a_n\}$ από $n$ ακεραίους και μια παράμετρο ακρίβειας $\varepsilon$ ενώ θα επιστρέφει σαν έξοδο δύο ξένα σύνολα $S_1, S_2$ και το λόγο των αθροισμάτων τους, $\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$, για τον οποίο θέλουμε να ισχύει ότι:

$$\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \in \left[(1-\varepsilon) \cdot \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}, \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}\right]$$

όπου $S_{1Opt}, S_{2Opt}$ είναι τα ξένα σύνολα που μας δίνουν τον βέλτιστο (μέγιστο στη συγκεκριμένη περίπτωση) λόγο:

$$\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} \leq 1$$

Η βασική ιδέα που θα χρησιμοποιήσουμε για την κατασκευή των FPTAS είναι να φράξουμε το συνολικό άθροισμα των στοιχείων της εισόδου από μια πολυωνυμική συνάρτηση του πλήθους των στοιχείων και του λόγου $\frac{1}{\varepsilon}$. Αυτό θα το πετύχουμε διαιρώντας με κάποια τιμή που μας βολεύει (παράμετρος μεγέθους ή scaling parameter) και στρογγυλοποιώντας κατάλληλα. Αυτή η ιδέα χρησιμοποιείται αρκετά σε προβλήματα αθροισμάτων είτε αν θέλουμε ένα άθροισμα (FPTAS για το Knapsack στο [21]) είτε για δύο αθροίσματα (FPTAS για το Subset-Sums Ratio Problem στα [2] και [18]). Αυτό θα το κάνουμε για όλα τα υποσύνολα του A της μορφής $A^m = \{a_1, a_2, ..., a_m\}$. Ο παρακάτω αλγόριθμος κατασκευάζει ένα παρόμοιο πίνακα με αυτόν του Sub-Algorithm 2.1 για τις προσαρμοσμένες τιμές ενός μόνο στιγμιότυπου $A^m$ και σε κάθε κελί εκτός των συνόλων θα κρατάει και τα πραγματικά αθροίσματα αυτών (και αν υπάρχει πάνω από ένας συνδυασμοί για κάποιο κελί κρατάει αυτόν που μεγιστοποιεί το πραγματικό άθροισμα του πρώτου συνόλου).

**Sub-Algorithm 2.2** Compute table F for SSR FPTAS

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ and a parameter $\varepsilon \in (0, 1)$

1: $K_n \leftarrow \frac{\varepsilon \cdot a_n}{2 \cdot n}$
2: **make** sets $A^u \leftarrow \emptyset$ **and** $A^l \leftarrow \emptyset$
3: **for** $i = 1$ to $n$ **do**
4: $\quad a_i^u \leftarrow \lceil \frac{a_i}{K_n} \rceil$, $A^u \leftarrow A^u \cup a_i^u$
5: $\quad a_i^l \leftarrow \lfloor \frac{a_i}{K_n} \rfloor$, $A^l \leftarrow A^l \cup a_i^l$
6: **end for**
7: **make** table $F_k[x, y]$ with size $k = n$ and $x = y = \sum_{i=1}^{n} a_i^u$
8: **for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a_i^u\}$ **do**
9: $\quad F_k[x, y] \leftarrow \vec{\emptyset}$
10: **end for**

**for** $i = 1$ to $n$ **do**
    $F_i[a_i^u, 0] \leftarrow (\{i\}, \emptyset, a_i, 0)$
    $F_i[0, a_i^l] \leftarrow (\emptyset, \{i\}, 0, a_i)$
**end for**
**for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a_i^u\}$ **do**
    **for all** $0 < q < k$ **do**
        **if** $F_q[x - a_k^u, y] \neq \vec{\emptyset}$ **then**
            **if** $F_k[x, y] = \vec{\emptyset}$ **then**
                $(S_1, S_2, sum_1, sum_2) \leftarrow F_q[x - a_k^u, y]$
                $F_k[x, y] \leftarrow (S_1 \cup \{k\}, S_2, sum_1 + a_k, sum_2)$
            **else if** $F_k[x, y] \neq \vec{\emptyset}$ **then**
                $(S_{1q}, S_{2q}, sum_{1q}, sum_{2q}) \leftarrow F_q[x - a_k^u, y]$
                $(S_{1k}, S_{2k}, sum_{1k}, sum_{2k}) \leftarrow F_k[x, y]$
                **if** $sum_{1q} + a_k > sum_{1k}$ **then**
                    $F_k[x, y] \leftarrow (S_{1q} \cup k, S_{2q}, sum_{1q} + a_k, sum_{2q})$
                **end if**
            **end if**
        **end if**
        **if** $F_q[x, y - a_k^l] \neq \vec{\emptyset}$ **then**
            **if** $F_k[x, y] = \vec{\emptyset}$ **then**
                $(S_1, S_2, sum_1, sum_2) \leftarrow F_q[x, y - a_k^l]$
                $F_k[x, y] \leftarrow (S_1, S_2 \cup \{k\}, sum_1, sum_2 + a_k)$
            **else if** $F_k[x, y] \neq \vec{\emptyset}$ **then**
                $(S_{1q}, S_{2q}, sum_{1q}, sum_{2q}) \leftarrow F_q[x, y - a_k^l]$
                $(S_{1k}, S_{2k}, sum_{1k}, sum_{2k}) \leftarrow F_k[x, y]$
                **if** $sum_{1q} > sum_{1k}$ **then**
                    $F_k[x, y] \leftarrow (S_{1q}, S_{2q} \cup \{k\}, sum_{1q}, sum_{2q} + a_k)$
                **end if**
            **end if**
        **end if**
    **end for**
**end for**
**return** $F$

Εδώ καλό είναι να αναφερθεί ότι το μέγεθος του πίνακα, λόγο της διαίρεσης που κάναμε, είναι $n \times \sum_{i=1}^{n} a_i^u, \times \sum_{i=1}^{n} a_i^u$ όπου το άθροισμα $\sum_{i=1}^{n} a_i^u$ φράσσετε πολυωνυμικά από το $n$ και το $\frac{1}{\varepsilon}$ αφού:

$$\sum_{i=1}^{n} a_i^u \leq \sum_{i=1}^{n} \lceil \frac{a_i}{K_n} \rceil \leq \sum_{i=1}^{n} \lceil \frac{a_n}{K_n} \rceil \leq \sum_{i=1}^{n} (\frac{a_n}{K_n} + 1)$$
$$= n \cdot (a_n \cdot \frac{2 \cdot n}{\varepsilon \cdot a_n} + 1) = \frac{2 \cdot n^2}{\varepsilon} + n$$

Τώρα πρέπει να επιλέξουμε τα κατάλληλα σύνολα. Ο Sub-Algorithm 2.3 επιστρέφει τα σύνολα $S_1, S_2$ για τα οποία ισχύει ότι ή για $y \neq 0$ υπάρχουν $k, x, y$ τέτοια ώστε $F_k[x, y] = (S_1, S_2, sum_1, sum_2) \neq \vec{\emptyset}$ και τα $x, y$ μεγιστοποιούν την τιμή $\frac{x}{y} \leq 1 + \varepsilon$ ή για $y = 0$ μεγιστοποιούμε την τιμή $x \leq C$ (όπου $C$ προσδιορίζετε από ένα επιπλέον στοιχείο $a$ το οποίο επηρεάζει και το $S_2$ που θα επιστραφεί). Επιπλέον και στις δύο περιπτώσεις θέλουμε το πραγματικό άθροισμα $\sum_1$ να είναι μεγαλύτερο του στοιχείου $a_n$. Η τιμή του επιπλέον στοιχείου καθώς και γιατί βάλαμε αυτές τις προϋποθέσεις θα ξεκαθαρίσουν κατά την διάρκεια της απόδειξης.

---

**Sub-Algorithm 2.3** Find maximum $\frac{x}{y} \leq \frac{1}{1-\varepsilon}$ and $\frac{x}{C} \leq \frac{1}{1-\varepsilon}$ with the above conditions

---

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$, an integer $a$ and a parameter $\varepsilon \in (0, 1)$

$C = \lfloor \frac{2 \cdot n \cdot a}{\varepsilon \cdot a_n} \rfloor$
**run** Sub-Algorithm 2.2 with **input** $A$, $\varepsilon$ and **output** $F$
$k_0 \leftarrow 0$, $x_0 \leftarrow 0$, $y_0 \leftarrow 0$, $max \leftarrow 0$
**for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a_i^u\}$ **do**
    **if** $F_k[x, y] \neq \vec{\emptyset}$ **then**
        $(S_1, S_2, sum_1, sum_2) \leftarrow F_k[x, y]$
        **if** $y = 0$ **then**
            **if** $C \neq 0$ **and** $max \leq \frac{x}{C} \leq \frac{1}{1-\varepsilon}$ **and** $sum_1 \geq a_n$ **then**
                $k_0 \leftarrow k$, $x_0 \leftarrow x$, $y_0 \leftarrow y$, $max \leftarrow \frac{x}{C}$
            **end if**
        **else if** $max \leq \frac{x}{y} \leq \frac{1}{1-\varepsilon}$ **and** $sum_1 \geq a_n$ **then**
            $k_0 \leftarrow k$, $x_0 \leftarrow x$, $y_0 \leftarrow y$, $max \leftarrow \frac{x}{y}$
        **end if**
    **end if**
**end for**
**if** $k_0 = 0$ **then**
    $S_1 \leftarrow \emptyset$, $S_2 \leftarrow \emptyset$, $max = 0$
**else**
    $(S_1, S_2, sum_1, sum_2) \leftarrow F_{k_0}[x_0, y_0]$
    **if** $sum_2 = \emptyset$ **then**
        $S_2 \leftarrow n + 1$, $sum_2 \leftarrow a$, $max = min\{\frac{sum_1}{sum_2}, \frac{sum_2}{sum_1}\}$
    **else**
        $max = min\{\frac{sum_1}{sum_2}, \frac{sum_2}{sum_1}\}$
    **end if**
**end if**
**return** $S_1, S_2$ **and** $max$

---

Ολοκληρώνουμε τρέχοντας την παραπάνω διαδικασία για όλα τα στιγμιότυπα. Ακριβέστερα λύνουμε ακριβώς το πρόβλημα μέχρι ένα $a_{m_0}$ και μετά τρέχουμε τον προσαρμοσμένο αλγόριθμο για τα μεγαλύτερα $A^m = \{a_1, a_2, ..., a_m\}$.

**Algorithm 2.2** FPTAS for SSR

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ and a parameter $\varepsilon \in (0, 1)$

**sort** $A$
**find** the first $a_{m_0} \in A$ such that $\sum_{i=1}^{m_0} a_i > 2 \cdot n^2$
$A' \leftarrow \{a_1, a_2, ..., a_{m_0-1}\}$
**run** Algorithm2.1 with **input** $A'$ and **output** $S_1^*, S_2^*$ and $Opt$
**for** $m = m_0 - 1$ to n **do**
    **if** $m \neq n$ **then**
        $a = a_{m+1}$
    **else**
        $a = 0$
    **end if**
    $A^m \leftarrow \{a_1, a_2, ..., a_m\}$
    **run** Sub-Algorithm2.3 with **input** $A^m$, $a$, $\varepsilon$ and **output** $(S_1, S_2), Opt_m$
    **if** $Opt \leq Opt_m$ **then**
        $Opt \leftarrow Opt_m$
        $S_1^* \leftarrow S_1$, $S_2^* \leftarrow S_2$
    **end if**
**end for**
**if** $\sum_{i \in S_1^*} a_i \leq \sum_{i \in S_2^*} a_i$ **then**
    $S_1 \leftarrow S_1^*$, $S_2 \leftarrow S_2^*$
**else**
    $S_1 \leftarrow S_2^*$, $S_2 \leftarrow S_1^*$
**end if**
**return** $S_1, S_2$ **and** $Opt$

### 2.2.2 Απόδειξη Ορθότητας

Εδώ θα δείξουμε ότι ο αλγόριθμος που αναπτύξαμε είναι FPTAS. Η απόδειξη έχει τρία βασικά σημεία. Πρώτον ότι επιστρέφει πάντα κάποια σύνολα, δεύτερον ότι η τιμή αυτή είναι όντος η προσέγγιση που ισχυριζόμαστε και τρίτον ότι η πολυπλοκότητα είναι φραγμένη πολυωνυμικά ως προς το $n$ και το $\frac{1}{\varepsilon}$. Αρχικά θα δείξουμε μια σχέση μεταξύ των στοιχείων $a_n, a_{n-1}$ και των βέλτιστων συνόλων, όταν ξέρουμε ότι το $a_n$ είναι το μέγιστο στοιχείο που σίγουρα χρησιμοποιείται (ενώ το $a_{n-1}$ είναι το αμέσως μεγαλύτερο αλλά δεν χρησιμοποιείται κατ' ανάγκη). Πριν προχωρήσουμε στην διατύπωση του λήμματος να υπενθυμίσουμε ότι τα σύνολα που μας ενδιαφέρουν είναι ξένα μεταξύ τους.

**Λήμμα 2.1.** *Έστω ένα ταξινομημένο σύνολο $A = \{a_1, a_2, ..., a_n\}$, αν $S_{1Opt}$, $S_{2Opt}$ είναι δύο ξένα υποσύνολα του A για τα οποία έχουμε:*

$$\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} = max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \mid \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \leq 1 \text{ , } S_1 \cap S_2 = \emptyset\} \text{ και}$$
$$max\{a_i \mid i \in S_{1Opt} \cup S_{2Opt}\} = a_n$$

*τότε ισχύει:*

$$a_{n-1} \leq \sum_{i \in S_{1Opt}} a_i \leq \sum_{j \in S_{2Opt}} a_j$$

*Απόδειξη.* Αρχικά αφού $\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} \leq 1$ συμπεραίνουμε άμεσα το δεύτερο κομμάτι της ανίσωσης. Όσο για το πρώτο κομμάτι θα εξετάσουμε τις ακόλουθες περιπτώσεις:

**Περίπτωση 1.** $n \in S_{1Opt}$
*Τότε έχουμε* $a_{n-1} \leq a_n \leq \sum_{i \in S_{1Opt}} a_i$.

**Περίπτωση 2.** $n \in S_{2Opt}$
*Εδώ αν υποθέσουμε πως* $\sum_{i \in S_{1Opt}} a_i \leq a_{n-1}$ *έχουμε:*

$$\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} \leq \frac{a_{n-1}}{\sum_{j \in S_{2Opt}} a_j} \leq \frac{a_{n-1}}{a_n} \leq 1$$

*το οποίο είναι άτοπο αφού τα σύνολα* $S_{1Opt}, S_{2Opt}$ *είναι αυτά που μας δίνουν το βέλτιστο λόγο.*

Άρα και στις δύο περιπτώσεις ισχύει η ανίσωση του λήμματος. □

Από εδώ και πέρα για κάποιο σύνολο A θα συμβολίζουμε με $S_{1Opt}$ και $S_{2Opt}$ τα, ξένα μεταξύ τους, υποσύνολα του A που μας δίνουν βέλτιστη προσέγγιση του SSR, δηλαδή $\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} = max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \mid \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \leq 1 \text{ , } S_1 \cap S_2 = \emptyset\}$ (χωρίς να γίνετε χρίση του μέγιστου στοιχείου όπως στο προηγούμενο λήμμα). Ακολουθεί ένα λήμμα που ασχολείται με κάποιες ανισότητες που προκύπτουν από τις στρογγυλοποιήσεις των $a_i^u$ και $a_i^l$.

**Λήμμα 2.2.** *Αν έχω ένα σύνολο ακεραίων* $A = \{a_1, a_2, ..., a_n\}$ *και τα προσαρμοσμένα στοιχεία* $a_i^u = \lceil \frac{a_i}{K} \rceil$, $a_i^l = \lfloor \frac{a_i}{K} \rfloor$ *(όπου* $K$ *μια σταθερά) τότε ισχύουν τα ακόλουθα:*

$$1.\ \frac{a_i}{K} \leq a_i^u \leq \frac{a_i}{K} + 1$$
$$2.\ \frac{a_i}{K} - 1 \leq a_i^l \leq \frac{a_i}{K}$$

*Απόδειξη.* Οι σχέσεις είναι προφανείς από τον ορισμό των $a_i^u$ και $a_i^l$. □

**Λήμμα 2.3.** *Αν* $A = \{a_1, a_2, ...a_n\}$ *και* $S_{1Opt}, S_{2Opt}$ *τα σύνολα που δίνουν τον βέλτιστο λόγο τότε υπάρχει κάποιο* $m \leq n$ *για το οποίο τα* $S_{1Opt}, S_{2Opt}$ *πληρούν τις προϋποθέσεις επιλογής του Sub-Algorithm 2.3 ή ο αλγόριθμος επιστρέφει ακριβώς τα* $S_{1Opt}$ *και* $S_{2Opt}$.

Πριν αποδείξουμε αυτό το λήμμα είναι καλό να δούμε τι μας λέει. Συμπεράσματα του λήμματος είναι ότι ο αλγόριθμος σε κάποιο σημείο έχει κάνει σύγκριση με την προσαρμοσμένη τιμή του λόγου των $S_{1Opt}, S_{2Opt}$ (εντός του Sub-Algorithm 2.3) ή της αληθινής τιμής του (στο αρχικό κομμάτι που λύνει το πρόβλημα για ακριβή λύση ενός στιγμιότυπου).

*Απόδειξη.* Αρχικά θέτω $m = max\{i \mid i \in S_{1Opt} \cup S_{2Opt}\}$. Αν $\sum_{i=1}^{m} a_i \leq 2 \cdot n^2$ θα επιστραφεί η ακριβής λύση αφού για αυτές τις τιμές τρέχουμε τον Algorithm 2.1 ο οποίος επιστρέφει τα ακριβή $S_{1Opt}$ και $S_{2Opt}$. Με δεδομένο πλέων ότι $\sum_{i=1}^{m} > 2 \cdot n^2$ εξετάζουμε δύο περιπτώσεις:

**Περίπτωση 1.** $a_m \leq \sum_{i \in S_{1Opt}} a_i$

*Κατά την κλήση του Sub-Algorithm 2.3 για το στιγμιότυπο $A^m = \{a_1, a_2, ..., a_m\}$ (το οποίο συμβαίνει σίγουρα αφού έχουμε $\sum_{i=1}^{m} a_i > 2 \cdot n^2$) έχω ότι για $x = \sum_{i \in S_{1Opt}} a_i^u$ και $y = \sum_{j \in S_{2Opt}} a_j^l$ το κελί $F_m[x, y] = (S_1, S_2, sum_1, sum_2) \neq \vec{\emptyset}$ καθώς υπάρχει τουλάχιστον ένα ζεύγος συνόλων ($S_{1Opt}$, $S_{2Opt}$) για αυτές τις τιμές. Επιπλέον αφού φροντίζουμε να μεγιστοποιούμε το $sum_1$ αυτό αναγκαστικά είναι μεγαλύτερο του $a_m$. Μένει να δείξουμε ότι $\frac{x}{y} \leq \frac{1}{1-\varepsilon}$. Με βάση τις στρογγυλοποιήσεις που κάναμε έχουμε:*

$$\frac{x}{y} = \frac{\sum_{i \in S_{1Opt}} a_i^u}{\sum_{j \in S_{2Opt}} a_j^l} \leq \frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \text{ (λήμμα 2.2)}$$

**Υπο-περίπτωση 1.1.** $\frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \leq 1$

*Τότε είναι προφανές ότι $\frac{x}{y} \leq 1 \leq \frac{1}{1-\varepsilon}$ και άρα πληρούνται οι προϋποθέσεις επιλογής.*

**Υπο-περίπτωση 1.2.** $\frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \geq 1$

*Αφού το κλάσμα είναι μεγαλύτερο της μονάδας αν αφαιρέσουμε ίση ποσότητα από αριθμητή και παρονομαστή αυτό θα μεγαλώσει και άρα έχουμε:*

$$\frac{x}{y} \leq \frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \leq \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m - n_{1Opt} \cdot K_m}$$
$$\leq \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j - (n_{1Opt} + n_{2Opt}) \cdot \frac{\varepsilon \cdot a_m}{2 \cdot m}}$$

*αλλά αφού $max\{i \mid i \in S_{1Opt} \cup S_{2Opt}\} = m$ σημαίνει ότι $n_{1Opt} + n_{2Opt} \leq m$ και*

*άρα:*

$$\frac{x}{y} \leq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j - \frac{\varepsilon\cdot a_m}{2}} = \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \cdot \frac{1}{1 - \frac{\varepsilon\cdot a_m}{2\cdot\sum_{j\in S_{2Opt}} a_j}}$$

$$\leq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \cdot \frac{1}{1-\varepsilon} \text{ (λήμμα 2.1)}$$

$$\leq \frac{1}{1-\varepsilon}$$

*Οπότε όταν* $a_m \leq \sum_{i\in S_{1Opt}} a_i$ *στην κλήση του Sub-Algorithm 2.3 με είσοδο* $A^m = \{a_1, a_2, ..., a_m\}$ *πληρούνται οι προϋποθέσεις επιλογής για τα* $S_{1Opt}$ *και* $S_{2Opt}$.

**Περίπτωση 2.** $a_m > \sum_{i\in S_{1Opt}} a_i$

*Αφού το* $m = max\{i \mid i \in S_{1Opt} \cup S_{2Opt}\}$ *προφανώς ανήκει στο* $S_{2Opt}$ *και μάλιστα είναι το μοναδικό αφού τα* $S_{1Opt}, S_{2Opt}$ *μας δίνουν το βέλτιστο λόγο. Τότε όταν τρέχουμε τον Sub-Algorithm 2.3 με είσοδο* $A^{m-1} = a_1, a_2, ..., a_{m-1}$ *(το οποίο συμβαίνει σίγουρα αφού έχουμε* $\sum_{i=1}^{m} > 2\cdot n^2$*) για* $x = \sum_{i\in S_{1Opt}} a_i^u$ *έχουμε το κελί* $F_k[x, 0] = (S_1, S_2, sum_1, sum_2) \neq \vec{\emptyset}$ *(για* $k = max\{i \mid i \in S_{1Opt}\}$*) και αφού κρατάμε τα σύνολα που μεγιστοποιούν το* $sum_1$ *έχω* $sum_1 \geq \sum_{i\in S_{1Opt}} a_i \geq a_{m-1}$ *(από λήμμα 2.1). Μένει να δείξουμε ότι* $\frac{x}{C} \leq \frac{1}{1-\varepsilon}$ *για το* $C$ *εκείνης της κλήσης. Για το* $C$ *σε εκείνη την επανάληψη έχουμε ότι* $C = \lfloor \frac{2\cdot m\cdot a_m}{\varepsilon\cdot a_{m-1}} \rfloor$*. Αν* $n_{1Opt} =\mid S_{1Opt} \mid$*:*

$$\frac{x}{C} = \frac{\sum_{i\in S_{1Opt}} a_i^u}{\lfloor \frac{2\cdot(m-1)\cdot a_m}{\varepsilon\cdot a_{m-1}} \rfloor} \leq \frac{\sum_{i\in S_{1Opt}} \frac{a_i}{K_{m-1}} + n_{1Opt}}{\frac{2\cdot(m-1)\cdot a_m}{\varepsilon\cdot a_{m-1}} - 1} \text{ (λήμμα 2.2)}$$

*και αφού* $K_{m-1} = \frac{\varepsilon\cdot a_{m-1}}{2\cdot(m-1)}$

$$\frac{x}{C} \leq \frac{\sum_{i\in S_{1Opt}} \frac{a_i}{K_{m-1}} + n_{1Opt}}{\frac{2\cdot(m-1)\cdot a_m}{\varepsilon\cdot a_{m-1}} - 1} \leq \frac{\sum_{i\in S_{1Opt}} a_i + n_{1Opt}\cdot K_{m-1}}{a_m - K_{m-1}}$$

*Όμοια με πριν θα πάρουμε δύο περιπτώσεις*

**Υπο-περίπτωση 2.1.** $\frac{\sum_{i\in S_{1Opt}} a_i + n_1\cdot K_{m-1}}{a_m - K_{m-1}} \leq 1$

*Εδώ είναι προφανές ότι* $\frac{x}{C} \leq 1 \leq \frac{1}{1-\varepsilon}$.

**Υπο-περίπτωση 2.2.** $\frac{\sum_{i\in S_{1Opt}} a_i + n_{1Opt}\cdot K_{m-1}}{a_m - K_{m-1}} > 1$

*Αν αφαιρέσουμε από αριθμητή και παρονομαστή το* $n_{1Opt}\cdot K_{m-1}$ *έχουμε:*

$$\frac{x}{C} \leq \frac{\sum_{i\in S_{1Opt}} a_i}{a_m - (n_{1Opt}+1)\cdot K_{m-1}} \leq \frac{\sum_{i\in S_{1Opt}} a_i}{a_m} \cdot \frac{1}{1 - \frac{(n_{1Opt}+1)\cdot K_{m-1}}{a_m}}$$

$$\leq \frac{1}{1 - \frac{(n_{1Opt}+1)\cdot\varepsilon\cdot a_{m-1}}{2\cdot(m-1)\cdot a_m}} \leq \frac{1}{1-\varepsilon}$$

*όπου η τελευταία ισχύει λόγο των ανισοτήτων* $a_{m-1} \leq a_m$ *και* $n_{1Opt}+1 \leq 2\cdot(m-1)$ *(που ισχύει* $\forall m \geq 1$ *αφού* $n_{1Opt} < m$*).*

Άρα αν έχω $m = max\{i \mid i \in S_{1Opt} \cup S_{2Opt}\}$ και $\sum_{i=1}^{m} a_i > 2 \cdot n^2$ τα $S_{1Opt}, S_{2Opt}$ πληρούν τα κριτήρια επιλογής για κάποιο στιγμιότυπο του προβλήματος. □

Μένει να δείξουμε ότι ο λόγος που επιστρέφει ο αλγόριθμος είναι στο διάστημα που ισχυριζόμαστε.

**Θεώρημα 2.1.** *Για τα σύνολα* $S_1$ *και* $S_2$ *που επιστρέφει ο αλγόριθμος ισχύει:*

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \in \Big[(1-\varepsilon)\cdot\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}\Big]$$

*Απόδειξη.* Αν $n_0 = max\{i \mid i \in S_{1Opt} \cup S_{2Opt}\}$ και $n_0 \leq m_0$ (γραμμή 2 του Algorithm 2.2) τότε θα έχει επιστραφεί ο βέλτιστος λόγος, αφού θα έχει βρεθεί από τον ακριβή αλγόριθμο. Αν δεν ισχύει αυτό, λόγο λήμματος 2.3, έχουμε ότι υπάρχει κάποιο m για το οποίο τα $S_{1Opt}$ και $S_{2Opt}$ πληρούν τις προϋποθέσεις επιλογής. Θέτουμε $S_1^m$ και $S_2^m$ τα σύνολα που επέστρεψε ο αλγόριθμος σε εκείνη την επανάληψη. Από τις προϋποθέσεις επιλογής έχουμε ότι:

$$\frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \leq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l} \leq 1+\varepsilon \tag{2.1}$$

Θα δείξουμε ότι ο λόγος που επιστρέφει αυτή η επανάληψη είναι μεταξύ του 1 και του $(1-\varepsilon)\cdot\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}$:

**Περίπτωση 1.** $\frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} \geq 1$

*Τότε έχουμε:*

$$\begin{aligned} 1 \leq \frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} &\leq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l} \text{ (λήμμα 2.2)} \\ &\leq \frac{1}{1-\varepsilon} \end{aligned}$$

*και άρα:*

$$(1-\varepsilon)\cdot\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \leq (1-\varepsilon) \leq \frac{\sum_{j\in S_2^m} a_j}{\sum_{i\in S_1^m} a_i} \leq 1$$

**Περίπτωση 2.** $\frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} \leq 1$

*Εδώ θα κάνουμε χρήση του λήμματος 2.2*

$$1 \geq \frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} \geq \frac{\sum_{i\in S_1^m} a_i^u - n_1}{\sum_{j\in S_2^m} a_j^l + n_2}$$

*όπου $n_1$ και $n_2$ οι πληθικότητες των $S_1^m$ και $S_2^m$ αντίστοιχα. Αφού αυτά είναι μικρότερα του ένα αν αφαιρέσω το $n_2$ από αριθμητή και παρονομαστή θα μικρύνει περισσότερο και άρα:*

$$\frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} \geq \frac{\sum_{i\in S_1^m} a_i^u - (n_1+n_2)}{\sum_{j\in S_2^m} a_j^l} = \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l} \cdot (1 - \frac{n_1+n_2}{\sum_{i\in S_1^m} a_i^u})$$

*Μένει να δείξουμε ότι* $\frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l} \geq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}$ *και* $(1 - \frac{n_1+n_2}{\sum_{i\in S_1^m} a_i^u}) \geq 1-\varepsilon$.
*Για το πρώτο έχουμε ότι:*

$$\frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l} \geq \frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \geq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \textit{ (λήμμα 2.2, εξ. 2.1)}$$

*ενώ για το δεύτερο αρκεί να δείξω ότι* $\frac{n_1+n_2}{\sum_{i\in S_1^m} a_i^u} \leq \varepsilon$ *το οποίο ισχύει αφού:*

$$\begin{aligned}
\frac{n_1+n_2}{\sum_{i\in S_1^m} a_i^u} &\leq \frac{n_1+n_2}{\sum_{i\in S_1^m} \frac{a_i}{K_m}} \textit{ (λήμμα 2.2)} \\
&\leq \frac{2\cdot m\cdot K_m}{\sum_{i\in S_1^m} a_i} = \frac{\varepsilon\cdot a_m}{\sum_{i\in S_1^m} a_i} \\
&\leq \varepsilon \textit{ (λήμμα 2.1)}
\end{aligned}$$

*οπότε και συμπεραίνουμε ότι* $1 \geq \frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} \geq (1-\varepsilon)\cdot \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}$

Οπότε, αν $S_1$ και $S_2$ τα σύνολα που επιστρέφει ο Algorithm 2.2, λόγο των γραμμών 13 έως 16 έχουμε:

$$(1-\varepsilon)\cdot \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \leq min\Big\{\frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j}, \frac{\sum_{j\in S_2^m} a_j}{\sum_{i\in S_1^m} a_i}\Big\} \leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \leq 1$$

και λόγο ορισμού των $S_{1Opt}$ και $S_{2Opt}$ είναι προφανές:

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \in \Big[(1-\varepsilon)\cdot \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}\Big]$$

□

Μας έμεινε να σχολιάσουμε την πολυπλοκότητα του αλγορίθμου. Η κατασκευή και το γέμισμα του πίνακα F στην χειρότερη περίπτωση θέλει $O(\frac{n^6}{\varepsilon^2})$ αφού για να γεμίσει κάθε ένα κελί θέλει να ελέγξει (στην χειρότερη περίπτωση) n κελιά. Επιπλέον θα κατασκευάσει τον F n φορές (στην χειρότερη περίπτωση) και άρα η συνολική πολυπλοκότητα είναι $O(\frac{n^7}{\varepsilon^2})$. Ο εντοπισμός κάθε φορά των συνόλων που πληρούν τις προϋποθέσεις επιλογής απαιτεί ένα πέρασμα του F και άρα δεν επιβαρύνει την πολυπλοκότητα όπως και το κομμάτι στο οποίο καλεί τον ακριβή αλγόριθμο που τρέχει σε χρόνο $O(n^6)$.

# Κεφάλαιο 3

# Υποσύνολα λόγου r

## 3.1 Ορισμός

Εδώ θα δούμε το πρόβλημα βελτιστοποίησης που αντιστοιχεί στο πρόβλημα απόφασης 2. Μπορούμε ορίσουμε αυτό το πρόβλημα βελτιστοποίησης ως εξής:

**Πρόβλημα 7** (Factor-r SSR)**.** *Δοθέντος ενός συνόλου από n θετικούς ακεραίους* $A = \{a_1, a_2, ..., a_n\}$ *και ενός θετικού αριθμού r, ψάχνουμε δύο μη κενά και ξένα σύνολα* $S_1, S_2 \subseteq \{1, 2, ..., n\}$ *τέτοια ώστε για οποιαδήποτε* $S'_1$ *και* $S'_2$ *μη κενά και ξένα υποσύνολα του* $\{1, 2, ..., n\}$ *να ισχύει:*

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} = \max_{S'_1,S'_2} \Big\{ \frac{\sum_{i\in S'_1} a_i}{\sum_{j\in S'_2} a_j} | \frac{r \cdot \sum_{i\in S'_1} a_i}{\sum_{j\in S'_2} a_j} \leq 1 \Big\}$$

Μπορούμε να κατασκευάσουμε διάφορα προβλήματα βελτιστοποίησης παρόμοια με το παραπάνω. Να σχολιάσουμε ότι ο ψευδοπολυωνυμικός αλγόριθμος που αναπτύσσουμε παρακάτω λύνει ακριβώς το πρόβλημα 7 αλλά το FPTAS αυτού του κεφαλαίου δεν προσεγγίζει ακριβώς αυτό.

## 3.2 Ψευδοπολυωνυμικός Αλγόριθμος

Για να λύσουμε αυτό το πρόβλημα βελτιστοποίησης θα κατασκευάσουμε ακριβώς τον ίδιο πίνακα με αυτόν της παραγράφου 2.1 και θα αλλάξουμε απλά την συνθήκη επιλογής στον Algorithm 2.1. Οπότε ο αλγόριθμος που λύνει το πρόβλημα είναι ο εξής:

**Algorithm 3.1** Solve Factor-r Sum Subsets problem

**Require:** a set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ and a parameter $r$

**sort** $A$
**run** Sub-Algorithm 2.1 with **input** $A$ and **output** $F$
$k_0 \leftarrow 0$, $x_0 \leftarrow 0$, $y_0 \leftarrow 0$
$max \leftarrow 0$
**for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{1, 2, ..., \sum_{i=1}^{n} a_i\}$ **do**
    **if** $F_k[x, y] \neq \vec{\emptyset}$ **and** $max \leq \frac{x}{y} \leq \frac{1}{r}$ **then**
        $k_0 \leftarrow k$, $x_0 \leftarrow x$, $y_0 \leftarrow y$
        $max \leftarrow \frac{x}{y}$
    **end if**
**end for**
**return** $F_{k_0}[x_0, y_0]$ **and** $max$

Ο αλγόριθμος αυτός είναι παρόμοιος με αυτόν που λύνει το πρόβλημα απόφασης 2 στο [7] με κύριες διαφορές ότι εδώ βελτιστοποιούμε το λόγο(ενώ εκεί έλεγε απλά αν υπάρχουν σύνολα με ακριβώς αυτόν τον λόγο) και ότι εμείς στα κελιά του πίνακα κρατάμε σύνολα.

## 3.3 FPTAS

### 3.3.1 Αλγόριθμος

Εδώ θα παρουσιάσουμε ένα FPTAS για το πρόβλημα 7 το οποίο για $r \geq 1$ θα επιστρέφει δύο ξένα $S_1$, $S_2$ υποσύνολα του $\{1, 2, ..., n\}$ και ένα λόγο $\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$ για τον οποίο ισχύει:

$$\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \in \Big[(1 - \varepsilon) \cdot \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}, \frac{1}{r} \cdot (1 + \varepsilon)\Big] \tag{3.1}$$

όπου με τον όρο $\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}$ αναφερόμαστε στο μέγιστο λόγο αθροισμάτων που είναι μικρότερος ή ίσος από το επιθυμητό $\frac{1}{r}$ και το $\varepsilon$ είναι η παράμετρος ακρίβειας. Αν το δοθέν r είναι μικρότερο του 1 τότε ισχύουν τα ίδια αν θέσουμε $\hat{r} = \frac{1}{r}$. Ο αλγόριθμος αυτός προσπαθεί να βελτιστοποιήσει το λόγο όπως στην προηγούμενη παράγραφο αλλά δυστυχώς, εδώ, δεν καταφέραμε να μας επιστρέφει λόγο σίγουρα μικρότερο του $\frac{1}{r}$.

Για να προσεγγίσουμε το πρόβλημα θα τροποποιήσουμε τον αλγόριθμο βελτιστοποίησης που αναπτύξαμε στην παράγραφο 2.1. Το FPTAS θα δέχεται επιπλέον σαν είσοδο έναν αριθμό $r$ ο οποίος προσδιορίζει το λόγο το οποίο θέλουμε να προσεγγίζει ο λόγος των συνόλων που θα επιστρέψει ο αλγόριθμος. Η μια από τις τροποποιήσεις που χρειάζεται να γίνουν είναι στις συνθήκες επιλογής αλλά αυτό

δεν είναι αρκετό. Παραθέτουμε τον κώδικα που λύνει το πρόβλημα προσπαθώντας να μην επαναλαμβάνουμε μεγάλα κομμάτια που έχουν ξανά γραφεί.

**Sub-Algorithm 3.1** Compute table F for Factor-r Sum Subsets problem FPTAS

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$, a parameter $r$ and a parameter $\varepsilon \in (0, 1)$

1: in Sub-Algorithm 2.2 **replace** line 1 with:
2: $K_n = \frac{\varepsilon \cdot a_n}{2 \cdot r \cdot n}$
3: **run** "the new Sub-Algorithm" with **input** $A$, $r$, $\varepsilon$ and **output** $F$
4: **return** $F$

Ο παραπάνω αλγόριθμος κατασκευάζει ακριβώς τον ίδιο πίνακα με τον Sub-Algorithm 2.2 απλά χρησιμοποιώντας διαφορετικές τιμές (εξού και η αλλαγή που κάναμε στην γραμμή 1). Ο Sub-Algorithm 3.2, που ακολουθεί, αποφασίζει ποια σύνολα θα επιστρέψει με βάση τον πίνακα F και το στοιχείο $a_+$.

**Sub-Algorithm 3.2** Select the sets from F

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$, an integer $a_+$ and three parameters $r$, $lim$, $\varepsilon$ with $\varepsilon \in (0, 1)$

1: **sort** $A$
2: $C = \lfloor \frac{2 \cdot r \cdot n \cdot a_+}{\varepsilon \cdot a_n} \rfloor$
3: **run** Sub-Algorithm 3.1 with **input** $A$, $r$, $\varepsilon$ and **output** F
4: $k_0 \leftarrow 0$, $x_0 \leftarrow 0$, $y_0 \leftarrow 0$, $max \leftarrow 0$
5: **for all** $k \in \{1, 2, ..., n\}$ **and** $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a_i^u\}$ **do**
6: **if** $F_k[x, y] \neq \vec{\emptyset}$ **then**
7: $(S_1, S_2, sum_1, sum_2) \leftarrow F_k[x, y]$
8: **if** $y = 0$ **then**
9: **if** $C \neq 0$ **and** $max \leq \frac{x}{C} \leq lim$ **and** $r \cdot sum_1 \geq a_n$ **then**
10: $k_0 \leftarrow k$, $x_0 \leftarrow x$, $y_0 \leftarrow y$, $max \leftarrow \frac{x}{C}$
11: **end if**
12: **else if** $max \leq \frac{x}{y} \leq lim$ **and** $r \cdot sum_1 \geq a_n$ **then**
13: $k_0 \leftarrow k$, $x_0 \leftarrow x$, $y_0 \leftarrow y$, $max \leftarrow \frac{x}{y}$
14: **end if**
15: **end if**
16: **end for**
17: **if** $k_0 = 0$ **then**
18: $S_1 \leftarrow \emptyset$, $S_2 \leftarrow \emptyset$, $max = 0$
19: **else**
20: $(S_1, S_2, sum_1, sum_2) \leftarrow F_{k_0}[x_0, y_0]$
21: **if** $sum_2 = \emptyset$ **then**
22: $S_2 \leftarrow n + 1$, $sum_2 \leftarrow a_+$, $max = min\{\frac{sum_1}{sum_2}, \frac{sum_2}{sum_1}\}$
23: **else**

24: $max = min\{\frac{sum_1}{sum_2}, \frac{sum_2}{sum_1}\}$
25: **end if**
26: **end if**
27: **return** $S_1$, $S_2$ **and** $max$

Παρακάτω εμφανίζουμε ένα κομμάτι κώδικα το οποίο θα χρησιμοποιούμε για να μετατρέπουμε την είσοδο του αλγορίθμου ώστε να τρέχει σωστά.

**Sub-Algorithm 3.3** Adapt the input

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$, a parameters $r$ and a parameter $\varepsilon \in (0, 1)$

1: **sort** $A$
2: **if** $r < 1$ **then**
3: $r \leftarrow \frac{1}{r}$
4: **end if**
5: **if** $\frac{1}{r} \cdot \frac{1}{1-\varepsilon} \leq 1$ **then**
6: $lim = \frac{1}{r} \cdot (1 + \varepsilon)$
7: **else if** $\frac{1}{r} \cdot (1 + \varepsilon) \leq 1 < \frac{1}{r} \cdot \frac{1}{1-\varepsilon}$ **then**
8: $\varepsilon \leftarrow \frac{\varepsilon}{r}$
9: $lim = \frac{1}{r} \cdot (1 + \varepsilon)$
10: **else**
11: $lim = r \cdot \frac{1}{1-\varepsilon}$
12: **end if**
13: **return** $A$, $r$, $lim$ **and** $\varepsilon$

Ακολουθεί το FPTAS που προσεγγίζει την βέλτιστη λύση του προβλήματος. Σε αυτό αρχικά προσαρμόζουμε τις παραμέτρους του προβλήματος μέσω του Sub-Algorithm3.3 και μετά ελέγχουμε την ακραία περίπτωση που δεν υπάρχει λόγος μικρότερος του $\frac{1}{r}$. Έπειτα υπολογίζουμε ένα στοιχείο $m_0$ μέχρι το οποίο τρέχουμε τον αλγόριθμο ακριβούς λύσης και στην συνέχεια προσεγγίζουμε για τα στιγμιότυπα $A^m \leftarrow \{a_1, a_2, ..., a_m\}$ με m μεγαλύτερο ίσο από την τιμή $m_0 - 1$. Ο αλγόριθμος ολοκληρώνεται επιστρέφοντας τον μεγαλύτερο από τους λόγους που υπολόγισε (κάνοντας χρίση των τιμών του $A$ και όχι των προσαρμοσμένων) και τα σύνολα που τον παράγουν.

**Algorithm 3.2** FPTAS for Factor-r Sum Subsets problem

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ a parameter $r$ and a parameter $\varepsilon \in (0, 1)$

1: **run** Sub-Algorithm3.3 with **input** $A$, $r$, $\varepsilon$ and **output** $A$, $r$, $lim$, $\varepsilon$
2: **if** $\frac{a_1}{\sum_i = 2^n a_i} \geq \frac{1}{r}$ **then**
3: **return** $\{1\}, \{2, 3, ..., n\}$ **and** $\frac{a_1}{\sum_{i=2}^{n} a_i}$
4: **else**

5: **find** the first $a_{m_0} \in A$ such that $\sum_{i=1}^{m_0} a_i > 2 \cdot r \cdot n^2$
6: $A' \leftarrow \{a_1, a_2, ..., a_{m_0-1}\}$
7: **run** Algorithm3.1 with **input** $A'$, $r$ and **output** $S_1^*, S_2^*$ and $max$
8: **for** $m = m_0 - 1$ to $n$ **do**
9: **if** $m \neq n$ **then**
10: $a_+ = a_{m+1}$
11: **else**
12: $a_+ = 0$
13: **end if**
14: $A^m \leftarrow \{a_1, a_2, ..., a_m\}$
15: **run** Sub-Algorithm3.2 with **input** $A^m$, $a_+$, $r$, $lim$, $\varepsilon$ and **output** $(S_1, S_2)$, $Opt$
16: **if** $max \leq Opt$ **then**
17: $max \leftarrow Opt$
18: $S_1^* \leftarrow S_1$, $S_2^* \leftarrow S_2$
19: **end if**
20: **end for**
21: **if** $\sum_{i \in S_1^*} a_i \leq \sum_{i \in S_2^*} a_i$ **then**
22: $S_1 \leftarrow S_1^*$, $S_2 \leftarrow S_2^*$
23: **else**
24: $S_1 \leftarrow S_2^*$, $S_2 \leftarrow S_1^*$
25: **end if**
26: **return** $S_1, S_2$ **and** $max$
27: **end if**

### 3.3.2 Απόδειξη Ορθότητας

Στο συγκεκριμένο κεφάλαιο όπου αναφερόμαστε σε βέλτιστη λύση θα εννοούμε τα σύνολα που αποτελούν την λύση του προβλήματος 7 για κάποια είσοδο (που θα διευκρινίζουμε ποία είναι όταν δεν είναι προφανές). Όπως και στα προηγούμενα κεφάλαια θα ξεκινήσουμε την απόδειξη με ένα λήμμα που μας δίνει μια σχέση μεταξύ των αθροισμάτων των συνόλων που βελτιστοποιούν τον λόγο και των στοιχείων που χρησιμοποιούνται.

**Λήμμα 3.1.** *Αν για ένα ταξινομημένο σύνολο* $A = \{a_1, a_2, ..., a_n\}$ *και τα ξένα* $S_{1Opt}$, $S_{2Opt}$ *υποσύνολα του* $\{1, 2, ..., n\}$ *έχουμε:*

$$\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} = \max_{S_1, S_2 \subseteq \{1,2,...,n\}} \left\{ \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \mid \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \leq \frac{1}{r} \text{ και } S_1 \cap S_2 = \emptyset \right\}$$

*και* $\max\{S_1 \cup S_2\} = n$ *τότε ισχύει:*

$$a_{n-1} \leq r \cdot \sum_{i \in S_{1Opt}} a_i \leq \sum_{j \in S_{2Opt}} a_j \leq r \cdot \sum_{j \in S_{2Opt}} a_j$$

*Απόδειξη.* Αρχικά αφού $\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \leq \frac{1}{r}$ και $r \geq 1$ συμπεραίνουμε τις δύο εκ' των ανισοτήτων. Όσο για το πρώτο κομμάτι θα εξετάσουμε τις ακόλουθες περιπτώσεις:

**Περίπτωση 1.** $n \in S_{1Opt}$

*Τότε συμπεραίνουμε* $a_{n-1} \leq a_n \leq \sum_{i\in S_{1Opt}} a_i \leq r \cdot \sum_{i\in S_{1Opt}} a_i$.

**Περίπτωση 2.** $n \in S_{2Opt}$

*Εδώ αν υποθέσουμε πως το* $r \cdot \sum_{i\in S_{1Opt}} a_i \leq a_{n-1}$ *τότε μπορούμε να δούμε ότι* $\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \leq \frac{a_{n-1}}{\sum_{j\in S_{2Opt}} a_j} \cdot \frac{1}{r} \leq \frac{a_{n-1}}{a_n} \cdot \frac{1}{r} \leq \frac{1}{r}$ *και αφού τα σύνολα* $S_{1Opt}, S_{2Opt}$ *μας δίνουν το μέγιστο λόγο* $\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \leq \frac{1}{r}$ *καταλήξαμε σε* ***άτοπο***.

Άρα και στις δύο περιπτώσεις ισχύει η ανίσωση του λήμματος. □

**Λήμμα 3.2.** *Αν* $S_1$, $S_2$ *τα σύνολα που μας δίνουν το βέλτιστο λόγο τότε ισχύει ότι:*

$$\frac{\sum_{i\in S_1} a_i^u}{\sum_{j\in S_2} a_j^l} \leq \frac{\sum_{i\in S_1} a_i + n_1 \cdot K_m}{\sum_{j\in S_2} a_j - n_2 \cdot K_m} \leq \begin{cases} \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot (1+\varepsilon), \text{ αν } \frac{\sum_{i\in S_1} a_i + n_1 \cdot K_m}{\sum_{j\in S_2} a_j - n_2 \cdot K_m} \leq 1 \\ \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot \frac{1}{1-\varepsilon}, \text{ αν } \frac{\sum_{i\in S_1} a_i + n_1 \cdot K_m}{\sum_{j\in S_2} a_j - n_2 \cdot K_m} > 1 \end{cases}$$

*όπου* $n_1, n_2$ *οι πληθικότητες των αντίστοιχων συνόλων ενώ το* $m$ *και τα* $a_i^u$, $a_j^l$ *είναι αυτά που προκύπτουν από τον αλγόριθμο στην m επανάληψη διαιρώντας με* $K_m$ *όπου* $m = k-1$ *αν το* $S_2 = \{k\}$ *(δηλαδή είναι μονοσύνολο) ή* $m = max\{i \mid i \in S_1 \cup S_2\}$ *αλλιώς.*

*Απόδειξη.* Αρχικά να παρατηρήσουμε ότι για το m που μας δίνει το λήμμα ισχύει $r \cdot \sum_{i\in S_1} a_i \geq a_m$ καθώς για $k = max\{i \mid i \in S_1 \cup S_2\}$ τότε:

**Περίπτωση 1.** *Αν* $r \cdot \sum_{i\in S_1} a_i \geq a_k$

*Τότε για οποιοδήποτε m από αυτά που περιγράφουμε* $a_m \leq a_k \leq r \cdot \sum_{i\in S_1} a_i$

**Περίπτωση 2.** *Αν* $r \cdot \sum_{i\in S_1} a_i < a_k$

*Τότε αφού* $r \geq 1$ *ισχύει ότι* $k \in S_2$ *(αφού* $k = max\{i \mid i \in S_1 \cup S_2\}$*). Επιπλέον, αφού το τα σύνολα παράγουν τον βέλτιστο λόγο, το* $S_2$ *θα είναι μονοσύνολο (αν περιείχε περισσότερα στοιχεία δεν θα ήταν βέλτιστη η λύση) και άρα* $m = k - 1$ *το οποίο από το λήμμα 3.1 μας δίνει την παραπάνω σχέση.*

Αφού αποδείξαμε την σχέση $r \cdot \sum_{i\in S_1} a_i \geq a_m$, θα συνεχίσουμε παίρνοντας τις περιπτώσεις του λήμματος.

**Περίπτωση 1.** $\frac{\sum_{i\in S_1} a_i + n_1 \cdot K_m}{\sum_{j\in S_2} a_j - n_2 \cdot K_m} \leq 1$

*Εδώ αν προσθέσουμε την ίδια ποσότητα σε αριθμητή και παρονομαστή θα μεγαλώσει το κλάσμα οπότε:*

$$\frac{\sum_{i\in S_1} a_i^u}{\sum_{j\in S_2} a_j^l} \leq \frac{\sum_{i\in S_1} a_i + n_1 \cdot K_m}{\sum_{j\in S_2} a_j - n_2 \cdot K_m} \leq \frac{\sum_{i\in S_1} a_i + (n_1+n_2)\cdot K_m}{\sum_{j\in S_2} a_j}$$
$$= \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot (1 + \frac{(n_1+n_2)\cdot K_m}{\sum_{i\in S_1} a_i})$$

*και αντικαθιστώντας το $K_m$ έχουμε:*

$$\frac{\sum_{i\in S_1} a_i^u}{\sum_{j\in S_2} a_j^l} \leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot (1 + \frac{(n_1+n_2)\cdot a_m \cdot \varepsilon}{2\cdot r\cdot m \sum_{i\in S_1} a_i})$$
$$\leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot (1 + \frac{(n_1+n_2)\cdot \varepsilon}{2\cdot m})$$
$$\leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot (1+\varepsilon)$$

**Περίπτωση 2.** $\frac{\sum_{i\in S_1} a_i + n_1\cdot K_m}{\sum_{j\in S_2} a_j - n_2\cdot K_m} > 1$

*Εδώ η διαφορά με πριν είναι ότι θα αφαιρέσουμε για να μικρύνει το κλάσμα.*

$$\frac{\sum_{i\in S_1} a_i^u}{\sum_{j\in S_2} a_j^l} \leq \frac{\sum_{i\in S_1} a_i + n_1 \cdot K_m}{\sum_{j\in S_2} a_j - n_2 \cdot K_m} \leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j - (n_1+n_2)\cdot K_m}$$
$$= \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot \frac{1}{1 - \frac{(n_1+n_2)\cdot K_m}{\sum_{j\in S_2} a_j}}$$

*Τέλος αν αντικαταστήσουμε το $K_m$ και κάνουμε και χρήση του λήμματος 3.1 έχουμε:*

$$\frac{\sum_{i\in S_1} a_i^u}{\sum_{j\in S_2} a_j^l} \leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot \frac{1}{1 - \frac{(n_1+n_2)\cdot a_m\cdot\varepsilon}{2\cdot r\cdot m\sum_{j\in S_2} a_j}} \leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot \frac{1}{1-\frac{(n_1+n_2)\cdot\varepsilon}{2\cdot m}}$$
$$\leq \frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \cdot \frac{1}{1-\varepsilon}$$

□

Μια σημαντική σχέση που προκύπτει από το προηγούμενο λήμμα είναι ότι:

$$\frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \leq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \cdot \frac{1}{1-\varepsilon} \tag{3.2}$$

αφού $1+\varepsilon \leq \frac{1}{1-\varepsilon}$, $\forall\, \varepsilon \in (0,1)$.

Από εδώ και κάτω θα χωρίσουμε την απόδειξη με βάση την επιλογή που γίνετε στον Sub-Algorithm 3.3 λόγο των τιμών του r και του $\varepsilon$. Αρχικά να παρατηρήσουμε ότι ο αλγόριθμος αλλάζει την τιμή του r ώστε να είναι μεγαλύτερο του ένα και έπειτα κάνει την όποια επιλογή.

**Περίπτωση a.** $\frac{1}{r} \cdot \frac{1}{1-\varepsilon} \leq 1$

**Λήμμα 3.3.** *Αν $A = \{a_1, a_2, ...a_n\}$ και $S_{1Opt}, S_{2Opt}$ τα σύνολα που δίνουν τον βέλτιστο λόγο τότε υπάρχει κάποιο $m \leq n$ για το οποίο τα $S_{1Opt}, S_{2Opt}$ πληρούν τις προϋποθέσεις επιλογής από τον Sub-Algorithm 3.2 ή ο αλγόριθμος επιστρέφει ακριβώς τα $S_{1Opt}$ και $S_{2Opt}$.*

*Απόδειξη.* Αρχικά θέτω $m = max\{i \mid i \in S_{1Opt} \cup S_{2Opt}\}$. Αν $\sum_{i=1}^{m} a_i \leq 2 \cdot n^2$ θα επιστραφεί η ακριβής λύση αφού για αυτές τις τιμές τρέχουμε τον Algorithm 2.1 ο οποίος επιστρέφει τα ακριβή $S_{1Opt}$ και $S_{2Opt}$. Με δεδομένο πλέων ότι $\sum_{i=1}^{m} a_i > 2 \cdot n^2$ μας ενδιαφέρει να δούμε αν πληρούνται οι προϋποθέσεις επιλογής.

**Περίπτωση 1.** $a_m \leq r \cdot \sum_{i \in S_{1Opt}} a_i$

*Κατά την κλήση του Sub-Algorithm 3.2 για το στιγμιότυπο $A^m$ έχω ότι για $x = \sum_{i \in S_{1Opt}} a_i^u$ και $y = \sum_{j \in S_{2Opt}} a_j^l$ το κελί $F_m[x, y] = (S_1, S_2, \sum_1, \sum_2) \neq \emptyset$ καθώς υπάρχει τουλάχιστον ένα ζεύγος συνόλων($S_{1Opt}$, $S_{2Opt}$) για αυτές τις τιμές. Επιπλέον αφού φροντίζουμε να μεγιστοποιούμε το $\sum_1$ αυτό αναγκαστικά είναι μεγαλύτερο του $a_m$ (αφού αυτό ισχύει για την βέλτιστη λύση). Μένει να δείξουμε ότι $\frac{x}{y} \leq lim(= \frac{1}{r} \cdot (1 + \varepsilon))$. Με βάση τις στρογγυλοποιήσεις έχουμε:*

$$\frac{x}{y} = \frac{\sum_{i \in S_{1Opt}} a_i^u}{\sum_{j \in S_{2Opt}} a_j^l} \leq \frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \text{ (λήμμα 2.2)}$$

*όπου $n_{1Opt}$ και $n_{2Opt}$ οι πληθικότητες των δύο συνόλων.*

**Υπο-περίπτωση 1.1.** $\frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \leq 1$

*Τότε έχουμε ότι:*

$$\begin{aligned} \frac{x}{y} &\leq \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} \cdot (1 + \varepsilon) \text{ (λήμμα 3.2)} \\ &\leq \frac{1}{r} \cdot (1 + \varepsilon) \end{aligned}$$

*και άρα πληρούνται οι προϋποθέσεις.*

**Υπο-περίπτωση 1.2.** $\frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} > 1$

*Εδώ μπορούμε να πούμε:*

$$\begin{aligned} 1 &< \frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \\ &\leq \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} \cdot \frac{1}{1 - \varepsilon} \text{ (λήμμα 3.2)} \\ &\leq \frac{1}{r} \cdot \frac{1}{1 - \varepsilon} \end{aligned}$$

*το οποίο δεν ισχύει λόγο του ότι βρισκόμαστε στην περίπτωση α. και άρα καταλήξαμε σε άτοπο.*

**Περίπτωση 2.** $a_m > r \cdot \sum_{i \in S_{1Opt}} a_i$

*Τότε ισχύει ότι το* $m \in S_{2Opt}$ *και μάλιστα θα πρέπει να είναι μοναδικό γιατί αλλιώς τα* $S_{1Opt}$ *και* $S_{2Opt}$ *δεν θα μας δίναν το βέλτιστο λόγο. Τότε όταν τρέχουμε τον Sub-Algorithm 3.2 με είσοδο* $A^{m-1}$ *για κάποιο k έχουμε το κελί* $F_k[x, 0] \neq \emptyset$ *όπου* $x = \sum_{i \in S_{1Opt}} a_i^u$ *και επιπλέον από το λήμμα 3.1 ισχύει ότι* $r \cdot \sum_{i \in S_{1Opt}} a_i \geq a_{m-1}$ *άρα για το κελί αυτό έχουμε* $r \cdot \sum_1 \geq r \cdot \sum_{i \in S_{1Opt}} a_i \geq a_{m-1}$*. Οπότε μένει να δείξουμε ότι σε εκείνη την κλήση έχουμε* $\frac{x}{C} \geq lim$ *όπου* $lim = \frac{1}{r} \cdot (1 + \varepsilon)$ *και* $C = \lfloor \frac{a_m}{K_{m-1}} \rfloor$ *(στη συγκεκριμένη επανάληψη). Για να το δείξουμε θα ακολουθήσουμε την ίδια διαδικασία με πριν:*

$$\begin{aligned} \frac{x}{C} &= \frac{\sum_{i \in S_{1Opt}} a_i^u}{\lfloor \frac{a_m}{K_{m-1}} \rfloor} = \frac{\sum_{i \in S_{1Opt}} a_i^u}{\sum_{j \in S_{2Opt}} a_j^l} \\ &\leq \frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \text{ (λήμμα 2.2)} \end{aligned}$$

*Και οι υποπεριπτώσεις αποδεικνύονται όμοια με αυτές της περίπτωσης 1, οπότε συνοπτικά:*

**Υπο-περίπτωση 2.1.** $\frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \leq 1$

$$\begin{aligned} \frac{x}{C} &\leq \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} \cdot (1 + \varepsilon) \text{ (λήμμα 3.2)} \\ &\leq \frac{1}{r} \cdot (1 + \varepsilon) \end{aligned}$$

*και άρα πληρούνται οι προϋποθέσεις.*

**Υπο-περίπτωση 2.2.** $\frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} > 1$

*Εδώ μπορούμε να πούμε:*

$$\begin{aligned} 1 &< \frac{\sum_{i \in S_{1Opt}} a_i + n_{1Opt} \cdot K_m}{\sum_{j \in S_{2Opt}} a_j - n_{2Opt} \cdot K_m} \\ &\leq \frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} \cdot \frac{1}{1 - \varepsilon} \text{ (λήμμα 3.2)} \\ &\leq \frac{1}{r} \cdot \frac{1}{1 - \varepsilon} \end{aligned}$$

*όπου όπως και πριν αυτό είναι* ***άτοπο*** *στην περίπτωση α.*

Οπότε ο αλγόριθμος είτε επιστρέφει το βέλτιστο λόγο είτε τα $S_{1Opt}, S_{2Opt}$ πληρούν τις προϋποθέσεις επιλογής για κάποιο m. □

**Θεώρημα 3.1.** *Για τα σύνολα $S_1$ και $S_2$ που επιστρέφει ο αλγόριθμος ισχύει:*

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \in \Big[(1-\varepsilon)\cdot\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{1}{r}\cdot(1+\varepsilon)\Big]$$

*Απόδειξη.* Αν τα $S_1$ και $S_2$ είναι τα βέλτιστα τότε προφανώς ο λόγος ανήκει στο διάστημα. Οπότε εξετάζουμε την περίπτωση που δεν επιστρέφει τα βέλτιστα και άρα έχουμε:

$$\begin{aligned}\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} &\leq \frac{\sum_{i\in S_1} a_i^u}{\sum_{j\in S_2} a_j^l} \text{ (λήμμα 2.2)}\\ &\leq \frac{1}{r}\cdot(1+\varepsilon) \text{ (λόγο κριτηρίων στον αλγόριθμο)}\end{aligned}$$

και άρα μας μένει να δείξουμε το κάτω όριο. Από το λήμμα 3.3 ξέρουμε ότι για κάποιο m τα $S_{1Opt}, S_{2Opt}$ πληρούν τις προϋποθέσεις επιλογής. Έστω $S_1^m$ και $S_2^m$ τα σύνολα που επιστρέφει ο αλγόριθμος σε εκείνη την κλήση του Sub-Algorithm 3.2, τότε έχουμε:

$$1 \geq \frac{1}{r}\cdot(1+\varepsilon) \geq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l} \geq \frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j}$$

και άρα:

$$\begin{aligned}1 \geq \frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} &= \frac{K_m\cdot\sum_{i\in S_1^m} a_i}{K_m\cdot\sum_{j\in S_2^m} a_j} \text{ (λήμμα 2.2)}\\ &\geq \frac{\sum_{i\in S_1^m} a_i^u - n_1}{\sum_{j\in S_2^m} a_j^l + n_2} \text{ (λήμμα 2.2)}\end{aligned}$$

όπου με $n_1$, $n_2$ συμβολίζουμε τις πληθικότητες των συνόλων. Αφού ο λόγος είναι μικρότερος του 1 αν αφαιρέσω την ίδια ποσότητα από αριθμητή και παρονομαστή θα μικρύνει επιπλέον οπότε:

$$\begin{aligned}1 \geq \frac{\sum_{i\in S_1^m} a_i^u - n_1 - n_2}{\sum_{j\in S_2^m} a_j^l} &= \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l}\cdot\Big(1-\frac{n_1+n_2}{\sum_{i\in S_1^m} a_i^u}\Big)\\ \geq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l}\cdot\Big(1-\frac{(n_1+n_2)\cdot K_m}{\sum_{i\in S_1^m} a_i}\Big) &\text{ (λήμμα 2.2)}\end{aligned}$$

Επιπλέον αφού ο αλγόριθμος επιστρέφει σύνολα τέτοια ώστε $r\cdot\sum_{i\in S_1^m} a_i \geq a_m$ έχουμε:

$$\begin{aligned}1 \geq \frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} &\geq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l}\cdot\Big(1-\frac{(n_1+n_2)\cdot a_m\cdot\varepsilon}{2\cdot r\cdot m\sum_{i\in S_1^m} a_i}\Big)\\ \geq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l}\cdot\Big(1-\frac{(n_1+n_2)\cdot\varepsilon}{2\cdot m}\Big) &\geq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l}\cdot(1-\varepsilon)\end{aligned}$$

Τέλος αφού τα $S_{1Opt}, S_{2Opt}$ πληρούν τις προϋποθέσεις επιλογής και από τα σύνολα που επιστρέφει ο Sub-Algorithm 3.2 κρατάμε αυτά που μεγιστοποιούν το λόγο τους είναι προφανές ότι:

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \geq \frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j} \geq \frac{\sum_{i\in S_1^m} a_i^u}{\sum_{j\in S_2^m} a_j^l} \cdot (1-\varepsilon)$$
$$\geq \frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \cdot (1-\varepsilon) \geq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \cdot (1-\varepsilon) \text{ (λήμμα 3.2)}$$

□

*Εδώ καλό είναι να τονιστεί ότι η επιστροφή του min στις γραμμές 22 και 24 του Sub-Algorithm 3.2 δεν κάνει κάτι γιατί* $\frac{1}{r} \cdot (1+\varepsilon) \leq \frac{1}{r} \cdot \frac{1}{1-\varepsilon} \leq 1$.

**Περίπτωση b.** $\frac{1}{r} \cdot (1+\varepsilon) \leq 1 < \frac{1}{r} \cdot \frac{1}{1-\varepsilon}$

*Εδώ όταν ξεκινάει ο αλγόριθμος αλλάζει την τιμή του* $\varepsilon$ *σε* $\hat{\varepsilon} = \frac{\varepsilon}{r}$. *Για το καινούριο* $\hat{\varepsilon}$ *ισχύει ότι ακριβώς και στην προηγούμενη περίπτωση αρκεί να δείξουμε:*

$$\frac{1}{r} \cdot \frac{1}{1-\hat{\varepsilon}} \leq 1$$

*το οποίο ισχύει αφού:*

$$\frac{1}{r} \cdot \frac{1}{1-\hat{\varepsilon}} \leq 1 \Leftrightarrow \frac{1}{r - r\cdot\hat{\varepsilon}} \leq 1 \Leftrightarrow \frac{1}{r-\varepsilon} \leq 1 \Leftrightarrow$$
$$1 \leq r - \varepsilon \Leftrightarrow 1 + \varepsilon \leq r \Leftrightarrow$$
$$\frac{1}{r} \cdot (1+\varepsilon) \leq 1$$

*Να τονίσουμε επιπλέον ότι αφού ισχύουν όλα για το* $\hat{\varepsilon}$ *έχω μεγαλύτερη ακρίβεια από αυτή που μου προσδιορίζει το* $\varepsilon$ *(αλλά αυτό δεν είναι πρόβλημα).*

**Περίπτωση c.** $\frac{1}{r} \cdot (1+\varepsilon) > 1$

*Η απόδειξη θα ακολουθήσει το ίδιο σκεπτικό με τις προηγούμενες περιπτώσεις.*

**Λήμμα 3.4.** *Αν* $A = \{a_1, a_2, ...a_n\}$ *και τα σύνολα που δίνουν τον βέλτιστο λόγο τότε υπάρχει κάποιο* $m \leq n$ *για το οποίο τα* $S_{1Opt}, S_{2Opt}$ *πληρούν τις προϋποθέσεις επιλογής από τον Sub-Algorithm 3.2 ή ο αλγόριθμος επιστρέφει ακριβώς τα* $S_{1Opt}$ *και* $S_{2Opt}$.

*Απόδειξη.* Αν τα σύνολα έχουν μέγιστο στοιχείο m τέτοιο ώστε $m < m_0$ τότε αφού τρέχουμε τον ακριβή αλγόριθμο θα επιστρέψει ακριβώς τα $S_{1Opt}$ και $S_{2Opt}$. Αν δεν ισχύει αυτό τότε θα χωρίσουμε σε 2 περιπτώσεις:

**Περίπτωση 1.** $a_m \leq r \cdot \sum_{i\in S_{1Opt}} a_i$

*Τότε στην κλήση του Sub-Algorithm 3.2 για το στιγμιότυπο* $A^m$ *το αντίστοιχο κελί είναι μη κενό. Επιπλέον αφού* $a_m \leq r \cdot \sum_{i\in S_{1Opt}} a_i$ *σημαίνει ότι το* $S_{2Opt}$

*δεν είναι μονοσύνολο (αλλιώς ο λόγος δεν είναι ο βέλτιστος) και άρα πληρούνται οι προϋποθέσεις του λήμματος 3.2 και άρα και της εξίσωσης 3.2 άρα:*

$$\frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \leq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \cdot \frac{1}{1-\varepsilon} \textit{(εξ. 3.2)}$$

**Περίπτωση 2.** $a_m > r \cdot \sum_{i\in S_{1Opt}} a_i$

*Από το λήμμα 3.1 έχουμε* $r \cdot \sum_{i\in S_{1Opt}} a_i \geq a_{m-1}$ *και το* $S_{2Opt} = m$*(μονοσύνολο) γιατί αλλιώς δεν θα είχα βέλτιστο λόγο άρα στην κλήση του Sub-Algorithm 3.2 για το στιγμιότυπο* $A^{m-1}$ *πληρούνται οι προϋποθέσεις επιλογής και της εξίσωσης 3.2 οπότε:*

$$\frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \leq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \cdot \frac{1}{1-\varepsilon} \textit{(εξ. 3.2)}$$

Τέλος αφού και στις δύο περιπτώσεις έχουμε $\frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \leq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \cdot \frac{1}{1-\varepsilon}$ και $\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \leq \frac{1}{r} \leq r$ είναι προφανές ότι:

$$\frac{\sum_{i\in S_{1Opt}} a_i^u}{\sum_{j\in S_{2Opt}} a_j^l} \leq r \cdot \frac{1}{1-\varepsilon}$$

□

**Θεώρημα 3.2.** *Για τα σύνολα* $S_1$ *και* $S_2$ *που επιστρέφει ο αλγόριθμος ισχύει:*

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \in \Big[(1-\varepsilon) \cdot \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{1}{r} \cdot (1+\varepsilon)\Big]$$

*Απόδειξη.* Αν τα $S_1$ και $S_2$ είναι τα βέλτιστα τότε προφανώς ο λόγος ανήκει στο διάστημα. Αν δεν είναι τα βέλτιστα τότε αφού ο αλγόριθμος επιστρέφει το ελάχιστο εκ των $\frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j}$ , $\frac{\sum_{j\in S_2^m} a_j}{\sum_{i\in S_1^m} a_i}$ σε κάθε κλήση του Sub-Algorithm 3.2 σημαίνει ότι για την επανάληψη που επέλεξε τα $S_1$ και $S_2$ (έστω m) ισχύει:

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} = min\{\frac{\sum_{i\in S_1^m} a_i}{\sum_{j\in S_2^m} a_j}, \frac{\sum_{j\in S_2^m} a_j}{\sum_{i\in S_1^m} a_i}\} \leq 1 < \frac{1}{r} \cdot (1+\varepsilon)$$

Άρα μένει να δείξουμε το κάτω όριο.

Τώρα θα χωρίσουμε την απόδειξη σε δύο περιπτώσεις:

**Περίπτωση 1.** *Αν ο λόγος ήταν μεγαλύτερος του 1 και τον αντιστρέψαμε.*

*Τότε πριν αντιστρέψουμε είχαμε:*

$$1 \leq \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} a_i} \leq \frac{\sum_{j\in S_2} a_j^u}{\sum_{i\in S_1} a_i^l} \leq r \cdot \frac{1}{1-\varepsilon}$$

*Οπότε αντιστρέφοντας:*

$$\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \geq \frac{1}{r}\cdot(1-\varepsilon) \geq \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}\cdot(1-\varepsilon)$$

**Περίπτωση 2.** *Αν ο λόγος ήταν μικρότερος του 1.*

*Τότε η απόδειξη είναι ίδια με αυτή της περίπτωσης α. οπότε και παραλείπεται.*

□

Μας μένει να σχολιάσουμε την πολυπλοκότητα του αλγορίθμου. Μπορούμε εύκολα να δούμε ότι ο πίνακας F στην χειρότερη περίπτωση χρειάζεται χρόνο $O(\frac{n^6\cdot r^4}{\varepsilon^2})$ (επειδή το $\varepsilon$ μπορεί να πάρει την τιμή $\frac{\varepsilon}{r}$) για να γεμίσει και αφού κατασκευάζουμε το πολύ n τέτοιους τελική πολυπλοκότητα είναι $O(\frac{n^7\cdot r^4}{\varepsilon^2})$.

# Κεφάλαιο 4

# Γενίκευση αποτελεσμάτων

Σε αυτό το κεφάλαιο θα αποδείξουμε ένα θεώρημα, το οποίο θα χρησιμοποιούμε κατά κόρων παρακάτω, για να αποδείξουμε ότι επιστρέφουμε λόγους με ακρίβεια $1+\varepsilon$ κοντά στη βέλτιστη λύση. Επιπλέον θα μελετήσουμε την ύπαρξη FPTAS αλγορίθμων στην κλάση προβλημάτων επιλογής τουλάχιστον δύο υποσυνόλων.

## 4.1 Βασικό θεώρημα

Το θεώρημα το οποίο αναπτύξαμε είναι γενίκευση της απόδειξης ορθότητας του αλγορίθμου που υπάρχει στο [18]. Αυτό το θεώρημα είναι πολύ βασικό καθώς όλες οι αποδείξεις ορθότητας των αλγορίθμων που θα παρουσιάσουμε από εδώ και κάτω θα βασίζονται σε αυτό. Το θεώρημα είναι το ακόλουθο:

**Θεώρημα 4.1.** *Έστω ότι έχουμε ένα σύνολο θετικών αριθμών* $A = \{a_1, a_2, ..., a_n\}$ *και* $S_{1Opt}, S_{2Opt} \subseteq \{1, 2, ..., n\}$ *τα σύνολα που ψάχνουμε. Έστω επιπλέον* $\delta = \frac{\varepsilon \cdot w}{3 \cdot m}$ *ένας θετικός αριθμός για τον οποίο ισχύει ότι* $\varepsilon \in (0, 1)$, $w \leq \min\{\sum_{i \in S_{1Opt}} a_i, \sum_{j \in S_{2Opt}} a_j\}$ *και* $m \geq \max\{|S_{1Opt}|, |S_{2Opt}|\}$. *Αν για δύο μη κενά σύνολα* $S_1, S_2 \subseteq \{1, 2, ..., n\}$ *ισχύει ότι:*

1. $n \geq m \geq \max\{|S_1|, |S_2|\}$
2. $w \leq \min\{\sum_{i \in S_1} a_i, \sum_{j \in S_2} a_j\}$
3. $1 \leq \max\left\{\frac{\sum_{i \in S_1} a'_i}{\sum_{j \in S_2} a'_j}, \frac{\sum_{j \in S_2} a'_j}{\sum_{i \in S_1} a'_i}\right\} \leq \max\left\{\frac{\sum_{i \in S_{1Opt}} a'_i}{\sum_{j \in S_{2Opt}} a'_j}, \frac{\sum_{j \in S_{2Opt}} a'_j}{\sum_{i \in S_{1Opt}} a'_i}\right\}$

*όπου* $a'_i = \lfloor \frac{a_i}{\delta} \rfloor$, *τότε έχουμε:*

$$1 \leq \max\left\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}, \frac{\sum_{j \in S_2} a_j}{\sum_{i \in S_1} a_i}\right\} \leq \max\left\{\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}, \frac{\sum_{j \in S_{2Opt}} a_j}{\sum_{i \in S_{1Opt}} a_i}\right\} \cdot (1 + \varepsilon)$$

Η ανάγκη για αυτό το θεώρημα προέκυψε από το γεγονός πως για κάθε πρόβλημα κάναμε και διαφορετική απόδειξη η οποία όμως ακολουθούσε ακριβώς το

ίδιο σκεπτικό. Να παρατηρήσουμε ότι τα $S_{1Opt}$, $S_{2Opt}$ ουσιαστικά θα είναι αυτά που μας δίνουν τους βέλτιστους λόγους των προβλημάτων αλλά δεν χρειάζεται να περιορίσουμε το θεώρημα. Το θεώρημα είναι αρκετά γενικό καθώς δεν επιβάλει στις τιμές $a_i$ να είναι ακέραιες αλλά δεν έχει και κανένα περιορισμό στα σύνολα $S_{1Opt}$, $S_{2Opt}$ πράγμα που μας δίνει την δυνατότητα να κάνουμε χρίση του θεωρήματος σε πολύ διαφορετικά προβλήματα. Οι αλλαγές σε σχέση με την απόδειξη ορθότητας του [18] είναι πρώτον πως η εδώ έχουμε πραγματικές τιμές, δεύτερον ότι γενικεύουμε την τιμή του $\delta$ μέσω του $w$ και του $m$ (τα οποία στην αρχική απόδειξη ήταν κάποιο στοιχείο και το n, μέγεθος του $|A|$, αντίστοιχα) και τρίτον μέσω της τρίτης συνθήκης στο θεώρημα γενικεύουμε την ακρίβεια από $(1+\varepsilon)$ κοντά στο βέλτιστο σε $(1+\varepsilon)$ κοντά σε κάποιο επιθυμητό.

## 4.2 Απόδειξη

Θα ξεκινήσουμε την απόδειξη με ένα λήμμα που μας δίνει μια συσχέτιση των $a_i$ και $a'_i$.

**Λήμμα 4.1.** *Για τα $a_i$, $a'_i$ και $\delta$ ισχύουν τα ακόλουθα:*

$$a_i/\delta - 1 \leq a'_i \leq a_i/\delta \text{ , } \forall 1 \leq i \leq n \tag{4.1}$$

$$m \cdot \delta \leq \frac{\varepsilon}{3} \sum_{i \in S} a_i \text{ , } \forall S \in \{S_1, S_2, S_{1Opt}, S_{2Opt}\} \tag{4.2}$$

*Απόδειξη.* Η απόδειξη τις πρώτης ανίσωσης είναι προφανής εκ' του ορισμού των $a'_i$ ενώ για την δεύτερη ισχύει ότι:

$$m \cdot \delta = \frac{\varepsilon \cdot w}{3} \leq \frac{\varepsilon}{3} \sum_{i \in S} a_i \text{ (εξ' υποθέσεως)}$$

□

Για να διευκολυνθούμε εδώ θα κάνουμε κάποιες αλλαγές στον συμβολισμό δηλαδή, αν $\sum_{i \in S_{1Opt}} a'_i \geq \sum_{j \in S_{2Opt}} a'_j$ τότε ο $A_{Opt} = S_{1Opt}$ και $B_{Opt} = S_{2Opt}$ αλλιώς $A_{Opt} = S_{2Opt}$ και $B_{Opt} = S_{1Opt}$ και ομοίως για τα $S_1$, $S_2$ $A = S_1$ και $B = S_2$ αν $\sum_{i \in S_1} a'_i \geq \sum_{j \in S_2} a'_j$ αλλιώς $A = S_2$ και $B = S_1$. Με αυτό το συμβολισμό η υπόθεση "3" του θεωρήματος μπορεί να γραφτεί ως εξής:

$$1 \leq \frac{\sum_{i \in A} a'_i}{\sum_{j \in B} a'_j} \leq \frac{\sum_{i \in A_{Opt}} a'_i}{\sum_{j \in B_{Opt}} a'_j}$$

**Λήμμα 4.2.** *Για τα σύνολα $A$ και $B$ έχουμε:*

$$\max\left\{\frac{\sum_{i \in A} a_i}{\sum_{j \in B} a_j}, \frac{\sum_{j \in B} a_j}{\sum_{i \in A} a_i}\right\} \leq \frac{\sum_{i \in A} a'_i}{\sum_{j \in B} a'_j} + \frac{\varepsilon}{3}$$

*Απόδειξη.* Θα δείξουμε την ανίσωση για κάθε ένα από τους δυο λόγους.

$$\frac{\sum_{i\in A} a_i}{\sum_{j\in B} a_j} \overset{\text{σχ.4.1}}{\leq} \frac{\delta \cdot \sum_{i\in A} a'_i + m\cdot\delta}{\sum_{j\in B} a_j} = \frac{\delta \cdot \sum_{i\in A} a'_i}{\sum_{j\in B} a_j} + \frac{m\cdot\delta}{\sum_{j\in B} a_j}$$
$$\overset{\text{σχ.4.1}}{\leq} \frac{\sum_{i\in A} a'_i}{\sum_{j\in B} a'_j} + \frac{m\cdot\delta}{\sum_{j\in B} a_j} \overset{\text{σχ.4.2}}{\leq} \frac{\sum_{i\in A} a'_i}{\sum_{j\in B} a'_j} + \frac{\varepsilon}{3}$$

Και όμοια για το αντίστροφο έχουμε:

$$\frac{\sum_{j\in B} a_j}{\sum_{i\in A} a_i} \leq \frac{\sum_{j\in B} a'_j}{\sum_{i\in A} a'_i} + \frac{\varepsilon}{3} \leq \frac{\sum_{i\in A} a'_i}{\sum_{j\in B} a'_j} + \frac{\varepsilon}{3}$$

όπου η τελευταία ισχύει από το θέσμιο των Α και Β που κάναμε πριν το λήμμα. □

**Λήμμα 4.3.** *Για κάθε* $\varepsilon \in (0,1)$ *έχουμε* $\frac{\sum_{i\in A_{Opt}} a'_i}{\sum_{j\in B_{Opt}} a'_j} \leq (1+\varepsilon)\cdot\frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j}$

*Απόδειξη.*

$$\begin{aligned}
\frac{\sum_{i\in A_{Opt}} a'_i}{\sum_{j\in B_{Opt}} a'_j} &\overset{\text{σχ.4.1}}{\leq} \frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j - m\cdot\delta} = \frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j - m\cdot\delta}\cdot\frac{\sum_{j\in B_{Opt}} a_j}{\sum_{j\in B_{Opt}} a_j} \\
&= \frac{\sum_{j\in B_{Opt}} a_j}{\sum_{j\in B_{Opt}} a_j - m\cdot\delta}\cdot\frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j} \\
&= (1+\frac{m\cdot\delta}{\sum_{j\in B_{Opt}} a_j - m\cdot\delta})\cdot\frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j} \\
&= (1+\frac{1}{\frac{\sum_{j\in B_{Opt}} a_j}{m\cdot\delta} - 1})\cdot\frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j} \\
&\overset{\text{σχ.4.2}}{\leq} (1+\frac{1}{\frac{3}{\varepsilon} - 1})\cdot\frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j} \\
&= (1+\frac{\varepsilon}{3-\varepsilon})\cdot\frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j} \leq (1+\frac{\varepsilon}{2})\cdot\frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j}
\end{aligned}$$

□

Μετά από αυτό είμαστε έτοιμοι να αποδείξουμε την σχέση του θεωρήματος.

*Απόδειξη.* (Θεωρηματος 4.1)

$$
\begin{aligned}
1 \leq \max\Big\{\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} a_i}\Big\} &= \frac{\sum_{i\in A} a_i}{\sum_{j\in B} a_j} \text{ , από ορισμό A,B} \\
&\leq \frac{\sum_{i\in A} a'_i}{\sum_{j\in B} a'_j} + \frac{\varepsilon}{3} \text{ , από λήμμα 4.2} \\
&\leq \frac{\sum_{i\in A_{Opt}} a'_i}{\sum_{j\in B_{Opt}} a'_j} + \frac{\varepsilon}{3} \text{ , από υπόθεση} \\
&\leq (1+\frac{\varepsilon}{2}) \cdot \frac{\sum_{i\in A_{Opt}} a_i}{\sum_{j\in B_{Opt}} a_j} + \frac{\varepsilon}{3}
\end{aligned}
$$

Αν για ευκολία συμβολίσουμε $\lambda_{Opt} = max\Big\{\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{j\in S_{2Opt}} a_j}{\sum_{i\in S_{1Opt}} a_i}\Big\}$ τότε από τον ορισμό των $A_{Opt}$, $B_{Opt}$ έχουμε:

$$
\begin{aligned}
1 \leq \max\Big\{\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} a_i}\Big\} &\leq (1+\frac{\varepsilon}{2}) \cdot \lambda_{Opt} + \frac{\varepsilon}{3} \\
&\leq (1+\frac{\varepsilon}{2})\lambda_{Opt} + \frac{\varepsilon}{3} \cdot \lambda_{Opt} \\
&\leq (1+\frac{\varepsilon}{2}+\frac{\varepsilon}{3}) \cdot \lambda_{Opt} \\
&\leq (1+\varepsilon) \cdot \lambda_{Opt}
\end{aligned}
$$

Και άρα αποδείξαμε τη σχέση του θεωρήματος. □

# Κεφάλαιο 5

# Εναλλακτική Προσέγγιση

## 5.1 Η μέθοδος του μεγαλύτερου στοιχείου

Εδώ θα ορίσουμε ένα παρόμοιο, με όσα είδαμε, πρόβλημα με σκοπό να χρησιμοποιήσουμε παραλλαγές του αλγορίθμου που θα το λύνει για να βελτιώσουμε την πολυπλοκότητα, την περιγραφή και την απόδειξη των αλγορίθμων που αναπτύξαμε. Το πρόβλημα αυτό είναι μια παραλλαγή του ESS αλλά μας υποχρεώνει να κάνουμε χρήση του μεγαλύτερου στοιχείου.

**Πρόβλημα 8** (ESS with use of greatest integer problem)**.** *Δοθέντος ενός συνόλου από n ακεραίους* $A = \{a_1, a_2, ..., a_n\}$ *με μέγιστο στοιχείο* $a_n$*, υπάρχουν δύο μη κενά και ξένα σύνολα* $S_1$*,* $S_2 \subseteq 1, 2, ..., n$ *τέτοια ώστε* $\sum_{i \in S_1} a_i = \sum_{j \in S_2} a_i$ *και* $n \in S_1 \cup S_2$*;*

Η ιδέα αυτή σε συνδυασμό με το λήμμα 2.1 θα μας δώσει μια βελτιωμένη έκδοση του αλγορίθμου που υπάρχει στο [18] όχι μόνο για το SSR αλλά και για τα υπόλοιπα προβλήματα που ασχοληθήκαμε μέχρι τώρα (κάνοντας χρήση αντίστοιχων λημμάτων). Η βελτίωση που υπάρχει σε σχέση με το [18] είναι ότι, λόγο του λήμματος 2.1, έχουμε ένα κάτω φράγμα για τα σύνολα και άρα δεν χρειάζεται να τρέχει ο αλγόριθμος για όλα τα ζεύγη μεγίστων αλλά μόνο για το μέγιστο στοιχείο. Όπως και στα προηγούμενα κεφάλαια δεν θα λύνουμε το ακριβές πρόβλημα αλλά το ακόλουθο πρόβλημα βελτιστοποίησης:

**Πρόβλημα 9** (SSR with use of greatest integer optimization problem)**.** *Δοθέντος ενός συνόλου από n ακεραίους* $A = \{a_1, a_2, ..., a_n\}$ *με μέγιστο στοιχείο* $a_n$*, ψάχνουμε δύο μη κενά και ξένα σύνολα* $S_1$*,* $S_2 \subseteq 1, 2, ..., n$ *τέτοια ώστε να ελαχιστοποιείτε ο λόγος* $\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$ *με* $\sum_{i \in S_1} a_i \geq \sum_{j \in S_2} a_j$ *και* $n \in S_1 \cup S_2$*.*

Να παρατηρήσουμε ότι εδώ, σε αντίθεση με τα προβλήματα που έχουμε προσεγγίσει μέχρι τώρα, ελαχιστοποιούμε το λόγο των αθροισμάτων και αυτό θα συνεχίσουμε να κάνουμε στους αλγορίθμους που θα αναπτύξουμε παρακάτω.

### 5.1.1 Ψευδοπολυωνυμικός Αλγόριθμος

Εδώ θα αναπτύξουμε ένα ψευδοπολυωνυμικό αλγόριθμο ο οποίος, για ένα ταξινομημένο σύνολο $A = \{a_1, a_2, ..., a_n\}$, θα μας λύνει το προηγούμενο πρόβλημα βελτιστοποίησης. Αυτό θα το καταφέρουμε κατασκευάζοντας ένα πίνακα $F_k[x, y]$ στα κελιά του οποίου θα κρατάμε δύο σύνολα τέτοια ώστε $F_k[x, y] = (S_1, S_2)$ αν $\sum_{i \in S_1} a_i = x$, $\sum_{j \in S_2} a_j = y$ και $max\{(S_1 \cup S_2) \setminus n\} \leq k$ ενώ πάντα $max\{S_1 \cup S_2\} = n$.

**Algorithm 5.1** Solve SSR with use of greatest integer optimization problem

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$

1: **sort** $A$
2: **for all** $0 \leq k \leq n$, $0 \leq x \leq \sum_{i=1}^{n} a_i$ **and** $0 \leq y \leq \sum_{i=1}^{n} a_i$ **do**
3: $F_k[x, y] \leftarrow \vec{\emptyset}$
4: **end for**
5: $F_0[0, a_n] \leftarrow (\emptyset, \{n\})$
6: $F_0[a_n, 0] \leftarrow (\{n\}, \emptyset)$
7: **for** $k = 1$ to $n$ **do**
8: **for all** $x, y \leq \sum_{i=1}^{n} a_i$ **do**
9: **if** $F_{k-1}[x, y] \neq \vec{\emptyset}$ **then**
10: $(S_1, S_2) \leftarrow F_{k-1}[x, y]$
11: $F_k[x, y] \leftarrow (S_1, S_2)$
12: **if** $k < n$ **then**
13: $F_k[x + a_k, y] \leftarrow (S_1 \cup \{k\}, S_2)$
14: $F_k[x, y + a_k] \leftarrow (S_1, S_2 \cup \{k\})$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: **for all** $F_n[x, y] \neq \vec{\emptyset}$ **and** $\frac{x}{y} \geq 1$ **do**
20: **find** $\frac{x_0}{y_0}$ = **min** $\frac{x}{y}$
21: **end for**
22: **return** $F_n[x_0, y_0]$

Να αναφέρουμε ότι σε αντίθεση με τους προηγούμενους αλγορίθμους η μέθοδος γεμίσματος του πίνακα F είναι κατά μία τάξη του n γρηγορότερη. Ακριβέστερα στα προηγούμενα για να γεμίσει ένα κελί έπρεπε να ελέγξει το πολύ n κελιά (ο δείκτης k του πίνακα παίρνει τιμές μικρότερες ίσες του n) ενώ εδώ δεν χρειάζεται. Επιπλέον από εδώ και κάτω θα αποφεύγουμε να γράφουμε αναλυτικά την επιλογή μεγίστων αλλά θα το κάνουμε όπως στην γραμμή 20.

## 5.2 Χρήση της μεθόδου στο SSR

### 5.2.1 Ψευδοπολυωνυμικός Αλγόριθμος

Πριν ξεκινήσουμε να περιγράψουμε τον αλγόριθμο για το SSR καλό είναι να παρατηρήσουμε ότι αν το σύνολο $A = \{a_1, a_2, ..., a_n\}$ είναι ταξινομημένο σε αύξουσα και $S_1$, $S_2$ τα υποσύνολα του A που βελτιστοποιούν το SSR τότε τα $S_1$, $S_2$ είναι αυτά που βελτιστοποιούν και το ESS with use of greatest integer για το σύνολο $A' = \{a_1, a_2, ..., a_k\}$ όπου k προφανώς το μέγιστο στοιχείο των $S_1$, $S_2$. Αυτή η διαπίστωση μας οδηγεί στον εξής αλγόριθμο για το SSR:

**Algorithm 5.2** Solve SSR

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$

1: **sort** $A$
2: **for** $i = 2$ to $n$ **do**
3: $A' \leftarrow \{a_1, ..., a_i\}$
4: **run** Algorithm 5.1 with **input** $A'$ and **output** ($S_1^i$, $S_2^i$, $\frac{x_i}{y_i}$)
5: **end for**
6: **find** $k$ such that $\frac{x_k}{y_k}$ = **min** $\frac{x_i}{y_i}$
7: $S_1 \leftarrow S_1^k$, $S_2 \leftarrow S_2^k$
8: **return** $S_1$, $S_2$ and $\frac{x_k}{y_k}$

Η ορθότητα αυτού του αλγορίθμου είναι προφανής λόγο της παρατήρησης που κάναμε παραπάνω.

### 5.2.2 FPTAS

Ο προηγούμενος αλγόριθμος δεν είναι πολυονυμικού χρόνου καθώς εξαρτάτε από το συνολικό άθροισμα των στοιχείων. Η μέθοδος με την οποία θα φράξουμε το συνολικό άθροισμα μοιάζει με αυτή των προηγούμενων αλγορίθμων και είναι βασισμένη στο λήμμα 2.1. Για ακρίβεια αντί να χρησιμοποιούμε το μέγιστο στοιχείο στο $K_n$ θα κάνουμε χρήση του δευτέρου μεγαλύτερου. Όπως και στα προηγούμενα θα χωρίσουμε τον αλγόριθμο σε μικρότερα κομμάτια για ευκολία. Η ακριβής διαδικασία φαίνεται παρακάτω

**Sub-Algorithm 5.1** FPTAS for RSS with use of greatest integer

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ and a parameter $\varepsilon \in (0, 1)$

1: **sort** $A$
2: $\delta \leftarrow \frac{a_{n-1} \cdot \varepsilon}{3 \cdot n}$
3: $a_i' \leftarrow \lfloor \frac{a_i}{\delta} \rfloor$

**if** $\sum_{i=1}^{n-1} a'_i < a'_n$ **then**
  **if** $\sum_{i=1}^{n-1} a_i < a_n$ **then**
    $S_1 \leftarrow \{n\}$, $S_2 \leftarrow \{1, 2, ..., n-1\}$
    **return** ($S_1$, $S_2$, $a_n$, $\sum_{i=1}^{n-1} a_i$)
  **else**
    $S_1 \leftarrow \{1, 2, ..., n-1\}$, $S_2 \leftarrow \{n\}$
    **return** ($S_1$, $S_2$, $\sum_{i=1}^{n-1} a_i$, $a_n$)
  **end if**
**else**
  **for all** $0 \leq k \leq n$ **and** $0 \leq x \leq y \leq \sum_{i=1}^{n} a'_i$ **do**
    $F_k[x, y] \leftarrow \vec{\emptyset}$
  **end for**
  $F_0[0, a'_n] \leftarrow (\emptyset, \{n\}, 0, a_n)$
  **for** $k = 1$ to $n$ **do**
    **for all** $x, y \leq \sum_{i=1}^{n} a'_i$ **do**
      **if** $F_{k-1}[x, y] \neq \vec{\emptyset}$ **then**
        $(S_1, S_2, sum_1, sum_2) \leftarrow F_{k-1}[x, y]$
        $F_k[x, y] \leftarrow (S_1, S_2, sum_1, sum_2)$
        (if $F_k[x, y] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
        **if** $k < n$ **then**
          $F_k[x + a'_k, y] \leftarrow (S_1 \cup \{k\}, S_2, sum_1 + a_k, sum_2)$
          (if $F_k[x + a'_k, y] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
          $F_k[x, y + a'_k] \leftarrow (S_1, S_2 \cup \{k\}, sum_1, sum_2 + a_k)$
          (if $F_k[x, y + a'_k] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
        **end if**
      **end if**
    **end for**
  **end for**
  **for all** $F_n[x, y] = (S_1, S_2, sum_1, sum_2) \neq \vec{\emptyset}$ **and** $sum_1 \geq a_{n-1}$ **do**
    **find** $\frac{x_1}{y_1}$ = **min** $\frac{x}{y} \geq 1$
    **find** $\frac{y_2}{x_2}$ = **min** $\frac{y}{x} \geq 1$
  **end for**
  **if** $\frac{x_1}{y_1} \leq \frac{y_2}{x_2}$ **then**
    $x_0 \leftarrow x_1$, $y_0 \leftarrow y_1$
  **else**
    $x_0 =\leftarrow x_2$, $y_0 \leftarrow y_2$
  **end if**
  $(S_1, S_2, sum_1, sum_2) \leftarrow F_n[x_0, y_0]$
  **if** $sum_1 \geq sum_2$ **then**
    **return** $(S_1, S_2, sum_1, sum_2)$
  **else**
    **return** $(S_2, S_1, sum_2, sum_1)$
  **end if**
**end if**

**Algorithm 5.3** FPTAS for RSS

**Require:** set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ and a parameter $\varepsilon \in (0, 1)$

1: **sort** $A$
2: **for** $i = 2$ to $n$ **do**
3: $\quad A' \leftarrow \{a_1, ..., a_i\}$
4: $\quad$ **run** Sub-Algorithm 5.1 with **input** $A'$, $\varepsilon$ and **output** $(S_1^i, S_2^i, sum_1^i, sum_2^i)$
5: **end for**
6: **find** $k$ such that $\frac{sum_1^k}{sum_2^k} = \mathbf{min}_{2 \leq i \leq n} \left\{ \frac{sum_1^i}{sum_2^i} \right\}$
7: $ratio \leftarrow \frac{sum_1^k}{sum_2^k}$
8: **return** $S_1^k$, $S_2^k$, $ratio$

### 5.2.3 Απόδειξη Ορθότητας

Η απόδειξη ορθότητας των αλγορίθμων από εδώ και πέρα θα βασίζεται στο θεώρημα που αποδείξαμε στο προηγούμενο κεφάλαιο. Αυτό θα το κάνουμε δίνοντας ότι αν $n_0 = max\{S_{1Opt} \cup S_{2Opt}\}$ τότε στην επανάληψη όπου $i = n_0$ τα σύνολα και ο λόγος που επιστρέφει ο αλγόριθμος θα πληρούν τις προϋποθέσεις του θεωρήματος.

Θα ξεκινήσουμε με την κλασική παρατήρηση ότι η βέλτιστη λύση του προβλήματος για $A = \{a_1, a_2, ...a_n\}$ είναι ίδια με αυτή για $B = \{a_1, a_2, ..., a_{n_0}\}$ από τον ορισμό του $n_0$ και έστω $S_1$, $S_2$ τα σύνολα που επιστρέφει ο αλγόριθμος στην $n_0$ επανάληψη. Έπειτα θα θυμίσουμε το λήμμα 2.1:

**Λήμμα 2.1.** *Έστω ένα ταξινομημένο σύνολο $A = \{a_1, a_2, ..., a_n\}$, αν $S_{1Opt}$, $S_{2Opt}$ είναι δύο ξένα υποσύνολα του A για τα οποία έχουμε:*

$$\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} = max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \mid \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j} \leq 1 \text{ , } S_1 \cap S_2 = \emptyset\} \text{ και}$$
$$max\{a_i \mid i \in S_{1Opt} \cup S_{2Opt}\} = a_n$$

*τότε ισχύει:*

$$a_{n-1} \leq \sum_{i \in S_{1Opt}} a_i \leq \sum_{j \in S_{2Opt}} a_j$$

Τα σύνολα του βέλτιστου λόγου που περιγράφει είναι τα ίδια για το πρόβλημα που λύνουμε εδώ αντιστρέφοντας τον λόγο (αν υπήρχε μικρότερος λόγος τότε αντιστρέφοντας δεν θα ήταν ο βέλτιστος αυτός του λήμματος). Άρα έχουμε ότι το $a_{n-1}$ (όπου $a_n$ το μέγιστο στοιχείο που χρησιμοποιείτε) είναι μικρότερο από τα δύο αθροίσματα που δίνουν τον βέλτιστο λόγο στο SSR. Άρα για $n_0 = max\{S_{1Opt} \cup S_{2Opt}\}$ ισχύει:

$$a_{n_0-1} \leq \sum_{j \in S_{2Opt}} a_j \leq \sum_{i \in S_{1Opt}} a_i \tag{5.1}$$

(όπου $\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j} \geq 1$ ο βέλτιστος λόγος του προβλήματος όπως το προσεγγίζουμε σε αυτό το κεφάλαιο). Το λήμμα που ακολουθεί ασχολείται με τις συνθήκες για το $\delta$.

**Λήμμα 5.1.** *Στην επανάληψη όπου $i = n_0$ ισχύουν οι σχέσεις του Θεωρήματος 4.1 μεταξύ του $\delta$ και των συνόλων $S_1, S_2, S_{1Opt}, S_{2Opt}$, δηλαδή:*
*1.* $\delta = \frac{\varepsilon \cdot w}{3 \cdot m}$
*2.* $w \leq \sum_{i\in S} a_i, \forall S \in \{S_1, S_2, S_{1Opt}, S_{2Opt}\}$ *και*
*3.* $m \geq |S|, \forall S \in \{S_1, S_2, S_{1Opt}, S_{2Opt}\}$.

*Απόδειξη.* Όταν $i = n_0$ στον αλγόριθμο το $\delta$ έχει τιμή $\frac{\varepsilon \cdot a_{n_0-1}}{3 \cdot n_0}$. Αλλά τότε ισχύουν οι ακόλουθες δύο σχέσεις:

$$1.\ n_0 \geq |S|, \forall S \in \{S_1, S_2, S_{1Opt}, S_{2Opt}\}$$
$$2.\ \sum_{i\in S} a_i \geq a_{n_0-1}, \forall S \in \{S_1, S_2, S_{1Opt}, S_{2Opt}\}$$

όπου το πρώτο είναι προφανές από το ότι $S \subseteq \{1, 2, ..., n_0\}, \forall\, S \in \{S_1, S_2, S_{1Opt}, S_{2Opt}\}$ ενώ το δεύτερο ισχύει από κατασκευή των $S_1$, $S_2$ και από την εξίσωση 5.1 για τα $S_{1Opt}$, $S_{2Opt}$. Τότε μπορώ να θεωρήσω τα $w = a_{n_0-1}$ και $m = n_0$ και άρα το $\delta$ του αλγορίθμου πληρεί τις προϋποθέσεις του Θεωρήματος 4.1. □

Μας μένει να δείξουμε την σχέση μεταξύ των λόγων των αθροισμάτων.

**Λήμμα 5.2.** *Τα $S_1, S_2$ που επιστρέφει ο αλγόριθμος στην $i = n_0$ επανάληψη πληρούν την συνθήκη των λόγων του Θεωρήματος 4.1, δηλαδή:*

$$1 \leq \max\Big\{\frac{\sum_{i\in S_1} a'_i}{\sum_{j\in S_2} a'_j}, \frac{\sum_{j\in S_2} a'_j}{\sum_{i\in S_1} a'_i}\Big\} \leq \max\Big\{\frac{\sum_{i\in S_{1Opt}} a'_i}{\sum_{j\in S_{2Opt}} a'_j}, \frac{\sum_{j\in S_{2Opt}} a'_j}{\sum_{i\in S_{1Opt}} a'_i}\Big\}$$

*Απόδειξη.* Θα διακρίνουμε τις ακόλουθες περιπτώσεις:

**Περίπτωση 1.** *Αν $\sum_{i=1}^{n_0-1} a'_i < a'_{n_0}$:*

*Αφού έχουμε ότι $n_0 = max\{S_{1Opt} \cup S_{2Opt}\}$ και από κατασκευή αλγορίθμου $n_0 = max\{S_1 \cup S_2\}$ τότε τα σύνολα $\{n_0\}$, $\{1, 2, ..., n_0 - 1\}$ που επιστρέφει ο αλγόριθμος μας δίνουν τον μικρότερο λόγο για τις προσαρμοσμένες τιμές με χρίση του $a_{n_0}$ (να υπενθυμίσουμε ότι παίρνουμε λόγους μεγαλύτερους του 1). Τότε ισχύει η σχέση των λόγων του Θεωρήματος 4.1*

**Περίπτωση 2.** *Αν $\sum_{i=1}^{n_0-1} a'_i \geq a'_{n_0}$*

*Ο αλγόριθμος από όλα τα κελιά $F_{n_0}[x, y] = (S_1, S_2, sum_1, sum_2) \neq \vec{\emptyset}$ του πίνακα που έχουν την μεταβλητή $sum_1$ μεγαλύτερη της τιμής $a_{n_0-1}$ επιστρέφει αυτό που η τιμή $\max\{x/y, y/x\}$ είναι ελάχιστη. Αυτό σημαίνει ότι αρκεί να δείξουμε ότι ένα εκ των κελιών $F_{n_0}[\sum_{i\in S_{1Opt}} a'_i, \sum_{j\in S_{2Opt}} a'_j]$, $F_{n_0}[\sum_{j\in S_{2Opt}} a'_j, \sum_{i\in S_{1Opt}} a'_i]$ δεν είναι κενό και έχει τιμή $sum_1 \geq a_{n_0-1}$. Ο αλγόριθμος κρατάει όλους τους συνδυασμούς που μπορούν να δημιουργηθούν (με το $n_0$ στο δεύτερο σύνολο του κελιού)*

*άρα προφανώς ένα εκ των δύο κελιών τουλάχιστον είναι μη κενό. Επιπλέον τα σύνολα τα κρατάμε με το σκεπτικό να μεγιστοποιείτε η τιμή $\sum_1$ και αφού για τα $S_{1Opt}$, $S_{2Opt}$ ισχύει η σχέση 5.1 τότε όποιο και να είναι μη κενό έχει αναγκαστικά $sum_1 \geq a_{n_0-1}$. Άρα πάλι ισχύει η σχέση του Θεωρήματος 4.1.*

□

Τώρα μπορούμε να κάνουμε χρήση του θεωρήματος και να αποδείξουμε ότι ο αλγόριθμος έχει την ακρίβεια που θέλουμε. Αυτό συνοψίζεται στο ακόλουθο θεώρημα.

**Θεώρημα 5.1.** *Ο Algorithm 5.3 επιστρέφει δύο σύνολα $S_1$, $S_2$ για τα οποία ισχύει:*

$$1 \leq \max\left\{\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} a_i}\right\} \leq \max\left\{\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{j\in S_{2Opt}} a_j}{\sum_{i\in S_{1Opt}} a_i}\right\} \cdot (1+\varepsilon)$$

*όπου $S_{1Opt}$, $S_{2Opt}$ τα σύνολα που αποτελούν λύση του SSR (όταν ελαχιστοποιούμε τον λόγο) και $\varepsilon$ η ακρίβεια που έχει δοθεί σαν είσοδος.*

*Απόδειξη.* Από τα προηγούμενα έχουμε πως ισχύουν οι προϋποθέσεις του θεωρήματος 4.1 και άρα για το $A = \{a_1, a_2, ...a_n\}$ και τα ζεύγη συνόλων ($S_{1Opt}$, $S_{2Opt}$), ($S_1$, $S_2$) έχουμε ότι $\forall \varepsilon \in (0,1)$:

$$1 \leq \max\left\{\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} a_i}\right\} \leq \max\left\{\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{j\in S_{2Opt}} a_j}{\sum_{i\in S_{1Opt}} a_i}\right\} \cdot (1+\varepsilon)$$

Άρα ο αλγόριθμος έχει όντος την ακρίβεια που θέλουμε. □

## 5.3 Alternating SSR

Στο Alternating ESS μας δίνεται ένα σύνολο ζεύγη $(a_i, b_i)$ ακεραίων και για κάθε ζεύγος πρέπει να επιλέξουμε αν θα χρησιμοποιηθεί και αν ναι σε ποιο σύνολο του λόγου που ψάχνουμε θα βάλουμε το $a_i$ και σε ποιο το $b_i$ (ο ακριβής ορισμός υπάρχει στην εισαγωγή). Εμείς θα ασχοληθούμε εδώ με το αντίστοιχο πρόβλημα βελτιστοποίησης. Ακριβέστερα το πρόβλημα που θα προσεγγίσουμε είναι το ακόλουθο:

**Πρόβλημα 10** (Alternating SSR problem)**.** *Δοθέντος ενός συνόλου n ζευγών φυσικών $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$, ψάχνουμε δύο μη κενά και ξένα σύνολα $S_1$, $S_2 \subseteq \{1, 2, ..., n\}$ τέτοια ώστε $\sum_{i\in S_1} a_i + \sum_{i\in S_2} b_i \geq \sum_{j\in S_1} b_j + \sum_{j\in S_2} a_j$ και ο λόγος $\frac{\sum_{i\in S_1} a_i + \sum_{i\in S_2} b_i}{\sum_{j\in S_1} b_j + \sum_{j\in S_2} a_j}$ να ελαχιστοποιείτε.*

### 5.3.1 FPTAS

Εδώ δεν θα κατασκευάσουμε αλγόριθμο που λύνει ακριβώς το πρόβλημα βελτιστοποίησης αλλά θα πάμε κατευθείαν στην κατασκευή του FPTAS.

**Sub-Algorithm 5.2** Sub Algorithm for Alternating SSR

**Require:** a sorted set $A$ of n pairs of positive integers $\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ and a parameter $\varepsilon \in (0, 1)$

$\delta \leftarrow \frac{a_{n-1} \cdot \varepsilon}{6 \cdot n}$
$a'_i \leftarrow \lfloor \frac{a_i}{\delta} \rfloor$
$b'_i \leftarrow \lfloor \frac{b_i}{\delta} \rfloor$
**if** $\sum_{i=1}^{n-1} a'_i + b'_n < a'_n + \sum_{i=1}^{n-1} b'_i$ **then**
    **if** $\sum_{i=1}^{n-1} a_i + b_n < a_n + \sum_{i=1}^{n-1} b_i$ **then**
        $S_1 \leftarrow \{n\}$, $S_2 \leftarrow \{1, 2, ..., n-1\}$
        **return** ($S_1$, $S_2$, $a_n + \sum_{i=1}^{n-1} b_i$, $\sum_{i=1}^{n-1} a_i + b_n$)
    **else**
        $S_1 \leftarrow \{1, 2, ..., n-1\}$, $S_2 \leftarrow \{n\}$
        **return** ($S_1$, $S_2$, $\sum_{i=1}^{n-1} a_i + b_n$, $a_n + \sum_{i=1}^{n-1} b_i$)
    **end if**
**else**
    **for all** $0 \leq k \leq n$ **and** $x, y \in \{0, 1, ..., \sum_{i=1}^{n} a'_i\}$ **do**
        $F_k[x, y] \leftarrow \vec{\emptyset}$
    **end for**
    $F_0[b'_n, a'_n] \leftarrow (\emptyset, \{n\}, b_n, a_n)$
    **for** $k = 1$ to $n$ **do**
        **for all** $x, y \leq \sum_{i=1}^{n} a'_i$ **do**
            **if** $F_{k-1}[x, y] \neq \vec{\emptyset}$ **then**
                $(S_1, S_2, sum_1, sum_2) \leftarrow F_{k-1}[x, y]$
                $F_k[x, y] \leftarrow (S_1, S_2, sum_1, sum_2)$
                (if $F_k[x, y] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
                **if** $k < n$ **then**
                    $F_k[x + a'_k, y + b'_k] \leftarrow (S_1 \cup \{k\}, S_2, sum_1 + a_k, sum_2 + b_k)$
                    (if $F_k[x + a'_k, y + b'_k] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
                    $F_k[x + b'_k, y + a'_k] \leftarrow (S_1, S_2 \cup \{k\}, sum_1 + b_k, sum_2 + a_k)$
                    (if $F_k[x + b'_k, y + a'_k] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
                **end if**
            **end if**
        **end for**
    **end for**
    **for all** $F_n[x, y] = (S_1, S_2, sum_1, sum_2) \neq \vec{\emptyset}$ **and** $sum_1 \geq \frac{a_{n-1}}{2}$ **do**

**find** $\frac{x_1}{y_1}$ = **min** $\frac{x}{y} \geq 1$
**find** $\frac{y_2}{x_2}$ = **min** $\frac{y}{x} \geq 1$
**end for**
**if** $\frac{x_1}{y_1} \leq \frac{y_2}{x_2}$ **then**
$x_0 \leftarrow x_1$, $y_0 \leftarrow y_1$
**else**
$x_0 \leftarrow x_2$, $y_0 \leftarrow y_2$
**end if**
$(S_1, S_2, sum_1, sum_2) \leftarrow F_n[x_0, y_0]$
**if** $sum_1 \geq sum_2$ **then**
**return** $(S_1, S_2, sum_1, sum_2)$
**else**
**return** $(S_2, S_1, sum_2, sum_1)$
**end if**
**end if**

**Algorithm 5.4** FPTAS for Alternating SSR

**Require:** set $A$ of n pairs of positive integers $\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ and a parameter $\varepsilon \in (0, 1)$
**sort** $A$ (such that $a_i \geq b_i$, $\forall\ i = 1, 2, .., n$ and $a_i \leq a_j$, $\forall\ 1 \leq i \leq j \leq n$)
**for** $i = 2$ to $n$ **do**
$A' \leftarrow \{(a_1, b_1), ..., (a_i, b_i)\}$
**run** Sub-Algorithm 5.2 with **input** $A'$, $\varepsilon$ and **output** $(S_1^i, S_2^i, sum_1^i, sum_2^i)$
**end for**
**find** $k$ such that $\frac{sum_1^k}{sum_2^k}$ = **min**$_{2 \leq i \leq n} \left\{ \frac{sum_1^i}{sum_2^i} \right\}$
$ratio \leftarrow \frac{sum_1^i}{sum_2^i}\}$
**return** $S_1$, $S_2$, $ratio$

Το πρόβλημα το προσεγγίσαμε χρησιμοποιώντας το ίδιο σκεπτικό με την λύση του SSR που παρουσιάσαμε στην παράγραφο 5.2.2, δηλαδή λύνουμε το πρόβλημα με δεδομένη την χρήση του μεγίστου στοιχείου για όλα τα δυνατά μέγιστα.

### 5.3.2 Απόδειξη Ορθότητας

Η απόδειξη έχει την ίδια δομή με την τελευταία. Ξεκινάμε παρουσιάζοντας το λήμμα που μας δίνει ελάχιστες τιμές για τα αθροίσματα της βέλτιστες λύσης του Alternating SSR problem.

**Λήμμα 5.3.** *Αν για ένα ταξινομημένο σύνολο ζευγών* $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ *(*$a_i \geq b_i$ *και* $a_i \geq a_j$ $\forall 1 \leq i \leq j \leq n$*) τα σύνολα* $S_{1Opt}$*,* $S_{2Opt}$ *είναι αυτά που μας δίνουν τον ελάχιστο λόγο* $\frac{\sum_{i \in S_{1Opt}} a_i + \sum_{j \in S_{2Opt}} b_j}{\sum_{i \in S_{1Opt}} b_i + \sum_{j \in S_{2Opt}} a_j} \geq 1$ *με* $n \in S_{1Opt} \cup$

$S_{2Opt}$ *τότε:*

$$\frac{a_{n-1}}{2} \leq \sum_{i \in S_{1Opt}} b_i + \sum_{j \in S_{2Opt}} a_j \leq \sum_{i \in S_{1Opt}} a_i + \sum_{j \in S_{2Opt}} b_j$$

*Απόδειξη.* Η δεύτερη ανίσωση είναι προφανής από αφού θέλουμε ο λόγος να είναι μεγαλύτερος του ένα. Επιπλέον αν $n \in S_{2Opt}$ ισχύει και η πρώτη ανίσωση αφού $a_{n-1} \leq a_n$. Θα αποδείξουμε ότι η υπόθεση "$n \in S_{1Opt}$ και $a_{n-1} > \sum_{i \in S_{1Opt}} b_i + \sum_{j \in S_{2Opt}} a_j$" είναι άτοπη. Εδώ θα διακρίνουμε τις περιπτώσεις $n-1 \in S_{1Opt}$, $n-1 \in S_{1Opt}$ και $n-1 \notin S_{1Opt} \cup S_{2Opt}$.

**Περίπτωση 1.** *Αν* $n-1 \in S_{2Opt}$

*Εδώ προφανώς ισχύει η ανίσωση* $a_{n-1} \leq \sum_{i \in S_{1Opt}} b_i + \sum_{j \in S_{2Opt}} a_j$. *Οπότε η υπόθεση είναι* ***άτοπη***.

**Περίπτωση 2.** *Αν* $n-1 \in S_{1Opt}$

*Θέτω* $S_1 = S_{1Opt} \setminus \{n-1\}$.

**Υπο-περίπτωση 2.1.** *Αν* $\sum_{i \in S_1} b_i + \sum_{j \in S_{2Opt}} a_j + a_{n-1} < \sum_{i \in S_1} a_i + \sum_{j \in S_{2Opt}} b_j + b_{n-1}$

*Τότε έχουμε:*

$$\begin{aligned}
\sum_{i \in S_{1Opt}} b_i + \sum_{j \in S_{2Opt}} a_j &\leq \sum_{i \in S_1} b_i + \sum_{j \in S_{2Opt}} a_j + a_{n-1} \text{ (αφού } b_{n-1} \leq a_{n-1}\text{)} \\
&< \sum_{i \in S_1} a_i + \sum_{j \in S_{2Opt}} b_j + b_{n-1} \text{ (λόγο υποπερίπτωσης)} \\
&\leq \sum_{i \in S_{1Opt}} a_i + \sum_{j \in S_{2Opt}} b_j \text{ (αφού } b_{n-1} \leq a_{n-1}\text{)}
\end{aligned}$$

*όπου καταλήγει σε* ***άτοπο*** *αφού τότε τα* $S_1$, $S_2 = S_{2Opt} \cup \{n-1\}$ *θα έδιναν καλύτερο λόγο από τα* $S_{1Opt}$, $S_{2Opt}$.

**Υπο-περίπτωση 2.2.** *Αν* $\sum_{i \in S_1} a_i + \sum_{j \in S_{2Opt}} b_j + b_{n-1} \leq \sum_{i \in S_1} b_i + \sum_{j \in S_{2Opt}} a_j + a_{n-1}$

*Πριν σχολιάσουμε τις ανισώσεις να προσέξουμε ότι ελέγχουμε την υπόθεση* $n \in S_{1Opt}$ *και* $a_{n-1} > \sum_{i \in S_{1Opt}} b_i + \sum_{j \in S_{2Opt}} a_j$ *οπότε ισχύει:*

$$\sum_{i \in S_1} a_i + \sum_{j \in S_{2Opt}} b_j \geq a_n > \sum_{i \in S_1} b_i + \sum_{j \in S_{2Opt}} a_j \tag{5.2}$$

*το οποίο ισχύει λόγο της υπόθεσης που ελέγχουμε. Αναλύοντας τώρα τους όρους του*

*κλάσματος έχουμε:*

$$\begin{aligned}\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j &= \sum_{i\in S_1} a_i + \sum_{j\in S_{2Opt}} b_j + a_{n-1} \\ &> \sum_{i\in S_1} b_i + \sum_{j\in S_{2Opt}} a_j + a_{n-1} \textit{ (από εξ. 5.2)} \\ &\geq \sum_{i\in S_1} a_i + \sum_{j\in S_{2Opt}} b_j + b_{n-1} \textit{ (από υποπερίπτωση)} \\ &> \sum_{i\in S_1} b_i + \sum_{j\in S_{2Opt}} a_j + b_{n-1} \textit{ (από εξ. 5.2 )} \\ &= \sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j\end{aligned}$$

*το οποίο όμοια με πριν είναι* ***άτοπο*** *γιατί αλλιώς τα σύνολα* $S_1 = S_{2Opt} \cup \{n-1\}$, $S_2 = S_{1Opt} \setminus \{n-1\}$ *θα έδιναν καλύτερο λόγο από τα* $S_{1Opt}, S_{2Opt}$.

**Περίπτωση 3.** *Αν* $n-1 \notin S_{1Opt} \cup S_{2Opt}$

**Υπο-περίπτωση 3.1.** *Αν* $(\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j) - (\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j) > a_{n-1} - b_{n-1}$

*Εδώ έχω:*

$$(\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j) + b_{n-1} > (\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j) + a_{n-1}$$

*από το οποίο συμπεραίνουμε ότι:*

$$\begin{aligned}1 &< \frac{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j + b_{n-1}}{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j + a_{n-1}} \\ &< \frac{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j}{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j + (a_{n-1} - b_{n-1})} \\ &< \frac{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j}{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j}\end{aligned}$$

*οπότε τα* $S_{1Opt}, S_{2Opt}$ *δεν μας δίνουν το βέλτιστο* ***(άτοπο)***.

**Υπο-περίπτωση 3.2.** *Αν* $(\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j) - (\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j) \leq a_{n-1} - b_{n-1}$

*Εδώ έχω:*

$$(\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j) + b_{n-1} \leq (\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j) + a_{n-1}$$

*και αφού τα $S_{1Opt}$, $S_{2Opt}$ μας δίνουν το βέλτιστο λόγο έχουμε:*

$$\begin{aligned}
1 &\leq \frac{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j}{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j} \\
&\leq \frac{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j + a_{n-1}}{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j + b_{n-1}} \\
&\leq \frac{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j + a_{n-1}}{a_{n-1}}, \textit{(αφού } n \in S_{1Opt}\textit{)} \\
&= \frac{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j}{a_{n-1}} + 1 < 2
\end{aligned}$$

*όπου η τελευταία ανίσωση ισχύει λόγο της υπόθεσης $\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j < a_{n-1}$ και άρα συμπεραίνουμε ότι:*

$$\begin{aligned}
&\frac{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j}{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j} \leq 2 \Rightarrow \\
&a_{n-1} \leq a_n \leq \sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j \leq 2 \cdot (\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j) \Rightarrow \\
&\frac{a_{n-1}}{2} \leq \sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j
\end{aligned}$$

Άρα σε κάθε περίπτωση είτε καταλήγουμε σε **άτοπο** είτε ισχύει η ανίσωση του λήμματος. □

Εδώ δεν μπορούμε να χρησιμοποιήσουμε το θεώρημα 4.1 κατευθείαν γιατί αναφέρεται σε ένα σύνολο αριθμών ενώ εδώ έχουμε ένα σύνολο ζευγών. Λόγο αυτού θα κάνουμε κάποιες αλλαγές. Θεωρούμε το σύνολο $C = \{c_1, c_2, ...c_{2\cdot n}\}$ όπου $c_i = a_i, \forall i = 1, 2, ..., n$ και $c_i = b_{i-n} \forall i = n+1, n+2, ...2 \cdot n$. Επιπλέον πρέπει να δούμε ότι αν $S_1$, $S_2$ μια πιθανή λύση του Alternating SSR τότε ο λόγος που μας ενδιαφέρει έχει αριθμητή και παρονομαστή υποσύνολο του C. Ακριβέστερα αν θέσουμε $A_1^* = \{i | i \in A_1\} \cup \{j + n | j \in A_2\}$ και $A_2^* = \{i | i \in A_2\} \cup \{j + n | j \in A_1\}$ τότε ο λόγος $\max\{\sum_{i\in A_1^*} c_i / \sum_{j\in A_2^*} c_j, \sum_{i\in A_1^*} c_i \sum_{j\in A_2^*} c_j / \sum_{i\in A_1^*} c_i\}$ είναι ο ίδιος με αυτόν των $S_1$, $S_2$ για το Alternating SSR.

Θεωρούμε λιπόν $n_0 = \max\{S_{1Opt} \cup S_{2Opt}\}$ (όπου $S_{1Opt}$, $S_{2Opt}$ η βέλτιστη λύση του Alternating SSR) και αντί να ασχοληθούμε με τα $S_{1Opt}$, $S_{2Opt}$ και τα $S_1$, $S_2$ που επιστρέφει ο αλγόριθμος την $i = n_0$ επανάληψη, Θα ασχοληθούμε με τα $S_{1Opt}^*$, $S_{2Opt}^*$ και τα $S_1^*$, $S_2^*$ υποσύνολα του $C^* = \{c_1, c_2, ...c_{2\cdot n_0}\}$ που παράγονται όπως αναφέραμε προηγουμένως. Για τα καινούρια σύνολα μπορούμε να δείξουμε ότι ισχύουν οι προϋποθέσεις του θεωρήματος 4.1.

**Λήμμα 5.4.** *Στην επανάληψη όπου $i = n_0$ ισχύουν οι σχέσεις του Θεωρήματος 4.1 μεταξύ του $\delta$ και των συνόλων $S_1^*, S_2^*, S_{1Opt}^*, S_{2Opt}^*$, δηλαδή:*
*1.* $\delta = \frac{\varepsilon \cdot w}{3 \cdot m}$
*2.* $w \leq \sum_{i\in S} c_i, \forall S \in \{S_1^*, S_2^*, S_{1Opt}^*, S_{2Opt}^*\}$ *και*
*3.* $m \geq |S|, \forall S \in \{S_1^*, S_2^*, S_{1Opt}^*, S_{2Opt}^*\}$.

*Απόδειξη.* Όταν $i = n_0$ στον αλγόριθμο το $\delta$ έχει τιμή $\frac{\varepsilon \cdot a_{n_0-1}}{6 \cdot n_0}$. Αλλά τότε ισχύουν οι ακόλουθες δύο σχέσεις:

$$1.\ n_0 \geq |S|, \forall S \in \{S_1^*, S_2^*, S_{1Opt}^*, S_{2Opt}^*\}$$
$$2.\ \sum_{i \in S} c_i \geq \frac{a_{n_0-1}}{2}, \forall S \in \{S_1^*, S_2^*, S_{1Opt}^*, S_{2Opt}^*\}$$

όπου η πρώτη σχέση ισχύει γιατί $|S_1^*| = |S_2^*| = |S_1| + |S_2| \leq n_0$ (αφού $S_1$, $S_2$ ξένα υποσύνολα του $\{1, 2, ..., n_0\}$ από κατασκευή) και $|S_{1Opt}^*| = |S_{2Opt}^*| = |S_{1Opt}| + |S_{2Opt}| \leq n_0$ (αφού $S_{1Opt}$, $S_{2Opt}$ ξένα υποσύνολα του $\{1, 2, ..., n_0\}$ εξ ορισμού), ενώ η δεύτερη ισχύει από το λήμμα 5.3 και τον τρόπο επιλογής των $S_1$, $S_2$ από τον αλγόριθμο. Ακριβέστερα για την δεύτερη σχέση:

$$\sum_{i \in S_{1Opt}^*} c_i = \sum_{i \in S_{1Opt}} a_i + \sum_{i \in S_{2Opt}} b_i \geq \frac{a_{n_0-1}}{2}, \text{ λήμμα 5.3}$$
$$\sum_{i \in S_{2Opt}^*} c_i = \sum_{i \in S_{2Opt}} a_i + \sum_{i \in S_{1Opt}} b_i \geq \frac{a_{n_0-1}}{2}, \text{ λήμμα 5.3}$$

και από κατασκευή των $S_1$, $S_2$

$$\sum_{i \in S_1^*} c_i = \sum_{i \in S_1} a_i + \sum_{i \in S_2} b_i \sum_{i \in S_2^*} c_i = \sum_{i \in S_2} a_i + \sum_{i \in S_1} b_i$$

που έχουν επιλεχθεί ώστε το ένα άθροισμα να είναι αναγκαστικά μεγαλύτερο του $\frac{a_{n_0-1}}{2}$ και το άλλο να περιέχει το $a_{n_0} \geq \frac{a_{n_0-1}}{2}$.

Τότε μπορώ να θεωρήσω τα $w = \frac{a_{n_0-1}}{2}$ και $m = n_0$ και άρα το $\delta$ του αλγορίθμου πληρεί τις προϋποθέσεις του Θεωρήματος 4.1. □

Μας μένει να δείξουμε την σχέση μεταξύ των λόγων των αθροισμάτων αλλά για να το πετύχουμε αυτό θα δείξουμε πρώτα το ακόλουθο λήμμα

**Λήμμα 5.5.** *Αν για ένα ταξινομημένο σύνολο ζευγών* $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ *(με* $a_i \geq b_i \forall i = 1, 2, ..., n$ *και* $a_{i+1} \geq a_i \forall i = 1, 2, ..., n-1$*) ισχύει ότι* $\sum_{i=1}^{n-1} a_i + b_n < a_n + \sum_{i=1}^{n-1} b_i$ *τότε για κάθε ζεύγος* $S_1, S_2$ *ξένων υποσυνόλων του* $\{1, 2, ..., n\}$ *με* $\max\{S_1 \cap S_2\} = n$ *έχουμε:*

$$1 < \frac{a_n + \sum_{i=1}^{n-1} b_i}{\sum_{i=1}^{n-1} a_i + b_n} \leq \max\{\frac{\sum_{i \in S_1} a_i + \sum_{j \in S_2} b_j}{\sum_{j \in S_2} a_j + \sum_{i \in S_1} b_i}, \frac{\sum_{j \in S_2} a_j + \sum_{i \in S_1} b_i}{\sum_{i \in S_1} a_i + \sum_{j \in S_2} b_j}\}$$

*Απόδειξη.* Να ξεκινήσουμε παρατηρώντας ότι για κάθε σύνολο $S \subseteq \{1, 2, ..., n-1\}$ ισχύει η σχέση:

$$a_n + \sum_{i \in S} b_i > b_n + \sum_{i \in S} a_i \tag{5.3}$$

Αυτό προκύπτει θέτοντας $S' = \{1, 2, ..., n-1\} \setminus S$ και από την παρατήρηση ότι $-\sum_{i \in S'} b_i > -\sum_{i \in S'} a_i$ όπως φαίνεται παρακάτω:

$$\begin{aligned} a_n + \sum_{i=1}^{n-1} b_i &> \sum_{i=1}^{n-1} a_i + b_n \Rightarrow \\ a_n + \sum_{i=1}^{n-1} b_i - \sum_{i \in S'} b_i &> \sum_{i=1}^{n-1} a_i - \sum_{i \in S'} a_i + b_n \Rightarrow \\ a_n + \sum_{i \in S} b_i &> \sum_{i \in S} a_i + b_n \end{aligned}$$

Αφού αποδείξαμε την προηγούμενη σχέση θα δείξουμε δύο σχέσεις για δύο τυχαία σύνολα. Έστω $S_1$ και $S_2$ δύο ξένα υποσύνολα του $\{1, 2, ..., n\}$ και $\max\{S_1 \cup S_2\} = n$. Αν $n \in S_1$ θέτω $S_A = S_1 \setminus n$ και $S_B = S_2$ αλλιώς θέτω $S_A = S_2 \setminus n$ και $S_B = S_1$. Τότε έχουμε:

$$1.\ a_n + \sum_{i \in S_A} a_i + \sum_{j \in S_B} b_j > b_n + \sum_{j \in S_B} a_j + \sum_{i \in S_A} b_i \tag{5.4}$$

$$2.\ 1 < \frac{a_n + \sum_{i \in S_A \cup S_B} b_i}{b_n + \sum_{i \in S_A \cup S_B} a_i} < \frac{a_n + \sum_{i \in S_A} a_i + \sum_{j \in S_B} b_j}{b_n + \sum_{j \in S_B} a_j + \sum_{i \in S_A} b_i} \tag{5.5}$$

Το πρώτο ισχύει αφού:

$$\begin{aligned} a_n + \sum_{i \in S_A} a_i + \sum_{j \in S_B} b_j &> a_n + \sum_{i \in S_A} b_i + \sum_{j \in S_B} b_j, \text{ αφού } a_i \geq b_i \forall i = 1, 2, ..., n \\ &> b_n + \sum_{i \in S_A} a_i + \sum_{j \in S_B} a_j, \text{ από σχέση 5.3} \\ &> b_n + \sum_{i \in S_A} b_i + \sum_{j \in S_B} a_j, \text{ αφού } a_i \geq b_i \forall i = 1, 2, ..., n \end{aligned}$$

Ενώ για το δεύτερο έχω:

$$1 < \frac{a_n + \sum_{i \in S_A} a_i + \sum_{j \in S_B} b_j}{b_n + \sum_{j \in S_B} a_j + \sum_{i \in S_A} b_i}, \text{ από σχέση 5.3}$$

και

$$\begin{aligned} \frac{a_n + \sum_{i \in S_A} a_i + \sum_{j \in S_B} b_j}{b_n + \sum_{j \in S_B} a_j + \sum_{i \in S_A} b_i} &> \frac{a_n + \sum_{i \in S_A} b_i + \sum_{j \in S_B} b_j}{b_n + \sum_{j \in S_B} a_j + \sum_{i \in S_A} a_i} \\ &= \frac{a_n + \sum_{i \in S_A \cup S_B} b_i}{b_n + \sum_{j \in S_A \cup S_B} a_j} \\ &> 1, \text{ από σχέση } 5.3 \end{aligned}$$

Έχοντας αποδείξει τη σχέση 5.5 για να ισχύει το λήμμα μας μένει να δείξουμε ότι αν έχω $S_1 \subset S_2 \subseteq \{1, 2, ..., n-1\}$ τότε ισχύει ότι:

$$1 < \frac{a_n + \sum_{i \in S_2} b_i}{b_n + \sum_{j \in S_2} a_j} < \frac{a_n + \sum_{i \in S_1} b_i}{b_n + \sum_{j \in S_1} a_j} \tag{5.6}$$

Από τη σχέση 5.3 προκύπτει πως οι δύο λόγου είναι μεγαλύτεροι του 1 και άρα μένει να δείξουμε την μεταξύ τους σχέση. Αυτό είναι εύκολο καθώς:

$$\begin{aligned}
\frac{a_n + \sum_{i \in S_1} b_i}{b_n + \sum_{j \in S_1} a_j} &> \frac{a_n + \sum_{i \in S_1} b_i + \sum_{j \in S_2 \setminus S_1} b_j}{b_n + \sum_{j \in S_1} a_j + \sum_{j \in S_2 \setminus S_1} b_j} \\
&> \frac{a_n + \sum_{i \in S_2} b_i}{b_n + \sum_{j \in S_1} a_j + \sum_{j \in S_2 \setminus S_1} a_j}, \text{ αφού } a_i \geq b_i \forall i = 1, 2, ..., n \\
&> \frac{a_n + \sum_{i \in S_2} b_i}{b_n + \sum_{j \in S_2} a_j}
\end{aligned}$$

Οπότε από τις σχέσεις 5.5 και 5.6 προκύπτει το λήμμα. □

**Λήμμα 5.6.** *Τα $S_1^*$, $S_2^*$ και $S_{1Opt}^*$, $S_{2Opt}^*$ (όπως αυτά τα ορίσαμε παραπάνω) πληρούν την συνθήκη των λόγων του Θεωρήματος 4.1, δηλαδή:*

$$1 \leq \max\left\{\frac{\sum_{i \in S_1} c'_i}{\sum_{j \in S_2} c'_j}, \frac{\sum_{j \in S_2} c'_j}{\sum_{i \in S_1} c'_i}\right\} \leq \max\left\{\frac{\sum_{i \in S_{1Opt}} c'_i}{\sum_{j \in S_{2Opt}} c'_j}, \frac{\sum_{j \in S_{2Opt}} c'_j}{\sum_{i \in S_{1Opt}} c'_i}\right\}$$

*Απόδειξη.* Να υπενθυμίσουμε ότι οι τιμές του δίνουν τα αθροίσματα των $S_1^*$, $S_2^*$ και $S_{1Opt}^*$, $S_{2Opt}^*$ είναι οι ακόλουθες:

$$\begin{aligned}
\sum_{i \in S_1^*} c'_i &= \sum_{i \in S_1} a'_i + \sum_{i \in S_2} b'_i \\
\sum_{i \in S_2^*} c'_i &= \sum_{i \in S_2} a'_i + \sum_{i \in S_1} b'_i \\
\sum_{i \in S_{1Opt}^*} c'_i &= \sum_{i \in S_{1Opt}} a'_i + \sum_{i \in S_{2Opt}} b'_i \\
\sum_{i \in S_{2Opt}^*} c'_i &= \sum_{i \in S_{2Opt}} a'_i + \sum_{i \in S_{1Opt}} b'_i
\end{aligned}$$

και ότι $\max\{S_1 \cup S_2\} = \max\{S_{1Opt} \cup S_{2Opt}\} = n_0$.

Θα χωρίσουμε την απόδειξη σε περιπτώσεις.

**Περίπτωση 1.** *Αν $\sum_{i=1}^{n-1} a'_i + b'_n < a'_n + \sum_{i=1}^{n-1} b'_i$*

*Εδώ από το λήμμα 5.5 ισχύει η σχέση.*

**Περίπτωση 2.** *Αν $\sum_{i=1}^{n-1} a'_i + b'_n \geq a'_n + \sum_{i=1}^{n-1} b'_i$*

*Αρκεί να δείξουμε ότι ένα εκ των κελιών:*

$$F_n[\sum_{i\in S_{1Opt}} a'_i + \sum_{i\in S_{2Opt}} b'_i, \sum_{i\in S_{2Opt}} a'_i + \sum_{i\in S_{1Opt}} b'_i] \text{ και}$$
$$F_n[\sum_{i\in S_{2Opt}} a'_i + \sum_{i\in S_{1Opt}} b'_i, \sum_{i\in S_{1Opt}} a'_i + \sum_{i\in S_{2Opt}} b'_i]$$

*είναι μή κενό και η μεταβλητή $\sum_1$ είναι μεγαλύτερη του $\frac{a_{n-1}}{2}$.*

*Χωρίς βλάβη της γενικότητας θα θεωρήσουμε ότι $n \in S_2$. Τότε ο συνδυασμός των $S_{1Opt}, S_{1Opt}$ μας εγγυάται ότι το κελί $F_n[\sum_{i\in S_{1Opt}} a'_i + \sum_{i\in S_{2Opt}} b'_i, \sum_{i\in S_{2Opt}} a'_i + \sum_{i\in S_{1Opt}} b'_i]$ δεν είναι κενό, και αφού ο αλγόριθμος κρατά το συνδυασμό με μέγιστο $\sum_1$ σημαίνει ότι, από το λήμμα 5.3, έχει τιμή μεγαλύτερη από $\frac{a_{n-1}}{2}$. Άρα ο λόγος που επιστρέφει ο αλγόριθμος πληρεί την συνθήκη του θεωρήματος.*

□

Άρα μπορούμε να κάνουμε χρήση του θεωρήματος που σημαίνει ότι τα σύνολα που επιστρέψαμε μας δίνουν λόγο εντός του διαστήματος:

$$1 \leq \max\{\frac{\sum_{i\in S_1} a_i + \sum_{j\in S_2} b_j}{\sum_{i\in S_1} b_i + \sum_{j\in S_2} a_j}, \frac{\sum_{i\in S_1} b_i + \sum_{j\in S_2} a_j}{\sum_{i\in S_1} a_i + \sum_{j\in S_2} b_j}\}$$
$$\leq (1+\varepsilon)\cdot \max\{\frac{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j}{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{i\in S_{1Opt}} b_i + \sum_{j\in S_{2Opt}} a_j}{\sum_{i\in S_{1Opt}} a_i + \sum_{j\in S_{2Opt}} b_j}\}$$

## 5.4 Υποσύνολα λόγου r

Με το Factor-r Sum Subsets problem ασχοληθήκαμε αρκετά στο κεφάλαιο 3. Το FPTAS που αναπτύξαμε σε εκείνο το κεφάλαιο μας επέστρεφε δύο σύνολα $S_1$, $S_2$ τέτοια ώστε $\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j} \in \left[(1-\varepsilon)\cdot \frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{1}{r}\cdot(1+\varepsilon)\right]$ όπου τα $S_{1Opt}$ και $S_{2Opt}$ είναι τα σύνολα που μας δίνουν το μέγιστο λόγο που είναι μικρότερος του $\frac{1}{r}$. Η προσέγγιση που θα κάνουμε σε αυτή την παράγραφο είναι αρκετά διαφορετική. Αρχικά ας ορίσουμε το πρόβλημα βελτιστοποίησης που πάμε να προσεγγίσουμε.

**Πρόβλημα 11** (Factor-r ratio problem)**.** *Δοθέντος ενός συνόλου n θετικών ακεραίων $A = \{a_1, a_2, , ..., a_n\}$ και ενός θετικού $r$, ψάχνουμε δύο μη κενά και ξένα σύνολα $S_1$, $S_2 \subseteq \{1, 2, ...n\}$ τέτοια ώστε για οποιαδήποτε $S'_1$ και $S'_2$ μη κενά και ξένα υποσύνολα του $\{1, 2, ...n\}$ να ισχύει:*

$$\max\{\frac{\sum_{i\in S_1} r\cdot a_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} r\cdot a_i}\} = \min_{S'_1,S'_2}\Big\{\max\{\frac{\sum_{i\in S'_1} r\cdot a_i}{\sum_{j\in S'_2} a_j}, \frac{\sum_{j\in S'_2} a_j}{\sum_{i\in S'_1} r\cdot a_i}\}\Big\}$$

Εδώ καλό είναι να αναλύσουμε λίγο παραπάνω τον ορισμό, καθώς μπορεί να είναι δυσνόητος, και να εξηγήσουμε τι θα επιστρέφει ο αλγόριθμος που θα αναπτύξουμε. Καταρχάς να παρατηρήσουμε ότι η επιλογή του μεγίστου στον ορισμό μας

διασφαλίζει ότι το Opt που ψάχνουμε είναι μεγαλύτερο του 1. Αφού δεν υπάρχει περιορισμός για τα $S_1'$, $S_2'$ σημαίνει ότι ελέγχουμε όλους τους λόγους που το r βρίσκετε είτε στον παρονομαστή είτε στον αριθμητή και είναι μεγαλύτεροι του 1 ενώ στο τέλος διαλέγουμε τον μικρότερο από όλους αυτούς.

### 5.4.1 FPTAS

Ο αλγόριθμος που θα αναπτύξουμε θέλουμε να επιστρέφει δύο σύνολα $S_1^*$, $S_2^*$ τέτοια ώστε ένας εκ' των λόγων $\frac{r\cdot\sum_{i\in S_1^*} a_i}{\sum_{j\in S_2^*} a_j}$, $\frac{\sum_{i\in S_1^*} a_i}{r\cdot\sum_{j\in S_2^*} a_j}$ να βρίσκεται εντός του διαστήματος $\Big[Opt, Opt\cdot(1+\varepsilon)\Big]$. Η βασική διαφορά στον αλγόριθμο, που θα πρέπει να τονιστεί, είναι τελευταία μεταβλητή (r-set) που θα κρατάμε σε κάθε κελί. Η μεταβλητή αυτή θα παίρνει τιμές 1 ή 2 ανάλογα με το σε ποίο από τα δύο σύνολα που κρατάμε έχουμε πολλαπλασιάσει με το r. Ο αλγόριθμος που προσεγγίζει το πρόβλημα φαίνεται παρακάτω.

**Algorithm 5.5** FPTAS for Factor-r ratio

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ a parameter $r$ and a parameter $\varepsilon \in (0,1)$

1: **sort** $A$
2: **if** $r < 1$ **then**
3: $\quad r \leftarrow \frac{1}{r}$
4: **end if**
5: **for** $m = 2$ to $n$ **do**
6: $\quad A' \leftarrow \{a_1, a_2, ..., a_m\}$
7: $\quad$ **run** Sub-Algorithm 5.3 with **input** $A'$, $r$, $\varepsilon$ and **output** ($S_1^i, S_2^i, sum_1^i, sum_2^i, r-set_i$)
8: **end for**
9: **find** $k$ such that $\frac{sum_1^k}{sum_2^k}$ = **min**$_{2\le i\le n}\left\{\frac{sum_1^i}{sum_2^i}\right\}$
10: $ratio \leftarrow \frac{sum_1^k}{sum_2^k}$
11: **return** "the r-set is="$r-set_k$, $S_1^k$, $S_2^k$, $ratio$

**Sub-Algorithm 5.3** Sub Algorithm for Factor-r

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ a parameter $r \ge 1$ and a parameter $\varepsilon \in (0,1)$

1: $\delta \leftarrow \frac{a_{m-1}\cdot\varepsilon}{3\cdot m}$
2: $a_i' \leftarrow \lfloor\frac{a_i}{\delta}\rfloor$ for all $1 \le i \le n$
3: $b_i' \leftarrow \lfloor\frac{r\cdot a_i}{\delta}\rfloor$ for all $1 \le i \le n$
4: **if** $\sum_{i=1}^{n-1} b_i' < a_n'$ **then**

5: **if** $r \cdot \sum_{i=1}^{n-1} a_i < a_n$ **then**
6: $S_1 \leftarrow \{n\}$, $S_2 \leftarrow \{1, 2, ..., n-1\}$
7: **return** ($S_1$, $S_2$, $a_n$, $r \cdot \sum_{i=1}^{n-1} a_i$, 2)
8: **else**
9: $S_1 \leftarrow \{1, 2, ..., n-1\}$, $S_2 \leftarrow \{n\}$
10: **return** ($S_1$, $S_2$, $r \cdot \sum_{i=1}^{n-1} a_i$, $a_n$, 1)
11: **end if**
12: **else**
13: **for all** $0 \leq k \leq n$ **and** $0 \leq x \leq y \leq \sum_{i=1}^{n} b'_i$ **do**
14: $F_k[x, y] \leftarrow \vec{\emptyset}$
15: **end for**
16: $F_0[0, a'_n] \leftarrow (\{n\}, \emptyset, 0, a_n, 1)$
17: $F_0[0, b'_n] \leftarrow (\emptyset, \{n\}, 0, r \cdot a_n, 2)$
18: **for** $k = 1$ to $n$ **do**
19: **for all** $x, y \leq \sum_{i=1}^{n} b'_i$ **do**
20: **if** $F_{k-1}[x, y] \neq \vec{\emptyset}$ **then**
21: $(S_1, S_2, sum_1, sum_2, r - set) \leftarrow F_{k-1}[x, y]$
22: $F_k[x, y] \leftarrow (S_1, S_2, sum_1, sum_2, r - set)$
23: (if $F_k[x, y] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
24: **if** $k < n$ **and** $r - set = 1$ **then**
25: $F_k[x + b'_k, y] \leftarrow (S_1 \cup \{k\}, S_2, sum_1 + r \cdot a_k, sum_2, r - set)$
26: (if $F_k[x + b'_k, y] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
27: $F_k[x, y + a'_k] \leftarrow (S_1, S_2 \cup \{k\}, sum_1, sum_2 + a_k, r - set)$
28: (if $F_k[x, y + a'_k] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
29: **end if**
30: **if** $k < n$ **and** $r - set = 2$ **then**
31: $F_k[x + a'_k, y] \leftarrow (S_1 \cup \{k\}, S_2, sum_1 + a_k, sum_2, r - set)$
32: (if $F_k[x + a'_k, y] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
33: $F_k[x, y + b'_k] \leftarrow (S_1, S_2 \cup \{k\}, sum_1, sum_2 + r \cdot a_k, r - set)$
34: (if $F_k[x, y + b'_k] \neq \vec{\emptyset}$ we use the one with the bigest "third value")
35: **end if**
36: **end if**
37: **end for**
38: **end for**
39: **for all** $F_n[x, y] = (S_1, S_2, sum_1, sum_2, r - set) \neq \vec{\emptyset}$ **and** $sum_1 \geq a_{n-1}$ **do**

40: **find** $\frac{x_1}{y_1} = \min \frac{x}{y} \geq 1$
41: **find** $\frac{y_2}{x_2} = \min \frac{y}{x} \geq 1$
42: **end for**
43: **if** $\frac{x_1}{y_1} \leq \frac{y_2}{x_2}$ **then**
44: $x_0 \leftarrow x_1, y_0 \leftarrow y_1$
45: **else**
46: $x_0 \leftarrow x_2, y_0 \leftarrow y_2$
47: **end if**
48: $(S_1, S_2, sum_1, sum_2, r - set) \leftarrow F_n[x_0, y_0]$
49: **if** $sum_1 \geq sum_2$ **then**
50: **return** $(S_1, S_2, sum_1, sum_2, r - set)$
51: **else if** $r - set = 1$ **then**
52: **return** $(S_2, S_1, sum_2, sum_1, 2)$
53: **else**
54: **return** $(S_2, S_1, sum_2, sum_1, 1)$
55: **end if**
56: **end if**

### 5.4.2 Απόδειξη Ορθότητας

Όπως και στα προηγούμενα θα ξεκινήσουμε παρουσιάζοντας ένα λήμμα που μας δίνει ελάχιστες τιμές για τα αθροίσματα της βέλτιστης λύσης του Factor-r problem.

**Λήμμα 5.7.** *Αν για ένα ταξινομημένο σύνολο* $A = \{a_1, a_2, ..., a_n\}$ *τα σύνολα* $S_{1Opt}$, $S_{2Opt}$ *μας δίνουν την βέλτιστη λύση* $\max\{\frac{r \cdot \sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}, \frac{\sum_{j \in S_{2Opt}} a_j}{r \cdot \sum_{i \in S_{1Opt}} a_i}\}$ *του Factor-r Ratio problem (όπως αυτό ορίζεται στο 11) και επιπλέον* $n \in S_{1Opt} \cup S_{2Opt}$ *τότε:*

$$a_{n-1} \leq \sum_{j \in S_{2Opt}} a_j \leq r \cdot \sum_{i \in S_{1Opt}} a_i \qquad \text{ή}$$

$$a_{n-1} \leq r \cdot \sum_{i \in S_{1Opt}} a_i \leq \sum_{j \in S_{2Opt}} a_j$$

*(ανάλογα αν, στον βέλτιστο λόγο, το r βρίσκεται στον αριθμητή ή το παρονομαστή αντίστοιχα).*

*Απόδειξη.* Εξ' υποθέσεως έχουμε ότι $n \in S_{1Opt} \cup S_{2Opt}$ και αφού το $A$ είναι ταξινομημένο έχουμε $a_n = max\{a_i | i \in S_{1Opt} \cup S_{2Opt}\}$. Το δεύτερο μισό των ανισοτήτων είναι προφανές από τον ορισμό του προβλήματος.

**Περίπτωση 1.** *Αν το r είναι στον αριθμητή(πρώτη ανίσωση):*

*Έστω ότι* $a_{n-1} > \sum_{j \in S_{2Opt}} a_j$ *τότε θα είχαμε ότι* $n - 1 \notin S_{2Opt}$ *και* $n \in S_{1Opt}$ *άρα:*

$$\frac{r \cdot \sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j} > \frac{r \cdot a_{n-1}}{\sum_{j \in S_{2Opt}} a_j} > 1$$

*το οποίο είναι **άτοπο** καθώς τα σύνολα $S_{1Opt}$, $S_{2Opt}$ μας δίνουν την βέλτιστη λύση.*

**Περίπτωση 2.** *Αν το r είναι στο παρονομαστή(δεύτερη ανίσωση):*

*Έστω ότι $a_{n-1} > r \cdot \sum_{i \in S_{1Opt}} a_i$ τότε θα είχαμε ότι $n-1 \notin S_{1Opt}$ και $n \in S_{2Opt}$ άρα:*

$$\frac{\sum_{j \in S_{2Opt}} a_j}{r \cdot \sum_{i \in S_{1Opt}} a_i} > \frac{a_{n-1}}{r \cdot \sum_{i \in S_{1Opt}} a_i} > 1$$

*το οποίο, όμοια με πριν, είναι **άτοπο**.*

Άρα ισχύουν οι ανισώσεις του λήμματος. □

Εδώ όπως και στο προηγούμενο κεφάλαιο δεν μπορούμε να κάνουμε χρήση του θεωρήματος 4.1 κατευθείαν. Για να μπορούμε να κάνουμε χρήση του θεωρήματος πρέπει να θεωρήσουμε τα σύνολα του μας ενδιαφέρουν ως υποσύνολα του $C = \{c_1, c_2, ..., c_{2 \cdot n}\}$ όπου για τα $c_i$ ισχύει ότι $c_i = a_i \ \forall i = 1, 2, ..., n$ και $c_i = r \cdot a_{i-n}$ $\forall i = n+1, n+2, ..., 2 \cdot n$. Αν θεωρήσουμε τα σύνολα $S_{1Opt}$, $S_{2Opt}$ που μας δίνουν την βέλτιστη λύση $\max\{\frac{r \cdot \sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}, \frac{\sum_{j \in S_{2Opt}} a_j}{r \cdot \sum_{i \in S_{1Opt}} a_i}\}$ τότε αυτά μπορούν να αντικατασταθούν από τα $S^*_{1Opt} = \{i + n | i \in S_{1Opt}$, $S^*_{2Opt} = S_{2Opt}$ τα οποία μας δίνουν ίδιο λόγο μέσο μου C. Με βάση αυτή την μετατροπή όλοι οι όροι των αθροισμάτων όλων των συνδυασμών $S_1, S_2$ που είναι πιθανές λύσεις του αρχικού προβλήματος υπάρχουν στο C.

Θεωρούμε λιπών $n_0 = \max\{S_{1Opt} \cup S_{2Opt}\}$ (όπου $S_{1Opt}$, $S_{2Opt}$ η βέλτιστη λύση του προβλήματος). Θα ασχοληθούμε με τα $S_{1Opt}$, $S_{2Opt}$ και τα $S_1$, $S_2$ που επιστρέφει ο αλγόριθμος την $i = n_0$ επανάληψη. Αυτά τα σύνολα τα μετατρέπουμε όπως περιγράψαμε παραπάνω και παίρνουμε τα $S^*_{1Opt}$, $S^*_{2Opt}$ και $S^*_1$, $S^*_2$(όπου $S^*_1$ αυτό που χρησιμοποιεί το r στο λόγο που επιστρέφει ο αλγόριθμος). Για αυτά θα αποδείξουμε τις προϋποθέσεις του θεωρήματος 4.1.

**Λήμμα 5.8.** *Στην επανάληψη όπου $i = n_0$ ισχύουν οι σχέσεις του Θεωρήματος 4.1 μεταξύ του $\delta$ και των συνόλων $S^*_1, S^*_2, S^*_{1Opt}, S^*_{2Opt}$, δηλαδή:*
*1.* $\delta = \frac{\varepsilon \cdot w}{3 \cdot m}$
*2.* $w \leq \sum_{i \in S} c_i, \forall S \in \{S^*_1, S^*_2, S^*_{1Opt}, S^*_{2Opt}\}$ *και*
*3.* $m \geq |S|, \forall S \in \{S^*_1, S^*_2, S^*_{1Opt}, S^*_{2Opt}\}$.

*Απόδειξη.* Όταν $i = n_0$ στον αλγόριθμο το $\delta$ έχει τιμή $\frac{\varepsilon \cdot a_{n_0-1}}{3 \cdot n_0}$. Αλλά τότε ισχύουν οι ακόλουθες δύο σχέσεις:

1. $n_0 \geq |S|, \forall S \in \{S^*_1, S^*_2, S^*_{1Opt}, S^*_{2Opt}\}$
2. $\sum_{i \in S} c_i \geq a_{n_0-1}, \forall S \in \{S^*_1, S^*_2, S^*_{1Opt}, S^*_{2Opt}\}$

Όπου η πρώτη σχέση είναι προφανής από το ότι $n_0 = \max\{S_{1Opt} \cup S_{2Opt}\}$ και $n_0 = \max\{S_1 \cup S_2\}$, ενώ η δεύτερη ισχύει από το λήμμα 5.7 και τον τρόπο επιλογής των $S_1$, $S_2$(που παράγουν $S^*_1$, $S^*_2$) από τον αλγόριθμο. Τότε μπορώ να θεωρήσω τα

$w = a_{n_0-1}$ και $m = n_0$ και άρα το $\delta$ του αλγορίθμου πληρεί τις προϋποθέσεις του Θεωρήματος 4.1. □

Μας μένει να δείξουμε ότι ο λόγος που επιστρέφει ο αλγόριθμος ικανοποιεί πάντα τη σχέση των λόγων του Θεωρήματος 4.1. Αυτό θα δειχθεί σε δύο λήμματα.

**Λήμμα 5.9.** *Αν για τα ταξινομημένα σύνολα* $A = \{a_1, a_2, ..., a_n\}$, $B = \{b_1, b_2, ..., b_n\}$ *(με* $a_{i+1} \geq a_i$ *και* $b_{i+1} \geq b_i \, \forall i = 1, 2, ..., n-1$ *και* $a_i \leq b_i \forall i = 1, 2, ..., n$*) και τον πραγματικό αριθμό* $r \geq 1$ *ισχύει ότι* $a_n \geq \sum_{i=1}^{n-1} b_i$ *τότε για κάθε ζεύγος* $S_1, S_2$ *ξένων υποσυνόλων του* $\{1, 2, ..., n\}$ *με* $\max\{S_1 \cup S_2\} = n$ *έχουμε:*

$$1 \leq \frac{a_n}{\sum_{i=1}^{n-1} b_i} \leq max\{\frac{\sum_{i \in S_1} b_i}{\sum_{j \in S_2} a_j}, \frac{\sum_{j \in S_2} a_j}{\sum_{i \in S_1} b_i}\}$$

*Απόδειξη.* Να παρατηρήσουμε ότι η πρώτη ανίσωση της σχέσης είναι προφανής από την υπόθεση του λήμματος ότι $a_n \geq \sum_{i=1}^{n-1} b_i$. Επιπλέον είναι εύκολο να παρατηρήσουμε ότι ισχύουν οι παρακάτω ανισώσεις:

$$b'_n \geq a'_n \geq \sum_{i=1}^{n-1} b'_i \geq \sum_{i \in S} b'_i \geq \sum_{i \in S} a'_i \, , \forall S \subseteq \{1, 2, ..., n-1\} \tag{5.7}$$

και αφού $\max\{S_1 \cap S_2\} = n$ χωρίσουμε περιπτώσεις:

**Περίπτωση 1.** $n \in S_1$

*Τότε από την σχέση 5.7 συμπεραίνουμε με ευκολία ότι* $\sum_{i \in S_1} a'_i \geq a'_n \geq \sum_{j \in S_2} a'_j$ *και άρα ισχύει ότι:*

$$\begin{aligned} max\{\frac{\sum_{i \in S_1} b_i}{\sum_{j \in S_2} a_j}, \frac{\sum_{j \in S_2} a_j}{\sum_{i \in S_1} b_i}\} = \frac{\sum_{i \in S_1} b_i}{\sum_{j \in S_2} a_j} &\geq \frac{b_n}{\sum_{j \in S_2} a_j} \\ &\geq \frac{a_n}{\sum_{j \in S_2} a_j} \geq \frac{a_n}{\sum_{j \in S_2} b_j} \\ &\geq \frac{a_n}{\sum_{i=1}^{n-1} b_i} \end{aligned}$$

**Περίπτωση 2.** $n \in S_2$

*Τότε από την σχέση 5.7 συμπεραίνουμε με ευκολία ότι* $\sum_{i \in S_1} b_i \leq a_n \leq \sum_{j \in S_2} a_j$ *και άρα ισχύει ότι:*

$$\begin{aligned} max\{\frac{\sum_{i \in S_1} b_i}{\sum_{j \in S_2} a_j}, \frac{\sum_{j \in S_2} a_j}{\sum_{i \in S_1} b_i}\} = \frac{\sum_{j \in S_2} a_j}{\sum_{i \in S_1} b_i} &\geq \frac{a_n}{\sum_{i \in S_1} b_i} \\ &\geq \frac{a_n}{\sum_{i=1}^{n-1} b_i} \end{aligned}$$

□

Από το παραπάνω λήμμα συμπεραίνουμε ότι αν στην επανάληψη όπου $i = n_0$ ισχύει η σχέση $a'_{n_0} \geq \sum_{i=1}^{n_0-1} b'_i$ στον υποαλγόριθμο 5.3 τότε ο λόγος που επιστρέφεται πληρεί τις προϋποθέσεις του θεωρήματος 4.1. Μένει να ασχοληθούμε με την περίπτωση $a'_{n_0} < \sum_{i=1}^{n_0-1} b'_i$.

**Λήμμα 5.10.** *Αν για τα ταξινομημένα σύνολα* $A = \{a'_1, a'_2, ..., a'_n\}$, $B = \{b'_1, b'_2, ..., b'_n\}$ *του υποαλγορίθμου 5.3 ισχύει* $a'_n < \sum_{i=1}^{n-1} b'_i$ *τότε ο αλγόριθμος επιστρέφει δύο ξένα υποσύνολα του* $\{1, 2, ..., n\}$, $S_1$ *και* $S_2$, *τέτοια ώστε* $\max\{S_1 \cup S_2\} = n$ *και έχουν τον ελάχιστο λόγο. Δηλαδή για κάθε* $S*_1, S*_2$ *ξένα υποσύνολα του* $\{1, 2, ..., n\}$ *με* $\max\{S *_1 \cup S*_2\} = n$ *ισχύει η σχέση:*

$$max\{\frac{\sum_{i\in S_1} b_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} b_i}\} \leq max\{\frac{\sum_{i\in S*_1} b_i}{\sum_{j\in S*_2} a_j}, \frac{\sum_{j\in S*_2} a_j}{\sum_{i\in S*_1} b_i}\}$$

*Απόδειξη.* Ο υποαλγόριθμος 5.3 κατασκευάζει ένα πίνακα που καλύπτει όλα τα αθροίσματα μεγέθους μέχρι $\sum_{i=1}^{n_0} b_i$ και αφού αυτή είναι η μέγιστη τιμή που μπορεί να πάρει ένα άθροισμα σημαίνει ότι έχει όλα τα δυνατά ζεύγη αθροισμάτων. Τέλος αν παρατηρήσουμε ότι ο αλγόριθμος επιστρέφει τα σύνολα που δίνουν τον ελάχιστο λόγο(μεγαλύτερο του 1) σημαίνει ότι ισχύει το λήμμα. □

Με την ολοκλήρωση αυτού του λήμματος έχουμε ότι στην $i = n_0$ επανάληψη ισχύουν οι προϋποθέσεις του θεωρήματος 4.1 και άρα επιστρέφουμε λόγο $1 + \varepsilon$ κοντά στον βέλτιστο. Δηλαδή αν τα σύνολα $S_{1Opt}$, $S_{2Opt}$ μας δίνουν την βέλτιστη λύση και $S_1$, $S_2$ τα σύνολα που επιστέφει ο υποαλγόριθμος 5.3 την $i = n_0$ επανάληψη συνδέονται με την σχέση:

$$max\{\frac{\sum_{i\in S_1} b_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} b_i}\} \leq (1+\varepsilon) \cdot max\{\frac{\sum_{i\in S_{1Opt}} b_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{j\in S_{2Opt}} a_j}{\sum_{i\in S_{1Opt}} b_i}\}$$

και αφού επιστρέφουμε τον μικρότερο λόγο από όλες τις επαναλήψεις συνεχίζουμε να έχουμε αυτή την σχέση για τα τελικά σύνολα που επιστρέφουμε.

## 5.5 Πολυπλοκότητα

Έχοντας αποδείξει ότι ο αλγόριθμος επιστρέφει ένα λόγο στο διάστημα που μας ενδιαφέρει μας μένει να δούμε την πολυπλοκότητα του. Το σημαντικό κομμάτι είναι η κατασκευή του πίνακα και το πόσες φορές αυτή πραγματοποιείτε. Γεμίζουμε τον πίνακα το πολύ n-1 φορές (όσα είναι τα δυνατά μέγιστα) ενώ για το γέμισμα χρειαζόμαστε μια διαπέραση όλου του πίνακα. Από αυτά συμπεραίνουμε ότι η πολυπλοκότητα είναι ίση με τη διάσταση του πίνακα πολλαπλασιασμένη με το n. Η διάσταση του πίνακα διαφέρει ανάλογα με το πρόβλημα.

Για το SSR είναι το πολύ $n \times \sum_{i=1}^{n} a'_i \times \sum_{i=1}^{n} a'_i$ όπου για το άθροισμα ισχύει:

$$\begin{aligned}
\sum_{i=1}^{n} a'_i = \sum_{i=1}^{n-1} a'_i + a'_n &\leq \frac{\sum_{i=1}^{n-1} a_i + a_n}{K} && \text{(από ορισμό } a'_i\text{)} \\
&\leq \frac{\sum_{i=1}^{n-1} a_i + \sum_{i=1}^{n-1} a_i}{K} && \text{(αλλιώς δεν κατασκευάζω τον πίνακα)} \\
&\leq 2 \cdot n \cdot \frac{a_{n-1}}{K} \leq \frac{6 \cdot n^2}{\varepsilon} && \text{(από ορισμό } a'_i \text{ και K)}
\end{aligned}$$

Άρα ο αλγόριθμος είναι FPTAS με πολυπλοκότητα $O(\frac{n^6}{\varepsilon^2})$.

Στο Alternating η διαφορά εμφανίζεται στο K και στην πράξη αλλάζει μόνο την τελευταία ανίσωση του αθροίσματος που βγάζει $\sum_{i=1}^{n} a'_i \leq 12 \cdot n^2/\varepsilon$. Αυτό δεν επηρεάζει την πολυπλοκότητα που παραμένει $O(\frac{n^6}{\varepsilon^2})$.

Στο Factor-r έχουμε το ίδιο K με το ESS αλλά αλλάζει αρκετά το μέγεθος του πίνακα. Μπορούμε να δούμε ότι το μέγεθος είναι $n \times \sum_{i=1}^{n} b'_i \times \sum_{i=1}^{n} b'_i$ και ο πίνακας κατασκευάζεται όταν $a_n \leq r \cdot \sum_{i=1}^{n-1} a_i$ που μας δίνει:

$$\begin{aligned}
\sum_{i=1}^{n} b'_i = \sum_{i=1}^{n-1} b'_i + b'_n &\leq \frac{r \cdot \sum_{i=1}^{n-1} a_i + r \cdot a_n}{K} \\
&\leq \frac{r \cdot \sum_{i=1}^{n-1} a_i + r^2 \sum_{i=1}^{n-1} a_i}{K} \leq \frac{2 \cdot r^2 \sum_{i=1}^{n-1} a_i}{K} && (\text{r} \geq 1) \\
&\leq 2 \cdot r^2 \cdot n \cdot \frac{a_{n-1}}{K} \leq \frac{6 \cdot r^2 \cdot n^2}{\varepsilon} && (\text{από ορισμό K})
\end{aligned}$$

Άρα ο αλγόριθμος είναι FPTAS με πολυπλοκότητα $O(\frac{r^4 \cdot n^6}{\varepsilon^2})$.

# Κεφάλαιο 6

# Γενίκευση του ESS problem

Σε αυτό το κεφάλαιο Θα ασχοληθούμε με μια άλλη γενίκευση του κλασικού προβλήματος. Αυτό το πρόβλημα αν και δεν είναι ιδιαιτέρα διαφορετικό από τα προηγούμενα δεν να έχει αναφερθεί στο [7]. Η βασική ιδέα για το συγκεκριμένο πρόβλημα είναι ότι μπορεί η χρίση του i στοιχείου να έχει διαφορετική "αξία" για κάθε σύνολο. Ο ακριβής ορισμός του προβλήματος είναι ο ακόλουθος:

**Πρόβλημα 12** (TwoSets-ESS)**.** *Δοθέντος ενός συνόλου από n ζεύγη θετικών ακεραίων* $A = \{(a_1, b_1), ..., (a_n, b_n)\}$*, υπάρχουν δύο μη κενά και ξένα σύνολα* $S_1, S_2 \subseteq \{1, 2, ..., n\}$ *τέτοια ώστε* $\sum_{i \in S_1} a_i = \sum_{j \in S_2} b_j$ *;*

Αφού ορίσαμε το καινούριο αυτό πρόβλημα είναι καλό να δείξουμε ότι μπορούμε να το συσχετίσουμε με το ESS και κάποιες από τις παραλλαγές του. Είναι αρκετά προφανές ότι το καινούριο πρόβλημα είναι καθαρή επέκταση του ESS problem καθώς αν έχουμε ένα στιγμιότυπο του ESS, $A = \{a_1, a_2, ...a_n\}$ και ορίσουμε είσοδο για το καινούριο πρόβλημα το σύνολο ζευγών $A' = \{(a_1, a_1), ..., (a_n, a_n)\}$ τότε μπορούμε να λύσουμε το ESS problem μέσω του καινούριου προβλήματος. Όμοια αν ορίσουμε ως είσοδο το σύνολο $A' = \{(a_1, r \cdot a_1), (a_2, r \cdot a_2), ...(a_n, r \cdot a_n)\}$ θα μπορούμε να λύσουμε το Factor-r.

Όπως και στα προηγούμενα κεφάλαια εμάς μας ενδιαφέρει να προσεγγίσουμε την βέλτιστη λύση στο αντίστοιχο πρόβλημα βελτιστοποίησης. Αυτό το πρόβλημα ορίζεται ως εξής:

**Πρόβλημα 13** (TwoSets-SSR)**.** *Δοθέντος ενός συνόλου από n ζεύγη θετικών ακεραίων* $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ *ψάχνουμε δύο μη κενά και ξένα σύνολα* $S_1, S_2 \subseteq \{1, 2, ...n\}$ *τέτοια ώστε για οποιαδήποτε* $S'_1$ *και* $S'_2$ *μη κενά και ξένα υποσύνολα του* $\{1, 2, ...n\}$ *να ισχύει:*

$$\max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} b_j}, \frac{\sum_{i \in S_2} b_i}{\sum_{j \in S_1} a_j}\} = \min_{S'_1, S'_2} \Big\{ \max \Big\{ \frac{\sum_{i \in S'_1} a_i}{\sum_{j \in S'_2} b_j}, \frac{\sum_{i \in S'_2} b_i}{\sum_{j \in S'_1} a_j} \Big\} \Big\}$$

Να παρατηρήσουμε ότι ο ορισμός του προβλήματος βελτιστοποίησης μοιάζει με αυτόν του Factor-r (Ορισμός 11) στο προηγούμενο κεφάλαιο οπότε δεν θα τον αναλύσουμε περαιτέρω.

## 6.1 FPTAS για το TwoSets-SSR

Ο αλγόριθμος που θα αναπτύξουμε για να προσεγγίσουμε το πρόβλημα δεν μπορεί να βασιστεί στο μεγαλύτερο στοιχείο, όπως κάναμε στα προηγούμενα, καθώς δεν έχουμε κάποια σχέση μεταξύ των στοιχείων μας η των αθροισμάτων. Για να προσπεράσουμε αυτό το εμπόδιο θα ασχοληθούμε με το μικρότερο από τα μέγιστα κάθε συνόλου. Για παράδειγμα για το στιγμιότυπο:

$$a_1 = b_1 = 1$$
$$a_2 = b_2 = 3$$
$$a_3 = b_3 = 8$$
$$a_4 = b_4 = 13$$

τα σύνολα που μας δίνουν τον βέλτιστο λόγο είναι τα $S_1 = \{4\}$ και $S_2 = \{1, 2, 3\}$ με σύνολα τιμών $A_1 = \{13\}$ και $A_2 = \{1, 3, 8\}$ τότε το στοιχείο που μας ενδιαφέρει είναι το $b_3 = 8$(το μικρότερο από τα 8 και 13). Αυτό ουσιαστικά το κάνουμε για να μην χρειαζόμαστε τα λήμματα που μας έβαζαν κάποιο κάτω όριο στα αθροίσματα καθώς πλέων θα έχουμε κάτω όριο με βάση αυτό το στοιχείο. Η υλοποίηση κατά τα υπόλοιπα θα είναι παρόμοια με αυτή του προηγουμένου κεφαλαίου με προσοχή στο ότι θα πρέπει να ελέγχουμε όλα τα στοιχεία γιατί δεν έχουν κάποια σχέση μεταξύ τους ώστε να μπορώ να ταξινομήσω. Για να ξέρουμε ποια στοιχεία μπορούμε να χρησιμοποιήσουμε θα κατασκευάσουμε κατάλληλους πίνακες. Αρχικά Θα λύσουμε το πρόβλημα υποθέτοντας ότι το στοιχείο που αναφέραμε πριν είναι από τα $a_i$.

**Sub-Algorithm 6.1** Sub-algorithm with $a_i$ as smaller max

**Require:** a set $A$ of n pairs of positive integers $\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$

1: **for** $i = 1$ to $n$ **do**
2: **if** exist $j \neq i$ and $b_j \geq a_i$ **then**
3: **run** Sub-Algorithm 6.2 with **input** $A, i$ and **output** $A'$, $T$, $bound$
4: **run** Sub-Algorithm 6.3 with **input** $A', T, i$ and **output** a set $Set = (Opt_i, S_1^i, S_2^i, f - set_i)$
5: **run** Sub-Algorithm 6.4 with **input** $A', T, bound, i$ and **output** a table $H_f$
6: **run** Sub-Algorithm 6.5 with **input** $A, H, Set$ and **output** $Set_i = (Opt_i, S_1^i, S_2^i, a - set_i)$
7: **else**
8: $Set_i \leftarrow \varnothing$
9: **end if**
10: **end for**
11: **find** $m$ such that $Opt_m = \min\left\{Opt_i | Set_i \neq \varnothing\right\}$
12: **return** $Opt_m$, $S_1^m$, $S_2^m$, $f - set_m$

Ο Sub-Algorithm 6.2 κάνει προσαρμογή των τιμών του συνόλου και επιστέφει επιπλέον τον πίνακα $T$ που μας δείχνει αν μπορούμε να κάνουμε χρήση κάποιου στοιχείου αλλά και μιας τιμής $bound$ η οποία θα μας "φράξει" τους πίνακες που θα φτιάξουμε. Αυτό το υλοποιεί ως εξής:

**Sub-Algorithm 6.2** Make tables with "new" values and categorize them

**Require:** a set $A$ of n pairs of positive integers $\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ and a positive integer i

$K \leftarrow \frac{\varepsilon \cdot a_i}{3 \cdot n}$
$\sum_a \leftarrow 0$
**for** $j = 1$ to $n$ **do**
    $A' \leftarrow A' \cup (\lfloor \frac{a_j}{K} \rfloor, \lfloor \frac{b_j}{K} \rfloor)$
    **if** $a'_j \leq a'_i$ **then**
        $\sum_a \leftarrow \sum_a + a'_j$
    **end if**
**end for**
$T[j, 1] \leftarrow 3 \forall 1 \leq j \leq n$
$T[j, 2] \leftarrow 3 \forall 1 \leq j \leq n$
**for** $j = 1$ to $n$ and $j \neq i$ **do**
    $T[j, 1] \leftarrow 0$ only if $a'_j \leq a'_i$
    $T[j, 1] \leftarrow 3$ only if $a'_j > a'_i$
    $T[j, 2] \leftarrow 0$ only if $b'_j < a'_i$
    $T[j, 2] \leftarrow 1$ only if $a'_i \leq b'_j \leq \sum_a$
    $T[j, 2] \leftarrow 2$ only if $b'_j > \sum_a$
**end for**
$bound \leftarrow 2 \cdot n \cdot \lfloor \frac{a_i}{K} \rfloor$
**return** $A'$, $T$, $bound$

Παρατηρούμε ότι ο πίνακας $T$ παίρνει τιμές 0, 1 ,2 ή 3. Οι τιμές αυτές έχουν τις ακόλουθες ερμηνείες:

Τιμή 0: Το στοιχείο αυτό μπορεί να χρησιμοποιηθεί αλλά δεν μπορεί να είναι μέγιστο.

Τιμή 1: Το στοιχείο αυτό αποτελεί πιθανό μέγιστο αλλά το σύνολο που ανήκει δεν είναι απαραίτητα μονοσύνολο.

Τιμή 2: Το στοιχείο αυτό αποτελεί πιθανό μέγιστο και αν χρησιμοποιηθεί θα πρέπει να είναι μονοσύνολο.

Τιμή 3: Το στοιχείο αυτό δεν μπορεί να χρησιμοποιηθεί.
Να υπενθυμίσουμε ότι όλα αυτά που λέμε έχουν ως δεδομένη την χρήση του i στοιχείου ως το μικρότερο από τα δυο μέγιστα. Η χρησιμότητα του Sub-Algorithm 6.3 είναι να επιστρέφει τα σύνολα και τον λόγο που μας ενδιαφέρει όταν το μέγιστο στοιχείο έχει χαρακτηριστεί από τον προηγούμενο αλγόριθμο με την τιμή "2". Από όλες τις επιλογές ο υπο-αλγόριθμος γυρίζει αυτή που είναι πιο κοντά στην βέλτιστη τιμή.

**Sub-Algorithm 6.3** Find best ratio with use of the ”2” numbers

**Require:** a set $A'$ of n pairs of positive integers $\{(a'_1, b'_1), (a'_2, b'_2), ..., (a'_n, b'_n)\}$ a table $T$ and a positive integer i

1: $S \leftarrow i$,
2: **for** $j = 1$ to $n$ **do**
3: **if** $T[j, 1] = 0$ **then**
4: $S \leftarrow S \cup j$
5: **end if**
6: **end for**
7: $S_1 \leftarrow \{1, 2, ..., n\} \setminus i$, $S_2 \leftarrow i$
8: **if** $a'_i < \sum_{j \in S_1} b'_j$ **then**
9: $Opt = \frac{\sum_{j \in S_1} b'_j}{a'_i}$, $a - set = 2$
10: **else**
11: $S_2 \leftarrow \{1, 2, ..., n\} \setminus i$, $S_1 \leftarrow i$
12: $Opt \leftarrow \frac{a'_i}{\sum_{j \in S_2} b'_j}$, $a - set \leftarrow 1$
13: **end if**
14: **for** $j = 1$ to $n$ **do**
15: **if** $T[j, 2] = 2$ and $\frac{b'_j}{\sum_{k \in S \setminus j} a'_k} < Opt$ **then**
16: $S_1 \leftarrow j$, $S_2 \leftarrow S \setminus j$, $Opt \leftarrow \frac{b'_j}{\sum_{k \in S \setminus j} a'_k}$, $a - set \leftarrow 2$
17: **end if**
18: **end for**
19: $Set \leftarrow (Opt, S_1, S_2, a - set)$
20: **return** $Set$

Ένα πρώτο σχόλιο που πρέπει να γίνει είναι ότι μεταξύ των γραμμών 8 και 13 αρχικοποιούμε τις μεταβλητές που θα επιστρέψουμε τελικά κυρίως για να μπορούμε να κάνουμε σύγκριση του λόγου Opt αλλά και για να είμαστε σίγουροι ότι θα επιστρέψει κάτι ο υπο-αλγόριθμος. Πρακτικά δεν μας επηρεάζει η τιμή αυτή καθώς σίγουρα θα συγκριθεί με κάποια εντός του διαστήματος που θέλουμε. Ο επόμενος υπο-αλγόριθμος κατασκευάζει ένα πίνακα $H_f$. Εκεί θέλουμε να εμφανίζονται όλα τα δυνατά ζεύγη προσαρμοσμένων αθροισμάτων που προέρχονται από σύνολα με μέγιστα το i και ένα από τα στοιχεία που χαρακτηρίστηκε ως ”1” από τον Sub-Algorithm 6.2. Ακριβέστερα:

**Sub-Algorithm 6.4** Compute table $H_f$

**Require:** a set $A'$ of n pairs of positive integers $\{(a'_1, b'_1), (a'_2, b'_2), ..., (a'_n, b'_n)\}$ a table $T$ and two positive integers $bound$, $i$

1: **for all** $j = 0$ to $n$ **and** $x, y \in \{1, 2, ..., bound\}$ **do**
2: $H[j, x, y, k] \leftarrow \vec{\emptyset}$
3: **end for**

4: $H[0, a'_i, 0, 0] \leftarrow (\{i\}, \emptyset)$
5: **for** $j = 1$ to $n$ **do**
6: **for all** $x, y \leq bound$ **do**
7: $H[j, x, y, 0] \leftarrow H[j-1, x, y, 0]$
8: $H[j, x, y, 1] \leftarrow H[j-1, x, y, 1]$
9: **if** $H[j-1, x, y, 0] \neq \vec{\emptyset}$ **then**
10: $(S_1, S_2) \leftarrow H[j-1, x, y, 0]$
11: **if** $T[j, 1] = 0$ **and** $x + a'_j \leq bound$ **then**
12: $H[j, x + a'_j, y, 0] \leftarrow (S_1 \cup \{j\}, S_2)$
13: **end if**
14: **if** $T[j, 2] = 0$ **and** $y + b'_j \leq bound$ **then**
15: $H[j, x, y + b'_j, 0] \leftarrow (S_1, S_2 \cup \{j\})$
16: **end if**
17: **if** $T[j, 2] = 1$ **then**
18: $H[j, x, y + b'_j, 1] \leftarrow (S_1, S_2 \cup \{j\})$
19: **end if**
20: **end if**
21: **if** $H[j-1, x, y, 1] \neq \vec{\emptyset}$ **then**
22: $(S_1, S_2) \leftarrow H[j-1, x, y, 1]$
23: **if** $T[j, 1] = 0$ **and** $x + a'_j \leq bound$ **then**
24: $H[j, x + a'_j, y, 1] \leftarrow (S_1 \cup \{j\}, S_2)$
25: **end if**
26: **if** $T[j, 2] = 0$ **and** $y + b'_j \leq bound$ **then**
27: $H[j, x, y + b'_j, 1] \leftarrow (S_1, S_2 \cup \{j\})$
28: **end if**
29: **if** $T[j, 2] = 1$ **then**
30: $H[j, x, y + b'_j, 1] \leftarrow (S_1, S_2 \cup \{j\})$
31: **end if**
32: **end if**
33: **end for**
34: **end for**
35: **return** $H$

Μία σημαντική παρατήρηση είναι ότι δεν καλύπτουμε όλα τα δυνατά προσαρμοσμένα ζεύγη αθροισμάτων (λόγο του $bound$) αλλά θα δούμε, κατά την απόδειξη, ότι αυτά που υπερβαίνουν το μέγεθος του πίνακα δεν θα μας έδιναν κάποιο καλύτερο λόγο για το συγκεκριμένο i.

**Sub-Algorithm 6.5** Find the final ratio for the i

**Require:** a set $A$ of n pairs of positive integers $\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ a table $H$ and a set $Set = (Opt, S_1, S_2, a - set)$

1: **find** $x_0$, $y_0$ such that $\max\{\frac{x_0}{y_0}, \frac{y_0}{x_0}\} = \min\left\{\max\{\frac{x}{y}, \frac{y}{x}\} | H[n, x, y, 1] \neq \vec{\emptyset}\right\}$

2: $(S_A, S_B) \leftarrow H[n, x_0, y_0]$
3: **if** $Opt \leq max\{\frac{\sum_{i \in S_A} a_i}{\sum_{j \in S_B} b_j}, \frac{\sum_{j \in S_B} b_j}{\sum_{i \in S_A} a_i}\}$ **then**
4: **return** $Set$
5: **else if** $\sum_{i \in S_A} a_i \geq \sum_{j \in S_B} b_j$ **then**
6: **return** $\frac{\sum_{i \in S_A} a_i}{\sum_{j \in S_B} b_j}, S_A, S_B, 1$
7: **else**
8: **return** $\frac{\sum_{j \in S_B} b_j}{\sum_{i \in S_A} a_i}, S_B, S_A, 2$
9: **end if**

Ο τελευταίος υπο-αλγόριθμος επιλέγει ποιον λόγο και ποια σύνολα θα επιστρέψει για το συγκεκριμένο i. Με αυτόν έχουμε ολοκληρώσει την περίπτωση που το μικρότερο από μέγιστα των δύο συνόλων είναι κάποιο από τα $a_i$. Είναι αρκετά εύκολο να δούμε ότι αλλάζοντας την σειρά που δύνονται τα στοιχεία των ζευγών μπορούμε να βρούμε τα αντίστοιχα σύνολα όταν το μικρότερο από μέγιστα είναι κάποιο $b_i$. Ο τελικός αλγόριθμος τότε είναι:

**Algorithm 6.1** FPTAS for the TwoSets-SSR

**Require:** a set $A$ of n pairs of positive integers $\{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$
1: $B \leftarrow \{(b_1, a_1), (b_2, a_2), ..., (b_n, a_n)\}$
2: **run** Sub-Algorithm 6.1 with **input** $A$ and **output** $Opt_a$, $S_1^a$, $S_2^a$, $a - set$
3: **run** Sub-Algorithm 6.1 with **input** $B$ and **output** $Opt_b$, $S_1^b$, $S_2^b$, $b - set$
4: **if** $Opt_a \leq Opt_b$ **then**
5: **return** $Opt_a$, $S_1^a$, $S_2^a$, $a - set$
6: **else**
7: **if** $b - set = 1$ **then**
8: **return** $Opt_b$, $S_1^b$, $S_2^b$, 2
9: **else**
10: **return** $Opt_b$, $S_1^b$, $S_2^b$, 1
11: **end if**
12: **end if**

Οπότε ο αλγόριθμος μας επιστρέφει τον μικρότερο από τους λόγους, τα σύνολα που τον παράγουν με σειρά αριθμητή-παρονομαστή (λόγο τον συγκρίσεων - επιστροφών στους υπο-αλγορίθμους) και το ποιο σύνολο κάνει χρήση του $r$ για να βγει αυτός ο λόγος.

## 6.2 Απόδειξη ορθότητας

Αρχικά να θυμίσουμε ότι σε κάθε επανάληψη θέλουμε να εξασφαλίσουμε ότι αν ο βέλτιστος συνδυασμός έχει ως μικρότερο μέγιστο την τιμή $a_{n_0}$ ή $b_{n_0}$ θα επιστραφεί συνδυασμός προσαρμοσμένου λόγου καλύτερου από το βέλτιστο. Θα θε-

ωρήσουμε χωρίς βλάβη της γενικότητας ότι το μικρότερο μέγιστο είναι από τις τιμές $a_i$. τώρα μπορούμε να διατυπώσουμε το ακόλουθο λήμμα:

**Λήμμα 6.1.** *Κατά την $n_0$ επανάληψη αν υπάρχουν σύνολα $S_1$, $S_2$ τέτοια ώστε:*

*1.* $a_{n_0} = max\{a_i | i \in S_1\}$

*2.* $b_{n_1} = max\{b_i | i \in S_2\}$, *με* $a_{n_0} \leq b_{n_1} \leq \sum_{i=1}^{n_0} a_i$

*4. Τα* $\sum_{i \in S_1} a_i = x$, $\sum_{j \in S_2} b_j = y$ *είναι μικρότερα του* $bound$

*τότε το κελί $H[n, x, y, 1]$ θα είναι μη κενό.*

*Απόδειξη.* **Ισχυρισμός:** Για κάθε $i = 0$ έως $n$ αν υπάρχει ζεύγος συνόλων $S'_1$, $S'_2$ τέτοια ώστε:

1. $i \geq max\{S'_1 \cup S'_2 \setminus \{n_0\}\}$, (θεωρούμε $max\{\emptyset\} = 0$)
2. $n_0 \in S'_1$ και $a_{n_0} = max\{a_i | i \in S_1\}$
3. Δεν υπάρχει $m \in S_2$ τέτοιο ώστε $T[m, 2] = 2$
4. Τα $\sum_{i \in S'_1} a_i = x$, $\sum_{j \in S'_2} b_j = y$ είναι μικρότερα του $bound$

ισχύει ότι:

- Αν υπάρχει στοιχείο $m_1 \in S'_2$ με $T[m_1, 2] = 1$ το κελί $H[m, x, y, 1]$ είναι μη κενό.
- Αν δεν υπάρχει στοιχείο $m_1 \in S'_2$ με $T[m_1, 2] = 1$ το κελί $H[m, x, y, 0]$ είναι μη κενό.

**Απόδειξη Ισχυρισμού:**

• $i = 0$:

Εδώ ο μόνος συνδυασμός που υπάρχει είναι ο $S'_1 = \{n_0\}$, $S'_2 = \emptyset$ τον οποίο και αποθηκεύει στην αρχή ο αλγόριθμος στο κελί που μας ενδιαφέρει ($H[0, a_{n_0}, 0, 0]$).

• Θα θεωρήσουμε ότι ο ισχυρισμός ισχύει για κάποιο $i < n$ και θα αποδείξουμε ότι ισχύει για $i + 1$.

Έστω δύο σύνολα $S'_1$, $S'_2$ που ικανοποιούν τις υποθέσεις του ισχυρισμού για $i + 1$.

**Περίπτωση 1.** *Αν $i + 1 \notin S'_1 \cup S'_2$ ή $i + 1 = n_0$*

*Τότε για τα $S'_1$, $S'_2$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x, y, 0]$, $H[i, x, y, 1]$ είναι μη κενό. Αφού πάντα μεταφέρομε τα μη κενά κελιά στο επόμενο $i$ σημαίνει ότι το κελί που θέλουμε θα είναι μη κενό (είτε πρέπει να είναι το $H[i + 1, x, y, 0]$ είτε $H[i + 1, x, y, 1]$ αφού ισχύει από το $i$).*

**Περίπτωση 2.** *Αν $i+1 \in S'_1$ (και $i+1 \neq n_0$)*

*Τότε για το ζεύγος $S^*_1 = S'_1 \setminus \{i+1\}$, $S'_2$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x-a_{i+1}, y, 0]$, $H[i, x, y, 1]$ είναι μη κενό. Να παρατηρήσουμε ότι αν για τα $S'_1$, $S'_2$ έχει θέλουμε το κελί $H[i+1, x, y, 0]$ τότε έχουμε το $H[i, x, y, 0]$ από τα $S^*_1$, $S'_2$ ή αντίστοιχα αν θέλουμε $H[i+1, x-a_{i+1}, y, 1]$ τότε έχουμε το $H[i, x-a_{i+1}, y, 1]$ καθώς το στοιχείο $i+1$ δεν επηρεάζει αφού είναι στο $S'_1$. Αφού πάντα μεταφέρομε τα μη κενά κελιά στο επόμενο $i$ σημαίνει ότι το κελί που θέλουμε θα είναι μη κενό.*

**Περίπτωση 3.** *Αν $i+1 \in S'_2$ με $T[i+1, 2] = 0$*

*Τότε για το ζεύγος $S'_1$, $S^*_2 = S'_2 \setminus \{i+1\}$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x, y-b_{i+1}, 0]$, $H[i, x, y, 1]$ είναι μη κενό. Να παρατηρήσουμε ότι αν για τα $S'_1$, $S'_2$ έχει θέλουμε το κελί $H[i+1, x, y, 0]$ τότε έχουμε το $H[i, x, y-b_{i+1}, 0]$ από τα $S'_1$, $S^*_2$ ή αντίστοιχα αν θέλουμε $H[i+1, x, y, 1]$ τότε έχουμε το $H[i, x, y-b_{i+1}, 1]$ καθώς το στοιχείο $i+1$ δεν επηρεάζει αφού $T[i+1, 2] = 0$. Άρα βάζοντας το στοιχείο $i+1$.*

**Περίπτωση 4.** *Αν $i+1 \in S'_2$ με $T[i+1, 2] = 1$*

*Τότε για το ζεύγος $S'_1$, $S^*_2 = S'_2 \setminus \{i+1\}$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x, y-b_{i+1}, 0]$, $H[i, x, y-b_{i+1}, 1]$ είναι μη κενό. Να παρατηρήσουμε ότι δεν έχει σημασία ποιο από τα κελιά $H[i, x, y-b_{i+1}, 0]$ και $H[i, x, y-b_{i+1}, 1]$ είναι κενό αφού όταν $T[i+1, 2] = 1$ προσθέτουμε στο δεύτερο σύνολο το στοιχείο $i+1$ και γεμίζουμε πάντα το $H[i+1, x, y, 1]$.*

Τέλος αφού κάθε ζεύγος συνόλων που περιγράφει το λήμμα πληρεί τις υποθέσεις του ισχυρισμού για $i = n$ σημαίνει ότι ένα εκ των $H[n, x, y, 0]$, $H[n, x, y, 1]$ είναι μη κενό. Ακριβέστερα εξ υποθέσεως υπάρχει $b_{n_1} = max\{b_i | i \in S_2\}$ , με $a_{n_0} \leq b_{n_1} \leq \sum_{i=1}^{n_0} a_i$ το οποίο σημαίνει ότι $T[n_1, 2] = 1$ και άρα έχουμε το $H[n, x, y, 1]$ να είναι μη κενό. □

Θα συνεχίσουμε την απόδειξη κάνοντας χρήση του θεωρήματος 4.1.

**Θεώρημα 6.1.** *Αν ο Algorithm 6.1 επιστρέφει δύο σύνολα $S_1$, $S_2$ και $S_{1Opt}$, $S_{2Opt}$ τα σύνολα που μας δίνουν το βέλτιστο λόγο, τότε ισχύει:*

$$\max\{\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} b_j}, \frac{\sum_{j\in S_2} b_j}{\sum_{i\in S_1} a_i}\} \in [1, (1+\varepsilon)\cdot \max\{\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} b_j}, \frac{\sum_{j\in S_{2Opt}} b_j}{\sum_{i\in S_{1Opt}} a_i}\}]$$

*Απόδειξη.* Θεωρούμε όπως και στα προηγούμενα $S_{1Opt}$, $S_{2Opt}$ τα σύνολα της βέλτιστης λύσης του προβλήματος και $S_1$, $S_2$ τα σύνολα που επιστρέφει ο αλγόριθμος κατά την επανάληψη $n_0$ που έχει επιλέξει το σωστό, μικρότερο, μέγιστο στοιχείο ($a_{n_0}$ ή $b_{n_0}$). Έστω, χωρίς βλάβη της γενικότητας, ότι το μικρότερο από τα μέγιστα είναι το $a_{n_0}$. Οποιαδήποτε πιθανή λύση είναι υποσύνολο του συνόλου $\{c_1, c_2, ..., c_{2\cdot n}\}$ (όπου $c_i = a_i$ για $i \leq n$ και $c_i = b_{i-n}$ για $i > n$). Από κατασκευή, κάθε σύνολο έχει πληθικότητα μικρότερη του n και αφού το μικρότερο από τα μέγιστα είναι το $a_{n_0}$ τότε και τα δύο αθροίσματα είναι μεγαλύτερα από αυτό.

Αυτό σημαίνει ότι στην $n_0$ επανάληψη για $w = a_{n_0}$ και $m = n$ ισχύουν οι σχέσεις για το $\delta$ και για την βέλτιστη λύση αλλά και για τα σύνολα που επιστρέφουμε από κατασκευή. Άρα μας μένει να αποδείξουμε την σχέση των λόγων. Για να το πετύχουμε αυτό θα χωρίσουμε σε περιπτώσεις για το μέγιστο στοιχείο της βέλτιστης λύσης (έστω $b_{n_1}$).

**Περίπτωση 1.** *Αν στην $n_0$ επανάληψη $T[n_1, 2] = 2$*

*Τότε η καλύτερη επιλογή συνόλων που έχουν ως μέγιστο το $b'_{n_1}$ και μικρότερο μέγιστο το $a'_{n_0}$ είναι η $S_1 = n_1$ και $S_2 = \{i | a'_i \leq a_{n_0}, i \neq n_1\}$ καθώς αφού $T[n_1, 2] = 2$ έχουμε ότι $b'_{n_1} > \sum_{i \in S} a'_i$ και για κάθε $S'_1$, $S'_2$ που αποτελούν πιθανά βέλτιστα (με $n_1 \in S'_1$ και $n_0 \in S'_2$) έχω αναγκαστικά $S'_1 \supseteq S_1$ και$S'_2 \subseteq S_2$ το οποίο μας δίνει:*

$$\begin{aligned}
\max\{\frac{\sum_{i \in S_1} b'_i}{\sum_{j \in S_2} a'_j}, \frac{\sum_{j \in S_2} a'_j}{\sum_{i \in S_1} b'_i}\} &= \frac{\sum_{i \in S_1} b'_i}{\sum_{j \in S_2} a'_j} \\
&\leq \frac{\sum_{i \in S'_1} b'_i}{\sum_{j \in S'_2} a'_j} \\
&= \max\{\frac{\sum_{i \in S'_1} b'_i}{\sum_{j \in S'_2} a'_j}, \frac{\sum_{j \in S'_2} a'_j}{\sum_{i \in S'_1} b'_i}\}
\end{aligned}$$

*και άρα ο λόγος που θα επιστραφεί είναι μικρότερος ή ίσος από τον προσαρμοσμένο βέλτιστο (αφού τα $S'_1$, $S'_2$ τυχαία).*

**Περίπτωση 2.** *Αν στην $n_0$ επανάληψη $T[n_1, 2] = 1$*

*Τότε από όλους τους συνδυασμούς που δημιουργούνται στον πίνακα επιστρέφουμε τον μικρότερο ο οποίος είναι και μεγαλύτερος του ένα. Αν ο συνδυασμός των συνόλων του βέλτιστου λόγου είναι εντός του πίνακα τότε ισχύει η ανίσωση. Άρα μένει να δούμε τι γίνεται στην περίπτωση που ένα εκ΄των $\sum_{i \in S_{1Opt}} b'_i, \sum_{j \in S_{2Opt}} a'_j$ είναι μεγαλύτερο από το μέγεθος του πίνακα. Ακριβέστερα, είμαστε στην $n_0$ επανάληψη με μικρότερο μέγιστο το $a_{n_0}$ και μέγιστο το $b_{n_1}$ (για την βέλτιστη λύση). Αφού ο πίνακας έχει ως μέγεθος στις διαστάσεις των αθροισμάτων την τιμή $2 \cdot n \cdot a'_i$ και χρησιμοποιεί για το γέμισμα τα $b'_i$ για τα οποία $T[i, 2] = 0$ ή 1 τότε όλα τα αθροίσματα που έχουν ως μέγιστο το $b_{n_1}$ περιέχουν στοιχεία $b'_i \leq b_{n_1} \leq \sum_{a_i \leq a_{n_0}} a'_i$ και άρα μπορώ να αφαιρέσω στοιχεία από το σύνολο, έστω $S$ το καινούριο σύνολο, χωρίς να αφαιρέσω το $n_1$ ώστε να ισχύει ότι το άθροισμα είναι εντός του πίνακα και μεγαλύτερο τις τιμής $\sum_{a_i \leq a_{n_0}} a'_i$. Άρα όποια και να είναι τα $S_{1Opt}$ και $S_{2Opt}$ έχουμε ότι το ζεύγος $S_1$, $S$ έχει καλύτερο λόγο (και είναι εντός του πίνακα) αφού:*

$$\sum_{i \in S_{1Opt}} a'_i \leq \sum_{a'_i \leq a'_{n_0}} a'_i \leq \sum_{j \in S} b'_j \leq \sum_{j \in S_{2Opt}} b'_j$$

*που σημαίνει ότι πληρούμε πάντα την σχέση των λόγων.*

Άρα και στις δύο περιπτώσεις επιστρέφουμε σύνολα και λόγο που πληρούν τις προϋποθέσεις του θεωρήματος 4.1 (για το σύνολο $C = \{c_1, c_2, ..., c_{2 \cdot n}\}$ όπως το

ορίσαμε πιο πάνω), και άρα:

$$\max\{\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} b_j}, \frac{\sum_{j\in S_2} b_j}{\sum_{i\in S_1} a_i}\} \in [1, (1+\varepsilon)\cdot \max\{\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} b_j}, \frac{\sum_{j\in S_{2Opt}} b_j}{\sum_{i\in S_{1Opt}} a_i}\}]$$

□

Μας μένει να σχολάσουμε την πολυπλοκότητα του αλγορίθμου. Το σημαντικό κομμάτι είναι η κατασκευή του πίνακα και το πόσες φορές αυτή πραγματοποιείτε. Γεμίζουμε τον πίνακα το πολύ $2 \cdot n$ φορές, ενώ για το γέμισμα χρειαζόμαστε μια διαπέρασε όλου του πίνακα. Από αυτά συμπεραίνουμε ότι η πολυπλοκότητα είναι ίση με τη διάσταση του πίνακα πολλαπλασιασμένη με το n. Η διάσταση του πίνακα είναι $2 \times n \times bound \times bound$ όπου το $bound = 2 \cdot n \cdot \lfloor \frac{a_i}{K} \rfloor$. Γνωρίζοντας ότι $K = \frac{\varepsilon \cdot a_i}{3 \cdot n}$ έχουμε:

$$bound = 2 \cdot n \cdot \lfloor \frac{a_i}{K} \rfloor \leq \frac{2 \cdot n \cdot a_i}{K} + 2 = \frac{6 \cdot n^2}{\varepsilon} + 2 \cdot n$$

και άρα η πολυπλοκότητα είναι $O(\frac{n^6}{\varepsilon^2})$.

# Κεφάλαιο 7

# Μέθοδος διαφοράς αθροισμάτων

Στο κεφάλαιο αυτό θα αναπτύξουμε προσεγγιστικό αλγόριθμο με καλύτερη πολυπλοκότητα και ως προς το πλήθος των στοιχείων n και ως προς το $1/\varepsilon$. Αρχικά θα δούμε πως θα εφαρμοστεί στο αρχικό πρόβλημα (SSR problem) και στην συνέχεια θα την γενικεύσουμε και στα άλλα προβλήματα. Η ιδέα πίσω από τον καινούριο αλγόριθμο είναι να κρατάμε στον πίνακα αντί για τα αθροίσματα την διαφορά τους. Αυτό σε συνδυασμό με την παρατήρηση ότι από τα ζεύγη ακεραίων με ίδια διαφορά αυτό με το μέγιστο άθροισμα μας δίνει τον μικρότερο λόγο είναι ο σκελετός του καινούριου αλγόριθμου. Την ισχύ του παραπάνω ισχυρισμού θα την δούμε στο λήμμα 7.2.

## 7.1 Αλγόριθμοι για το SSR

### 7.1.1 Ψευδοπολυωνυμικός αλγόριθμος

Ο καινούριος ψευδοπολυωνυμικός αλγόριθμος για το SSR βασίζεται στην κατασκευή ενός πίνακα που κρατάει τις διαφορές των αθροισμάτων ζευγών υποσυνόλων. Έτσι για κάθε ζεύγος συνόλων έχουμε μόνο μία τιμή και άρα μία λιγότερη διάσταση στον πίνακα από ότι στις προηγούμενες προσεγγίσεις μας. Η ακριβής κατασκευή του πίνακα φαίνεται παρακάτω.

**Sub-Algorithm 7.1** Find best ratio with specific l

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ and a integer $l < n$

1: $set \leftarrow \vec{\emptyset}$
2: $m = l + 1$
3: $sum \leftarrow \sum_{i=1}^{l} a_i$
4: **while** $m \leq n$ **and** $a_m \leq sum$ **do**
5: $\quad m = m + 1$
6: **end while**

**if** $m \leq n$ **then**
    $Opt \leftarrow \frac{a_m}{\sum_{i=1}^{l} a_i}$
    $set \leftarrow (\{m\}, \{1, 2, ..., l\})$
**else**
    $Opt \leftarrow \frac{a_n}{a_l}$
    $set \leftarrow (\{n\}, \{l\})$
**end if**
$B = 2 \cdot sum$
$H[i, k] \leftarrow \vec{\emptyset}$, $\forall i \in \{0, 1, 2, ..., n\}$ and $k \in \{-B, ..., -10, 1, ..., B\}$
$H[0, -a_l] \leftarrow (\emptyset, \{l\}, a_l, 0, a_l)$
$k_1 \leftarrow 0$, $k_2 \leftarrow 0$
**for all** $i = 1$ to $l$ **do**
    **for** $k = -B$ to $B$ **do**
        **if** $H[i-1, k] \neq \vec{\emptyset}$ **then**
            $H[i, k] \leftarrow H[i-1, k]$
            (if $H[i, k] \neq \vec{\emptyset}$ we use the one with bigest "third value")
            $(S_1, S_2, sum, sum_1, sum_2) \leftarrow H[i-1, k]$
            $k_1 \leftarrow a_i + sum_1 - sum_2$
            $k_2 \leftarrow -a_i + sum_1 - sum_2$
            **if** $i \neq l$ **and** $-B \leq k_1 \leq B$ **then**
                $H[i, k_1] \leftarrow (S_1 \cup \{i\}, S_2, sum + a_i, sum_1 + a_i)$
                (if $H[i, k_1] \neq \vec{\emptyset}$ we use the one with bigest "third value")
            **end if**
            **if** $i \neq l$ **and** $-B \leq k_2 \leq B$ **then**
                $H[i, k_2] \leftarrow (S_1, S_2 \cup \{i\}, sum + a_i, sum_1, sum_2 + a_i)$
                (if $H[i, k_2] \neq \vec{\emptyset}$ we use the one with bigest "third value")
            **end if**
        **end if**
    **end for**
**end for**
**for** $i = l + 1$ to $m - 1$ **do**
    **for** $k = -B$ to $B$ **do**
        $(S_1, S_2, sum, sum_1, sum_2) \leftarrow H[l, k]$
        $H[i, sum_1 + a_i - sum_2] \leftarrow (S_1 \cup \{i\}, S_2, sum + a_i, sum_1 + a_i, sum_2)$
        **if** $H[i-1, k] \neq \vec{\emptyset}$ **and** $i - 1 \neq l$ **then**
            $H[i, k] \leftarrow H[i-1, k]$
            (if $H[i, k] \neq \vec{\emptyset}$ we use the one with bigest "third value")
            $(S_1, S_2, sum, sum_1, sum_2) \leftarrow H[i-1, k]$
            $k_1 \leftarrow a_i + sum_1 - sum_2$
            **if** $i \neq l$ **and** $-B \leq k_1 \leq B$ **then**
                $H[i, k_1] \leftarrow (S_1 \cup \{i\}, S_2, sum + a_i, sum_1 + a_i)$
                (if $H[i, k_1] \neq \vec{\emptyset}$ we use the one with bigest "third value")
            **end if**

```
            end if
        end for
    end for
    for k = −B to B do
        if H[n, k] ≠ ∅⃗ then
            (S1, S2, sum, sum1, sum2) ← H[n, k]
            if Opt ≥ max{sum1/sum2, sum2/sum1} then
                Set ← (S1, S2)
                Opt ← max{sum1/sum2, sum2/sum1}
            end if
        end if
    end for
    return Opt,Set
```

Ο παραπάνω αλγόριθμος λύνει ακριβώς το SSR με την επιπλέον συνθήκη ότι το $l$ στοιχείο είναι το μικρότερο από τα μέγιστα των συνόλων μας. Να σχολιάσουμε ότι όταν δεν υπάρχει στοιχείο $a_m > \sum_{i=1}^{l} a_i$ τότε το $m$ έχει την τιμή $n+1$ και αποθηκεύουμε το συνδυασμό συνόλων $S_1 = n$, $S_2 = l$ χωρίς να μας ενδιαφέρει η τιμή του λόγου γιατί αφού αποτελεί πιθανή λύση και ο συνδυασμός θα υπάρχει στον πίνακα δεν θα επηρεάσει το αποτέλεσμα. Οπότε με την βοήθεια του παραπάνω αλγόριθμου λύνουμε ακριβώς το SSR problem ως εξής:

**Algorithm 7.1** New algorithm for SSR

**Require:** a set of n positive integers $A = \{a_1, a_2, ..., a_n\}$

1: **sort** $A$
2: **for** i=1 to n-1 **do**
3: **run** Sub-Algorithm 7.1 with **input** $A, i$ and **output** $Opt_i$, $set_i$= ($S_1^i$, $S_2^i$)
4: **end for**
5: **find** k such that $Opt_k = min_{i=1ton-1} Opt_i$
6: **return** $(Opt_k, S_1^k, S_2^k)$

Η ορθότητα αυτού του αλγορίθμου θα αποδειχθεί μαζί με την ορθότητα του FPTAS της επόμενης παραγράφου καθώς έχουν πολλά κοινά στοιχεία.

### 7.1.2 FPTAS

Στο νέο προσεγγιστικό αλγόριθμο θα κάνουμε χρήση του Sub-Algorithm 7.1. Για να καταφέρουμε ο καινούριος αλγόριθμος να τρέχει σε πολυωνυμικό χρόνο θα προσαρμόσουμε τις τιμές του συνόλου όπως έχουμε κάνει και στα προηγούμενα κεφάλαια. Η προσαρμογή γίνετε με βάση το στοιχείο l κάθε φορά όπως αυτή φαίνεται παρακάτω:

**Sub-Algorithm 7.2** Make the ”new” values

**Require:** a sorted set $A$ of n positive integers $\{a_1, a_2, ..., a_n\}$ a integer $l < n$ and a number $\varepsilon \in (0, 1)$

1: $\delta \leftarrow \frac{\varepsilon \cdot a_l}{3 \cdot n}$
2: $A' \leftarrow \vec{\emptyset}$
3: **for** $j = 1$ to $n$ **do**
4: $\quad a'_j \leftarrow \lfloor \frac{a_j}{\delta} \rfloor$
5: $\quad A' \leftarrow A' \cup a'_j$
6: **end for**
7: **return** $A'$

Τέλος πρέπει να βρούμε το βέλτιστο λόγο για οποιοδήποτε $l$ και αυτό γίνεται ως εξής:

**Algorithm 7.2** New FPTAS for the SSR

**Require:** a set of n positive integers $A = \{a_1, a_2, ..., a_n\}$

1: **sort** $A$
2: **for** $i = 1$ to $n - 1$ **do**
3: $\quad$ **run** Sub-Algorithm 7.2 with **input** $A, i, \varepsilon$ and **output** $A'$
4: $\quad$ **run** Sub-Algorithm 7.1 with **input** $A', i$ and **output** $Opt_i \; set = (S_1^i, S_2^i)$
5: $\quad Opt_i \leftarrow max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}, \frac{\sum_{j \in S_2} a_j}{\sum_{i \in S_1} a_i}\}$
6: **end for**
7: **find** k such that $Opt_k = min\big\{Opt_i | i \in \{1, ..., n-1\}\big\}$
8: **return** $(Opt_k, S_1^k, S_2^k)$

Οπότε ο αλγόριθμος μας επιστρέφει τον μικρότερο από τους λόγους και τα σύνολα που τον παράγουν με σειρά αριθμητή-παρονομαστή.

### 7.1.3 Απόδειξη ορθότητας

**Λήμμα 7.1.** *Κατά την $n_0$ επανάληψη αν υπάρχουν σύνολα $S_1$, $S_2$ τέτοια ώστε:*

*1.* $n_0 = max\{S_2\}$

*2.* $n_1 = max\{S_1\}$, *με* $a_{n_0} \leq a_{n_1} \leq \sum_{i=1}^{n_0} a_i$

*4. Τα* $\sum_{i \in S_1} a_i = x$, $\sum_{j \in S_2} a_j = y$ *είναι μικρότερα του* $B$

*τότε το κελί $H[n, x - y]$ θα είναι μη κενό.*

*Απόδειξη.* **Ισχυρισμός 1:** Για κάθε $i = 0$ έως $n_0$ αν υπάρχει ζεύγος συνόλων $S'_1$,

$S'_2$ τέτοια ώστε:

1. $i \geq max\{S'_1 \cup S'_2 \setminus \{n_0\}\}$, (θεωρούμε $max\{\emptyset\} = 0$)
2. $n_0 = max\{S_2\}$
3. Τα $\sum_{i \in S'_1} a_i = x$, $\sum_{j \in S'_2} b_j = y$ είναι μικρότερα του $B$

ισχύει ότι το κελί $H[i, x-y]$ είναι μη κενό.

**Απόδειξη ισχυρισμού 1:**

• $i = 0$:

Εδώ ο μόνος συνδυασμός που υπάρχει είναι ο $S'_1 = \emptyset$, $S'_2 = \{n_0\}$ τον οποίο και αποθηκεύει στην αρχή ο αλγόριθμος στο κελί που μας ενδιαφέρει ($H[0, a_{n_0}]$).

• Θα θεωρήσουμε ότι ο ισχυρισμός ισχύει για κάποιο $i < n_0$ και θα αποδείξουμε ότι ισχύει για $i+1$.

Έστω δύο σύνολα $S'_1$, $S'_2$ που ικανοποιούν τις υποθέσεις του ισχυρισμού για $i+1$.

**Περίπτωση 1.** *Αν $i+1 \notin S'_1 \cup S'_2$ ή $i+1 = n_0$*

*Τότε για τα $S'_1$, $S'_2$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το κελί $H[i, x-y]$ είναι μη κενό. Αφού πάντα μεταφέρομε τα μη κενά κελιά στο επόμενο $i$ σημαίνει ότι το κελί που θέλουμε θα είναι μη κενό.*

**Περίπτωση 2.** *Αν $i+1 \in S'_1$ (και $i+1 \neq n_0$)*

*Τότε για το ζεύγος $S_1^* = S'_1 \setminus \{i+1\}$, $S'_2$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το κελί $H[i, x-a_{i+1}-y]$ είναι μη κενό. Αυτό σημαίνει ότι εκεί έχει αποθηκεύει δύο σύνολα με την συγκεκριμένη διαφορά αθροισμάτων. Άρα όταν θα βάλει το $a_{i+1}$ στο πρώτο σύνολο (που το κάνει για τα στοιχεία με δείκτη μικρότερο του $n_0$) θα γεμίσει το κελί $H[i+1, x-y]$.*

**Περίπτωση 3.** *Αν $i+1 \in S'_2$ (και $i+1 \neq n_0$)*

*Τότε για το ζεύγος $S'_1$, $S_2^* = S_2 \setminus \{i+1\}$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το κελί $H[i, x-(y-a_{i+1})]$ είναι μη κενό. Αυτό σημαίνει ότι εκεί έχει αποθηκεύει δύο σύνολα με την συγκεκριμένη διαφορά αθροισμάτων. Άρα όταν θα βάλει το $a_{i+1}$ στο δεύτερο σύνολο (που το κάνει για τα στοιχεία με δείκτη μικρότερο του $n_0$) θα γεμίσει το κελί $H[i+1, x-y]$.*

**Ισχυρισμός 1:** Για κάθε $i = n_0 + 1$ έως $n$ αν υπάρχει ζεύγος συνόλων $S'_1$, $S'_2$ τέτοια ώστε:

1. $i \geq n_1 = max\{S'_1\} > n_0$
2. $n_0 = max\{S_2\}$
3. Τα $\sum_{i \in S'_1} a_i = x$, $\sum_{j \in S'_2} b_j = y$ είναι μικρότερα του $B$

ισχύει ότι το κελί $H[i, x-y]$ είναι μη κενό.

**Απόδειξη ισχυρισμού 1:**

• $i = n_0 + 1$:

Θεωρήσουμε ότι υπάρχουν δύο σύνολα $S'_1, S'_2$ που πληρούν τις υποθέσεις. Τότε αναγκαστικά $n_1 = n_0 + 1$ και άρα για τα σύνολα $S_1^* S'_1 \setminus \{n_1\}$, $S'_2$ ισχύουν οι υποθέσεις από τον προηγούμενο ισχυρισμό για $i = n_0$ και άρα το κελί $H[n_0, x - a_{n_1} - y]$ είναι μη κενό. Ο αλγόριθμος για κάθε $i > n_0$ χρησιμοποιεί τα κελιά του $n_0$ και άρα θα δημιουργηθεί η διαφορά που θέλουμε και θα γεμίσει το κελί $H[i, x-y]$ που θέλουμε.

• Θα θεωρήσουμε ότι ο ισχυρισμός ισχύει για κάποιο $n_0 < i < n$ και θα αποδείξουμε ότι ισχύει για $i + 1$.

Έστω δύο σύνολα $S'_1$, $S'_2$ που ικανοποιούν τις υποθέσεις του ισχυρισμού για $i + 1$.

**Περίπτωση 1.** *Αν $i + 1 \notin S'_1 \cup S'_2$*

*Τότε για τα $S'_1, S'_2$ ικανοποιούν τις υποθέσεις του ισχυρισμού για $i$ και άρα το κελί $H[i, x - y]$ είναι μη κενό. Ο αλγόριθμος αν $i > n_0$ μεταφέρει το κελί στο επόμενο $i$ και άρα γεμίζει το κελί $H[i + 1, x - y]$*

**Περίπτωση 2.** *Αν $i + 1 \in S'_1 \cup S'_2$*

*Τότε το $i + 1 \in S'_1$ (λόγο μεγίστου του $S_2$). Τότε τα σύνολα $S_1^* S'_1 \setminus \{n_1\}$, $S'_2$ πληρούν είτε τις υποθέσεις του ισχυρισμού 1 ένα για $n_0$ (αν δεν υπάρχει άλλο στοιχείο μεγαλύτερο του $n_0$) είτε τις υποθέσεις του ισχυρισμού 2 ένα για την τιμή $i$. Άρα σε κάθε περίπτωση το κελί που χρειαζόμαστε είναι μη κενό (είτε πρέπει να γεμίσουμε από το κελί $H[n_0, x - a_{i+1} - y]$ είτε από το κελί $H[i, x - a_{i+1} - y]$) και άρα γεμίζει το $H[i + 1, x - y]$.*

Αφού ολοκληρώσαμε την απόδειξη του δεύτερου ισχυρισμού γίνετε προφανής η ισχύς του λήμματος. □

Να σχολιάσουμε ότι τελικά στα κελιά $H[n, x-y]$ περιέχουν μόνο συνδυασμούς που έχουν μέγιστο μεγαλύτερο του $n_0$ καθώς οι συνδυασμοί που δεν περιέχουν δεν έχουν μεταφερθεί μετά τα $H[n_0, x - y]$ κελιά.

**Θεώρημα 7.1.** *Ο Algorithm 7.1 λύνει ακριβώς το SSR.*

*Απόδειξη.* Το πρώτο πράγμα που θα σχολιάσουμε είναι ότι κατά την $n_0$ επανάληψη στον πίνακα $H[i, dif]$ όσο το $i < n_0$ δημιουργούμε όλες τις διαφορές των ζευγών συνόλων που είναι και τα δύο υποσύνολα του $\{1, 2, ..., n_0\}$ (καθώς πάνα μεταφέρουμε στο επόμενο $i$ ότι έχουμε δημιουργήσει και δεν υπάρχουν περιορισμοί στην εισαγωγή των στοιχείων στα σύνολα). Ενώ για $i > n_0$ βάζουμε στοιχεία μόνο στο σύνολο "$S_1$" και δεν μεταφέρουμε κελιά χωρίς να προσθέσουμε κάποιο $i > n_0$. Λόγο αυτού τα κελιά με $i = n$ έχουν όλους τους συνδυασμούς συνόλων $S_1$, $S_2$ με $max\{S_2\} = n_0$ και $max\{S_1\} > n_0$ και με διαφορά εντός πίνακα, δηλαδή μόνο συνδυασμούς που μας ενδιαφέρουν(όπως θα δούμε παρακάτω).

Η απόδειξη ορθότητας του αλγορίθμου έχει ως βάση ιδέα το ακόλουθο λήμμα:

**Λήμμα 7.2.** *Αν έχουμε δύο ζέυγη Θετικών ακαιρέων* $(a, b)$ *και* $(c, d)$ *τέτοια ώστε:*

$$a - b = c - d \geq 0 \text{ και } a + b \geq c + d$$

*τότε ισχύει ότι:*

$$\frac{a}{b} \leq \frac{c}{d}$$

*Απόδειξη.* Από την πρώτη σχέση μπορούμε να συμπεράνουμε τα ακόλουθα:

$$c = a - b + d$$

αν αντικαταστήσουμε το c στη δεύτερη σχέση του λήμματος έχουμε:

$$a + b \geq a - b + d + d \Rightarrow 2 \cdot b \geq 2 \cdot d \Rightarrow b \geq d$$

οπότε μπορούμε να καταλήξουμε στο ζητούμενο:

$$\frac{c}{d} = \frac{a - b + d}{d} = \frac{a - b}{d} + 1 \overset{b \geq d}{\geq} \frac{a - b}{b} + 1 = \frac{a}{b}$$

□

Αυτό μας επιβεβαιώνει ότι για οποιαδήποτε διαφορά δημιουργήθηκε εντός του πινάκα του Sub-Algorithm 7.1 έχει κρατηθεί το ζεύγος συνόλων που μας δίνει τον καλύτερο λόγο. Με βάση το παραπάνω λήμμα θα αποδείξουμε ότι ο Sub-Algorithm 7.1 μας επιστρέφει πάντα τον καλύτερο λόγο για το σύνολο A που του δώσαμε σε συνδυασμό με το ότι το στοιχείο $l$ είναι το μικρότερο από τα μέγιστα των δύο συνόλων.

**Λήμμα 7.3.** *Ο Sub-Algorithm 7.1 με είσοδο A και* $n_0$ *μας επιστρέφει δύο* $S_1$, $S_2$ *μη κενά και ξένα υποσύνολα του A τα οποία με δεδομένο ότι* $n_0 = min\{max\{S_1\}, max\{S_2\}\}$ *μας δίνουν την μικρότερη δυνατή τιμή για την μεταβλητή:*

$$Opt = \max\left\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}, \frac{\sum_{j \in S_2} a_j}{\sum_{i \in S_1} a_i}\right\}$$

*Απόδειξη.* Θα χωρίσουμε την απόδειξη σε περιπτώσεις σχετικά με το μέγιστο των $S_1$, $S_2$. Έστω $n_1 = \max\{S_1 \cup S_2\}$ τότε έχουμε:

**Περίπτωση 1.** *Αν* $a_{n_1} > \sum_{i=1}^{n_0} a_i$

*Σε αυτήν την περίπτωση ο Sub-Algorithm 7.1 στην αρχή βρήκε το μικρότερο* $m \leq n$ *για το οποίο ισχύει* $a_m > \sum_{i=1}^{n_0} a_i$. *Έστω τώρα τυχαία* $S_1' \supseteq n_1$ *με* $i \in S_1' \Rightarrow i \leq n_1$ *και* $S_2' \subseteq \{1, 2, ..., n_0\}$. *Με βάση τα μέγιστα που έχουμε τα* $S_1'$ *και* $S_2'$ *είναι πιθανές λύσεις. Τότε αφού το A είναι ταξινομημένο έχουμε ότι:*

$$\sum_{i \in S_1'} a_i \geq a_{n_1} \geq a_m > \sum_{i=1}^{n_0} a_i \geq \sum_{j \in S_2'} a_j$$

*το οποίο αποδεικνύει ότι για αυτά τα μέγιστα ο λόγος των συνόλων* $\{m\}$ *και* $\{1, 2, ..., n_0\}$ *είναι ο βέλτιστος (και αφού χρησιμοποιείται στις συγκρίσεις του Sub-Algorithm 7.1 σημαίνει ότι επιστρέφουμε αυτόν).*

**Περίπτωση 2.** *Αν* $a_{n_1} \leq \sum_{i=1}^{n_0} a_i$

*Εδώ το πρώτο που θα παρατηρήσουμε είναι ότι τα αθροίσματα του βέλτιστου λόγου δεν μπορούν να ξεπερνάνε την τιμή* $B = 2 \cdot \sum_{i=1}^{n_0}$ *που είναι το μέγεθος του πίνακα. Ακριβέστερα:*

***Ισχυρισμός:*** *Αν* $S_1, S_2$ *τα σύνολα του βέλτιστου λόγου τότε* $\sum_{i \in S_1} a_i$ *και* $\sum_{j \in S_2} a_j$ *είναι μικρότερα του* $2 \cdot \sum_{i=1}^{n_0}$.

*Απόδειξη.* Αυτό δείχνετε εύκολα καθώς αν ένα από τα δύο σύνολα είχε άθροισμα μεγαλύτερο από το B (έστω το $S_1$) τότε θα είχαμε:

$$\sum_{i \in S_1} a_i > 2 \cdot \sum_{i=1}^{n_0} a_i > \sum_{i=1}^{n_0} a_i \geq \sum_{j \in S_2} a_j$$

όπου η τελευταία ανίσωση ισχύει λόγο του ότι $n_0 = min\{max\{S_1\}, max\{S_2\}\}$ το οποίο επιπλέον μας δίνει $n_0 = max\{S_2\}$ και $n_1 = max\{S_1\}$. Αφού όμως όλα τα στοιχεία του $S_1$ είναι μικρότερα ίσα από $\sum_{i=1}^{n_0} a_i$ (εξ υποθέσεως μεγίστου) αφαιρώντας ένα οποιοδήποτε στοιχείο από το $S_1$ (εκτός του $n_1$ για να διατηρούμε τα μέγιστα) θα είχαμε ένα $S_1'$ τέτοιο ώστε

$$\sum_{i \in S_1} a_i > \sum_{i \in S_1'} a_i > \sum_{i=1}^{n_0} a_i \geq \sum_{j \in S_2} a_j$$

και άρα θα είχα συνδυασμό με καλύτερο λόγο πράγμα που είναι άτοπο αφού υποθέσαμε ότι είχαμε το βέλτιστο. Άρα αποδείξαμε τον ισχυρισμό.

□

*Αφού τα σύνολα* $S_1$, $S_2$ *έχουν αθροίσματα μικρότερα τις τιμής* $2 \cdot \sum_{i=1}^{n_0} a_i'$ *σημαίνει ότι η διαφορά οποιουδήποτε ζεύγους υποσυνόλου τους είναι εντός του πίνακα και άρα το κελί που αντιστοιχεί στην διαφορά των αθροισμάτων τους είναι μη κενό. Τέλος αφού η διαφορά υπάρχει και καθ όλη την διάρκεια του αλγορίθμου κρατάμε τους συνδυασμούς με μέγιστο συνολικό άθροισμα από το λήμμα 7.2 έχουμε κρατήσει τον καλύτερο λόγο για την διαφορά αυτή δηλαδή τη βέλτιστη λύση.*

Άρα ο Sub-Algorithm 7.1 επιστρέφει την βέλτιστη λύση με δεδομένο ότι $n_0 = min\{max\{S_1\}, max\{S_2\}\}$ □

Με βάση το παραπάνω λήμμα γίνετε προφανές ότι ο Algorithm 7.1 λύνει ακριβώς το SSR. □

Θα συνεχίσουμε με το να δείξουμε ότι ο Algorithm 7.2 είναι FPTAS.

**Θεώρημα 7.2.** *Αν ο Algorithm 7.2 επιστρέφει δύο σύνολα* $S_1$, $S_2$ *και* $S_{1Opt}$, $S_{2Opt}$ *τα σύνολα που μας δίνουν το βέλτιστο λόγο τότε ισχύει:*

$$\max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} b_j}, \frac{\sum_{j \in S_2} b_j}{\sum_{i \in S_1} a_i}\} \in [1, (1+\varepsilon) \cdot \max\{\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} a_j}, \frac{\sum_{j \in S_{2Opt}} b_j}{\sum_{i \in S_{1Opt}} a_i}\}]$$

*Απόδειξη.*

**Λήμμα 7.4.** *Αν $S_{1Opt}$, $S_{2Opt}$ τα σύνολα που μας δίνουν το βέλτιστο λόγο και $S_1$, $S_2$ τα σύνολα που επιστρέφει ο Sub-Algorithm 7.1 στην $n_0$ επανάληψη (του Algorithm 7.2 όπου $n_0 = \min\left\{\max S_{1Opt}, \max S_{2Opt}\right\}$) τότε*

$$max\{\frac{\sum_{i\in S_{1Opt}} a'_i}{\sum_{j\in S_{2Opt}} a'_j}, \frac{\sum_{j\in S_{2Opt}} a'_j}{\sum_{i\in S_{1Opt}} a'_i}\} \geq max\{\frac{\sum_{i\in S_1} a'_i}{\sum_{j\in S_2} a'_j}, \frac{\sum_{j\in S_2} a'_j}{\sum_{i\in S_1} a'_i}\} \geq 1$$

*Απόδειξη.* Προφανές αφού ο Sub-Algorithm 7.1 επιστρέφει την βέλτιστη λύση για το A' και τα $S_{1Opt}$, $S_{2Opt}$ είναι πιθανές λύσεις για αυτές τις εισόδους (A',$n_0$) □

Τέλος να παρατηρήσουμε ότι στην $n_0$ επανάληψη του Sub-Algorithm 7.1 πληρούνται οι προϋποθέσεις του θεωρήματος 4.1 αφού για τα $a'_i$ έχουμε:

$$a'_i = \lfloor \frac{a_i}{\delta} \rfloor \text{όπου} \delta = \frac{a_{n_0} \cdot \varepsilon}{3 \cdot n}$$

με το $a_{n_0}$ να είναι μικρότερο από οποιοδήποτε άθροισμα είτε λόγο υπόθεσης είτε λόγο κατασκευής και το n μεγαλύτερο από οποιαδήποτε πληθικότητα αφού όλα τα σύνολα είναι υποσύνολα του $\{1, 2, ..., n\}$. ενώ για τους λόγους των αθροισμάτων έχουμε το λήμμα 7.4. Τέλος από το θεώρημα 4.1 και αφού ο Algorithm 7.2 επιστρέφει τα σύνολα $S_1$, $S_2$ που δίνουν τον καλύτερο λόγο από όλες τις επαναλήψεις έχουμε:

$$\text{max}\{\frac{\sum_{i\in S_1} a_i}{\sum_{j\in S_2} a_j}, \frac{\sum_{j\in S_2} a_j}{\sum_{i\in S_1} a_i}\} \in [1, (1+\varepsilon) \cdot \text{max}\{\frac{\sum_{i\in S_{1Opt}} a_i}{\sum_{j\in S_{2Opt}} a_j}, \frac{\sum_{j\in S_{2Opt}} a_j}{\sum_{i\in S_{1Opt}} a_i}\}]$$

□

Για να είναι FPTAS μένει να δούμε αν ο αλγόριθμος είναι πολυωνυμικά φραγμένος ως προς το n και το $\frac{1}{\varepsilon}$. Η διαδικασία η οποία καθορίζει την πολυπλοκότητα του αλγορίθμου είναι όπως και στις προηγούμενες περιπτώσεις το γέμισμα του πίνακα. Η διάσταση του πίνακα στην επανάληψη $n_0$ είναι μικρότερη από $n \times 2 \cdot B$ όπου $B = \sum_{i=1}^{n_0} a'_{n_0} \frac{n^2}{\varepsilon}$ (από ορισμό των $a'_i$ στην $n_0$ επανάληψη) και άρα το κάθε γέμισμα θέλει $O(\frac{n^3}{\varepsilon})$. Αφού η όλη διαδικασία γίνεται n-1 φορές η τελική πολυπλοκότιτα του Algorithm 7.2 είναι $O(\frac{n^4}{\varepsilon})$ και άρα είναι FPTAS.

## 7.2 Αλγόριθμοι για το TwoSets-SSR

Ο αλγόριθμος που θα λύνει το Two Sets SSR δεν θα έχει ιδιαίτερες διαφορές από τον προηγούμενο. Η τροποποίηση που υπάρχει είναι λόγο του ότι επειδή δεν μπορούμε να ταξινομήσουμε πλήρως(μας ενδιαφέρει η σειρά των τιμών κάθε ζεύγους) θα κατασκευάσουμε πίνακα που μας δείχνει ποια στοιχεία μπορούμε να χρησιμοποιήσουμε κάθε φορά(όπως κάναμε στο κεφάλαιο 6)

### 7.2.1 Ψευδοπολυωνυμικός αλγόριθμος

Όπως και στο SSR θα λύσουμε αρχικά το πρόβλημα για δεδομένο ελάχιστο μέγιστο στοιχείο. Ακριβέστερα αν η είσοδος είναι $A = \{(a_1, b_1), ..., (a_n, b_n)\}$ και το μικρότερο από τα μέγιστα θέλουμε να παίρνει τιμές από τα $a_i$ αρχικά θα φτιάξουμε ένα πίνακα $T_A$ που θα δείχνει ποιες τιμές μπορούμε να χρησιμοποιήσουμε και έπειτα θα λύσουμε το πρόβλημα για το δεδομένο $a_i$. Αυτά υλοποιούνται ως εξής:

**Sub-Algorithm 7.3** Make table of differences using element l

**Require:** a sorted by first values set $A$ of n pairs of positive integers $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$, and an integer $l \leq n$

1: $sum \leftarrow \sum_{i=1}^{l} a_i$
2: $T[i] \leftarrow 0$ for all $i = 1$ to $n$
3: **for** $j = 1$ to $n$ **do**
4: **if** $b_j > sum$ or $i = l$ **then**
5: $T[i] \leftarrow 2$
6: **else if** $b_j \geq a_l$ **then**
7: $T[i] \leftarrow 1$
8: **else**
9: $T[i] \leftarrow 0$
10: **end if**
11: **end for**
12: **return** $T$

Να σχολιάσουμε ότι δεν χρειάζεται να αποθηκεύσουμε κάτι για τα $a_i$ αφού αυτά είναι ταξινομημένα ενώ για τα $b_i$ έχουμε στον T την τιμή 0 για όσα είναι μικρότερα από το $a_l$ που μας ενδιαφέρει, την τιμή 1 για όσα είναι μεγαλύτερα από το $a_l$ και θα χρησιμοποιηθούν ως πιθανά μέγιστα στον πίνακα τον διαφορών και την τιμή 2 για όσα δεν θα χρησιμοποιηθούν στον πίνακα τον διαφορών. Από τα στοιχεία που έχουν την τιμή 2 όλα είναι πιθανά μέγιστα (ως μονοσύνολα) εκτός του $b_l$ το οποίο δεν μπορεί να χρησιμοποιηθεί αφού θα κάνουμε χρίση του $a_l$. Ο βέλτιστος λόγος με χρίση του $a_l$ ως μικρότερο μέγιστο καθώς και τα σύνολα που τον δημιουργούν βρίσκονται από τον παρακάτω αλγόριθμο.

**Sub-Algorithm 7.4** Best ratio with l and "2" values

**Require:** a sorted by first values set $A$ of n pairs of positive integers $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$, a table T and an integer $l \leq n$

1: $S_1 \leftarrow \{l\}$
2: $S_2 \leftarrow \{1, 2, ..., n\} \setminus l$
3: $set \leftarrow (S_1, S_2)$
4: $Opt \leftarrow \max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} b_j}, \frac{\sum_{j \in S_2} b_j}{\sum_{i \in S_1} a_i}\}$

5: **for** $k = 1$ to $n$ **do**
6: **if** $T[k] = 2$ **and** $k \neq l$ **then**
7: $S_1 \leftarrow \{1, 2, ..., l\} \setminus \{k\}$ , $S_2 \leftarrow \{k\}$
8: **if** $Opt > \frac{\sum_{i \in S_2} b_i}{\sum_{j \in S_1} a_j}$ **then**
9: $Opt \leftarrow \frac{\sum_{i \in S_1} b_i}{\sum_{j \in S_2} a_j}$
10: $set \leftarrow (S_1, S_2)$
11: **end if**
12: **end if**
13: **end for**
14: **return** $Opt$,$Set$

Εδώ πρέπει να σχολιάσουμε ότι αν υπάρχει $b_i > \sum_{y=1}^{l} a_y$ με $i \neq l$ τότε η αρχικοποίηση που κάνουμε στα σύνολα είναι πιθανή λύση και δεν θα επηρεάζει το αποτέλεσμα. Από την άλλη, αν δεν υπάρχει τέτοια τιμή, οτιδήποτε επιστρέψουμε τελικά δεν θα μας επηρεάσει γιατί αν δεν το θέλουμε θα το ξεφορτωθούμε κάνοντας συγκρίσεις για το μέγιστο.

**Sub-Algorithm 7.5** Best ratio with $a_l$ as smallest max

**Require:** a sorted by first values set $A$ of n pairs of positive integers $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ and an integer $l \leq n$

1: **run** Sub-Algorithm 7.3 with **input** $A, l$ and **output** a table $T_A$
2: **run** Sub-Algorithm 7.4 with **input** $A, T_A, l$ and **output** $Opt$, $set$= ($S_1$, $S_2$)
3: $B = 2 \cdot \sum_{i=1}^{l} a_i$
4: $H[i, k, j] \leftarrow \vec{\emptyset} \forall i \in \{0, 1, 2, ..., n\}, k \in \{-B, .., -1, 0, 1, .., B\}$ and $j \in \{0, 1\}$
5: $H[0, -a_l, 0] \leftarrow (\emptyset, \{l\}, a_l, 0, a_l)$
6: $k_1 \leftarrow 0$, $k_2 \leftarrow 0$
7: **for** $i = 1$ to $n$ **do**
8: **for** $k = -B$ to $B$ **do**
9: $H[i, k, 0] \leftarrow H[i - 1, k, 0]$
10: $H[i, k, 1] \leftarrow H[i - 1, k, 1]$
11: **if** $H[i - 1, k, 0] \neq \vec{\emptyset}$ **and** $i \neq l$ **then**
12: $(S_1, S_2, sum, sum_1, sum_2) \leftarrow H[i - 1, k, 0]$
13: $k_1 \leftarrow a_i + sum_1 - sum_2$
14: $k_2 \leftarrow sum_1 - sum_2 - b_i$
15: **if** $k_1 \leq B$ **then**
16: $H[i, k_1, 0] \leftarrow (S_1 \cup \{i\}, S_2, sum + a_i, sum_1 + a_i, sum_2)$
17: (if $H[i, k_1] \neq \vec{\emptyset}$ we use the one with bigest "third value")
18: **end if**
19: **if** $T[i] = 0$ **and** $-B \leq k_2 \leq B$ **then**
20: $H[i, k_2, 0] \leftarrow (S_1, S_2 \cup \{i\}, sum + b_i, sum_1, sum_2 + b_i)$

(if $H[i,k_2,0] \neq \vec{\emptyset}$ we use the one with bigest "third value")
**end if**
**if** $T[i] = 1$ **and** $-B \leq k_2 \leq B$ **then**
$H[i,k_2,1] \leftarrow (S_1, S_2 \cup \{i\}, sum + b_i, sum_1, sum_2 + b_i)$
(if $H[i,k_2,1] \neq \vec{\emptyset}$ we use the one with bigest "third value")
**end if**
**end if**
**if** $H[i-1,k,1] \neq \vec{\emptyset}$ **and** $i \neq l$ **then**
$(S_1, S_2, sum, sum_1, sum_2) \leftarrow H[i-1,k,1]$
$k_1 \leftarrow a_i + sum_1 - sum_2$
$k_2 \leftarrow sum_1 - sum_2 - b_i$
**if** $-B \leq k_1 \leq B$ **then**
$H[i,k_1,1] \leftarrow (S_1 \cup \{i\}, S_2, sum + a_i, sum_1 + a_i, sum_2)$
(if $H[i,k_1] \neq \vec{\emptyset}$ we use the one with bigest "third value")
**end if**
**if** $T[i] \neq 2$ **and** $-B \leq k_2 \leq B$ **then**
$H[i,k_2,1] \leftarrow (S_1, S_2 \cup \{i\}, sum + b_i, sum_1, sum_2 + b_i)$
(if $H[i,k_2,1] \neq \vec{\emptyset}$ we use the one with bigest "third value")
**end if**
**end if**
**end for**
**end for**
**for** $k = -B$ to $B$ **do**
**if** $H[n,k,1] \neq \vec{\emptyset}$ **then**
$(S_1, S_2, sum, sum_1, sum_2) \leftarrow H[n,k,1]$
**if** $Opt > \max\{\frac{sum_1}{sum_2}, \frac{sum_2}{sum_1}\}$ **then**
$Opt \leftarrow \max\{\frac{sum_1}{sum_2}, \frac{sum_2}{sum_1}\}$
$set \leftarrow (S_1, S_2)$
**end if**
**end if**
**end for**
**return** $set$

Εδώ όπως και πριν η αρχικοποίηση που κάνουμε (λόγο κλήσης του αλγοριθμου) δεν επηρεάζει το πρόβλημα και τελικός επιστρέφουμε τα σύνολα του βέλτιστου λόγου για το συγκεκριμένο $a_l$. Μένει να λύσουμε το πρόβλημα για όλα τα $a_i$ και $b_i$ και από όλα τα αποτελέσματα να κρατήσουμε το καλύτερο.

**Algorithm 7.3** New algorithm for Two Sets SSR

**Require:** a set $A$ of n pairs of positive integers $A = \{(a_1,b_1),(a_2,b_2),...,(a_n,b_n)\}$

1: **sort** $A$ by $a_i$
2: $B \leftarrow \{(b_1, a_1), (b_2, a_2), ..., (b_n, a_n)\}$
3: **sort** B by $b_i$
4: $Opt \leftarrow \max\{\frac{a_1}{b_2}, \frac{b_2}{a_1}\}$
5: $set \leftarrow (\{1\}, \{2\}$
6: **for** $i = 1$ to $2 \cdot n$ **do**
7: **if** $i \leq n$ **then**
8: **run** Sub-Algorithm 7.5 with **input** $A, i$ and **output** $set_i$= ($S_1^i$, $S_2^i$)
9: **else**
10: $y = i - n$
11: **run** Sub-Algorithm 7.5 with **input** $B, y$, and **output** $set_i$= ($S_2^i$, $S_1^i$)
12: **end if**
13: **if** $set_i \neq \vec{\emptyset}$ **and** $Opt > max\{\frac{\sum_{k \in S_1^i} a_k}{\sum_{l \in S_2^i} b_l}, \frac{\sum_{l \in S_2^i} b_l}{\sum_{k \in S_1^i} a_k}\}$ **then**
14: $Opt \leftarrow max\{\frac{\sum_{k \in S_1^i} a_k}{\sum_{l \in S_2^i} b_l}, \frac{\sum_{l \in S_2^i} b_l}{\sum_{k \in S_1^i} a_k}\}$
15: $set \leftarrow set_i$
16: **end if**
17: **end for**
18: **return** $Opt, set$

### 7.2.2 FPTAS

Εδώ όπως και στο SSR θα προσαρμόσουμε τις τιμές του συνόλου και με τις καινούριες τιμές θα τρέξουμε κάποιους από τους αλγορίθμους τις ψευδο-πολυωνυμικής λύσης. Οι τιμές προσαρμόζονται ως εξής:

**Sub-Algorithm 7.6** Make the ”new” values

**Require:** a sorted by first values set $A$ of n pairs of positive integers $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ a integer $l < n$ and a number $\varepsilon \in (0, 1)$
1: $\delta \leftarrow \frac{\varepsilon \cdot a_l}{3 \cdot n}$
2: $A' \leftarrow \vec{\emptyset}$
3: **for** $j = 1$ to $n$ **do**
4: $a'_j \leftarrow \lfloor \frac{a_j}{\delta} \rfloor$
5: $b'_j \leftarrow \lfloor \frac{b_j}{\delta} \rfloor$
6: $A' \leftarrow A' \cup (a'_j, b'_j)$
7: **end for**
8: **return** $A'$

Τώρα αφού έχουμε τον αλγόριθμο που δημιουργεί τις καινούριες τιμές μπορούμε να προσαρμόσουμε τον ψευδο-πολυωνυμικό αλγόριθμο ώστε να τρέχει τους υπο-αλγορίθμους για τις κατάλληλες τιμές κάθε φορά.

**Algorithm 7.4** FPTAS for Two Sets SSR

**Require:** a set $A$ of n pairs of positive integers $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$

**sort** $A$ by $a_i$
$A_2 \leftarrow \{(b_1, a_1), (b_2, a_2), ..., (b_n, a_n)\}$
**sort** $A_2$ by $b_i$
$Opt \leftarrow \max\{\frac{a_1}{b_2}, \frac{b_2}{a_1}\}$
$set \leftarrow (\{1\}, \{2\}$
**for** $i = 1$ to $2 \cdot n$ **do**
    **if** $i \leq n$ **then**
        **run** Sub-Algorithm 7.6 with **input** $A, i, \varepsilon$ and **output** a table $A'$
        **run** Sub-Algorithm 7.5 with **input** $A', T_A, i, set$ and **output** $set_i$= ($S_1^i$, $S_2^i$)
    **else**
        $y = i - n$
        **run** Sub-Algorithm 7.6 with **input** $A_2, i, \varepsilon$ and **output** a table $A'_2$
        **run** Sub-Algorithm 7.5 with **input** $A'_2, y$ and **output** $set_i$= ($S_2^i$, $S_1^i$)
    **end if**
    **if** $set_i \neq \vec{\emptyset}$ **and** $Opt > max\{\frac{\sum_{k \in S_1^i} a_k}{\sum_{l \in S_2^i} b_l}, \frac{\sum_{l \in S_2^i} b_l}{\sum_{k \in S_1^i} a_k}\}$ **then**
        $Opt \leftarrow max\{\frac{\sum_{k \in S_1^i} a_k}{\sum_{l \in S_2^i} b_l}, \frac{\sum_{l \in S_2^i} b_l}{\sum_{k \in S_1^i} a_k}\}$
        $set \leftarrow set_i$
    **end if**
**end for**
**return** $Opt, set$

### 7.2.3 Απόδειξη ορθότητας

Αρχικά να θυμίσουμε ότι σε κάθε επανάληψη θέλουμε να εξασφαλίσουμε ότι αν ο βέλτιστος συνδυασμός έχει ως μικρότερο μέγιστο την τιμή $a_{n_0}$ ή $b_{n_0}$ θα επιστραφεί συνδυασμός προσαρμοσμένου λόγου καλύτερου από το βέλτιστο. Θα θεωρήσουμε χωρίς βλάβη της γενικότητας ότι το μικρότερο μέγιστο είναι από τις τιμές $a_i$. τώρα μπορούμε να διατυπώσουμε το ακόλουθο λήμμα:

**Λήμμα 7.5.** *Κατά την $n_0$ επανάληψη αν υπάρχουν σύνολα $S_1$, $S_2$ τέτοια ώστε:*

*1.* $a_{n_0} = max\{a_i | i \in S_1\}$

*2.* $b_{n_1} = max\{b_i | i \in S_2\}$, *με* $a_{n_0} \leq b_{n_1} \leq \sum_{i=1}^{n_0} a_i$

*4. Τα* $\sum_{i \in S_1} a_i = x$, $\sum_{j \in S_2} b_j = y$ *είναι μικρότερα του* $B$

*τότε το κελί* $H[n, x - y, 1]$ *θα είναι μη κενό.*

*Απόδειξη.* **Ισχυρισμός:** Για κάθε $i = 0$ έως $n$ αν υπάρχει ζεύγος συνόλων $S'_1$, $S'_2$ τέτοια ώστε:

1. $i \geq max\{S'_1 \cup S'_2 \setminus \{n_0\}\}$, (θεωρούμε $max\{\emptyset\} = 0$)
2. $n_0 \in S'_1$ και $a_{n_0} = max\{a_i | i \in S_1\}$
3. Δεν υπάρχει $m \in S_2$ τέτοιο ώστε $T[m] = 2$
4. Τα $\sum_{i \in S'_1} a_i = x$, $\sum_{j \in S'_2} b_j = y$ είναι μικρότερα του $B$

ισχύει ότι:

- Αν υπάρχει στοιχείο $m_1 \in S'_2$ με $T[m_1] = 1$ το κελί $H[i, x-y, 1]$ είναι μη κενό.
- Αν δεν υπάρχει στοιχείο $m_1 \in S'_2$ με $T[m_1] = 1$ το κελί $H[i, x-y, 0]$ είναι μη κενό.

**Απόδειξη Ισχυρισμού:**

• $i = 0$:

Εδώ ο μόνος συνδυασμός που υπάρχει είναι ο $S'_1 = \{n_0\}$, $S'_2 = \emptyset$ τον οποίο και αποθηκεύει στην αρχή ο αλγόριθμος στο κελί που μας ενδιαφέρει ($H[0, a_{n_0}, 0]$).

• Θα θεωρήσουμε ότι ο ισχυρισμός ισχύει για κάποιο $i < n$ και θα αποδείξουμε ότι ισχύει για $i + 1$.

Έστω δύο σύνολα $S'_1$, $S'_2$ που ικανοποιούν τις υποθέσεις του ισχυρισμού για $i + 1$.

**Περίπτωση 1.** *Αν $i + 1 \notin S'_1 \cup S'_2$ ή $i + 1 = n_0$*

*Τότε για τα $S'_1$, $S'_2$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x-y, 0]$, $H[i, x-y, 1]$ είναι μη κενό. Αφού πάντα μεταφέρομε τα μη κενά κελιά στο επόμενο $i$ σημαίνει ότι το κελί που θέλουμε θα είναι μη κενό (είτε πρέπει να είναι το $H[i+1, x-y, 0]$ είτε $H[i+1, x-y, 1]$ αφού ισχύει από το $i$).*

**Περίπτωση 2.** *Αν $i + 1 \in S'_1$ (και $i + 1 \neq n_0$)*

*Τότε για το ζεύγος $S^*_1 = S'_1 \setminus \{i + 1\}$, $S'_2$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x - a_{i+1} - y, 0]$, $H[i, x - a_{i+1} - y, 1]$ είναι μη κενό. Να παρατηρήσουμε ότι αν για τα $S'_1$, $S'_2$ θέλουμε το κελί $H[i + 1, x - y, 0]$ τότε έχουμε το $H[i, x - a_{i+1} - y, 0]$ από τα $S^*_1$, $S'_2$ ή αντίστοιχα αν θέλουμε $H[i + 1, x - y, 1]$ τότε έχουμε το $H[i, x - a_{i+1} - y, 1]$ καθώς το στοιχείο $i + 1$ δεν επηρεάζει αφού είναι στο $S'_1$. Για να γεμίσει το κελί που θέλουμε πρέπει να πούμε ότι, το κελί που είναι μη κενό έχει σύνολα με διαφορά $x - a_{i+1} - y$ και άρα όταν βάλουμε το $a_{i+1}$ στο πρώτο σύνολο (γίνεται πάντα για τα στοιχεία που μπορούν να μπουν στο $S_1$, δηλαδή $i + 1 < n_0$) θα έχουν διαφορά $x - y$ και άρα γεμίζει το $H[i + 1, x - y, 0]$*

**Περίπτωση 3.** *Αν $i + 1 \in S'_2$ με $T[i + 1] = 0$*

*Τότε για το ζεύγος $S_1'$, $S_2^* = S_2' \setminus \{i+1\}$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x-y+b_{i+1}, 0]$, $H[i, x-y, 1]$ είναι μη κενό. Να παρατηρήσουμε ότι αν για τα $S_1'$, $S_2'$ έχει θέλουμε το κελί $H[i+1, x-y, 0]$ τότε έχουμε το $H[i, x-y+b_{i+1}, 0]$ από τα $S_1'$, $S_2^*$ ή αντίστοιχα αν θέλουμε $H[i+1, x-y, 1]$ τότε έχουμε το $H[i, x-y+b_{i+1}, 1]$ καθώς το στοιχείο $i+1$ δεν επηρεάζει αφού $T[i+1] = 0$. Άρα βάζοντας το στοιχείο $i+1$ η διαφορά δημιουργείται όπως αναφέραμε και στην προηγούμενη περίπτωση. Άρα το*

**Περίπτωση 4.** *Αν $i+1 \in S_2'$ με $T[i+1] = 1$*

*Τότε για το ζεύγος $S_1'$, $S_2^* = S_2' \setminus \{i+1\}$ ισχύουν οι υποθέσεις του ισχυρισμού για το $i$ και άρα το ένα εκ των κελιών $H[i, x-y+b_{i+1}, 0]$, $H[i, x-y+b_{i+1}, 1]$ είναι μη κενό. Να παρατηρήσουμε ότι δεν έχει σημασία ποιο από τα κελιά $H[i, x-y+b_{i+1}, 0]$ και $H[i, x-y+b_{i+1}, 1]$ είναι κενό αφού όταν $T[i+1] = 1$ προσθέτουμε στο δεύτερο σύνολο το στοιχείο $i+1$ γεμίζουμε πάντα το $H[i+1, x-y, 1]$ (όπως και πριν δεν έχει σημασία αν είχαμε ακριβώς τα σύνολα που αναφέραμε αποθηκευμένα).*

Τέλος αφού κάθε ζεύγος συνόλων που περιγράφει το λήμμα πληρεί τις υποθέσεις του ισχυρισμού για $i = n$ σημαίνει ότι ένα εκ των $H[n, x-y, 0]$, $H[n, x-y, 1]$ είναι μη κενό. Ακριβέστερα εξ υποθέσεως υπάρχει $b_{n_1} = max\{b_i | i \in S_2\}$ , με $a_{n_0} \leq b_{n_1} \leq \sum_{i=1}^{n_0} a_i$ το οποίο σημαίνει ότι $T[n_1] = 1$ και άρα έχουμε το $H[n, x-y, 1]$ να είναι μη κενό. □

**Θεώρημα 7.3.** *Ο Algorithm 7.3 λύνει ακριβώς το Two Sets SSR.*

*Απόδειξη.* Πριν από οτιδήποτε άλλο να σχολιάσουμε ότι όλες οι αρχικοποιήσεις που έχουμε κάνει δεν επηρεάζουν την επιστροφή της βέλτιστης λύσης. Ας υποθέτουμε ότι τα σύνολα $S_1$, $S_2$ μας δίνουν την βέλτιστη λύση. Χωρίς βλάβη της γενικότητας θα θεωρήσουμε ότι το μικρότερο μέγιστο είναι από τις τιμές $a_i$ και ότι τα ζεύγη είναι ταξινομημένα κατά αύξοντα τρόπο προς τα $a_i$. Έστω $a_{n_0} = max\{a_i | i \in S_1\}$ και $b_{n_1} = max\{b_j | j \in S_2\}$ (με $a_{n_0} \leq b_{n_1}$). Θα χωρίσουμε περιπτώσεις με βάση την τιμή του $b_{n_0}$.

**Περίπτωση 1.** *Αν $b_{n_1} > \sum_{i=1}^{n_0} a_i$*

*Αρχικά να παρατηρήσουμε ότι αφού τα $a_i$ είναι ταξινομημένα τότε για το σύνολο $S_1$ ισχύει ότι $S_1 \subseteq \{1, 2, ..., n_0\} \setminus \{n_1\}$ (το $n_1$ ανήκει εξ υποθέσεως στο $S_2$). Αν συμβολίσουμε $S_A$ ένα τυχαίο υποσύνολο του $\{1, 2, ..., n_0\} \setminus \{n_1\}$ και $S_B$ ένα τυχαίο υπερσύνολο του $\{n_1\}$ με $b_{n_1} \geq b_i$ για κάθε $i \in S_B$ τότε ο συνδυασμός $S_A$, $S_B$ αποτελεί πιθανό βέλτιστο. Από τα παραπάνω έχουμε τις ανισότητες:*

$$\sum_{j \in S_B} b_j \geq b_{n_1} > \sum_{i=1}^{n_0} a_i \geq \sum_{i \in \{1,2,...,n_0\} \setminus \{n_1\}} a_i \geq \sum_{i \in S_A} a_i$$

*και άρα η βέλτιστη λύση είναι αυτή των συνόλων $S_1 = \{1, 2, ..., n_0\} \setminus \{n_1\}$ , $S_2 = \{n_1\}$. Αυτός είναι ένας από τους συνδυασμούς που δημιουργούνται στον Sub-Algorithm 7.4 και αφού είναι ο βέλτιστος θα είναι και αυτός που θα κρατηθεί από τις συγκρίσεις.*

**Περίπτωση 2.** *Αν* $b_{n_1} \leq \sum_{i=1}^{n_0} a_i$

*Αν για τον βέλτιστο λόγο ισχύει ότι και τα δύο αθροίσματα είναι μικρότερα η ίσα της τιμής* $B = 2 \cdot \sum_{i=1}^{n_0} a_i$ *από το λήμμα 7.5 το κελί που* $H[n, dif, 1]$ *που αντιστοιχεί στην διαφορά των αθροισμάτων των συνόλων θα είναι μη κενό. Αυτό σε συνδυασμό με το λήμμα 7.2 σημαίνει ότι έχει κρατηθεί ο βέλτιστος λόγος και άρα αυτός θα απομείνει από τις συγκρίσεις. Αυτό σημαίνει ότι πρέπει δείξουμε πως για τον βέλτιστο λόγο ισχύει ότι και τα δύο αθροίσματα είναι μικρότερα η ίσα της τιμής* $B = 2 \cdot \sum_{i=1}^{n_0} a_i$*. Αυτό είναι εύκολο καθώς αν δεν ίσχυε θα είχαμε:*

$$\sum_{i \in S_2} b_i > 2 \cdot \sum_{i=1}^{n_0} a_i > \sum_{i=1}^{n_0} a_i \geq \sum_{j \in S_1} a_j$$

*όπου* $S_1$ *και* $S_2$ *τα σύνολα του βέλτιστου λόγου όπως τα αναφέραμε στην αρχή της απόδειξης. Αφού το μέγιστο του πρώτου αθροίσματος είναι το* $b_{n_1}$ *το οποίο είναι μικρότερο ίσο του* $\sum_{i=1}^{n_0} a_i$ *σημαίνει πως μπορώ να αφαιρέσω οποιοδήποτε άλλο στοιχείο του* $S_2$ *(υπάρχει αφού* $\sum_{i \in S_2} b_i > 2 \cdot \sum_{i=1}^{n_0} a_i$*) χωρίς να χαλάσω τα μέγιστα και να έχω ένα* $S_2'$ *με* $\sum_{i \in S_2'} b_i > \sum_{i=1}^{n_0} a_i$ *(αφού κάθε στοιχείο είναι μικρότερο του* $b_{n_1}$*) και άρα ο λόγος δεν θα ήταν ο βέλτιστος. Άρα τα σύνολα έχουν αθροίσματα μικρότερο του B και όπως προείπαμε αυτό συνεπάγεται επιστροφή του βέλτιστου λόγου.*

□

Μας μένει να δείξουμε ότι ο Algorithm 7.4 είναι FPTAS.

**Θεώρημα 7.4.** *Αν ο Algorithm 7.4 επιστρέφει δύο σύνολα* $S_1$, $S_2$ *και* $S_{1Opt}$, $S_{2Opt}$ *τα σύνολα που μας δίνουν το βέλτιστο λόγο τότε ισχύει:*

$$\max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} b_j}, \frac{\sum_{j \in S_2} b_j}{\sum_{i \in S_1} a_i}\} \in [1, (1+\varepsilon) \cdot \max\{\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} b_j}, \frac{\sum_{j \in S_{2Opt}} b_j}{\sum_{i \in S_{1Opt}} a_i}\}]$$

*Απόδειξη.* Αρχικά να παρατηρήσουμε ότι ο Algorithm 7.4 έχει μια επανάληψη κατά την οποία αν $i \leq n$ θεωρεί μικρότερο μέγιστο το $a_i$ μετατρέπει την είσοδο με βάση αυτό το στοιχείο και λύνει το πρόβλημα με την βοήθεια των άλλων υποαλγορίθμων. Αν το $i$ είναι μεταξύ του $n+1$ και του $2 \cdot n$ τότε κάνει το ίδιο για μικρότερο μέγιστο το $b_{i-n}$. Αυτή η παρατήρηση μας βοηθά να δούμε ότι αν αποδείξουμε το θεώρημα με μικρότερο μέγιστο κάποιο από τα $a_i$ τότε θα ισχύει γενικά. Θα υποθέσουμε οπότε ότι το μικρότερο μέγιστο είναι κάποιο $a_{n_0}$.

**Λήμμα 7.6.** *Αν* $S_{1Opt}$, $S_{2Opt}$ *τα σύνολα που μας δίνουν το βέλτιστο λόγο και* $S_1$, $S_2$ *τα σύνολα που επιστρέφει ο Sub-Algorithm 7.5 στην* $n_0$ *επανάληψη(όπου* $n_0 = max\{S_{1Opt}\}$ *και* $\exists i \in S_{2Opt}$ *τέτοιο ώστε* $b_i \geq a_{n_0}$*) τότε*

$$max\{\frac{\sum_{i \in S_{1Opt}} a_i'}{\sum_{j \in S_{2Opt}} b_j'}, \frac{\sum_{j \in S_{2Opt}} b_j'}{\sum_{i \in S_{1Opt}} a_i'}\} \geq max\{\frac{\sum_{i \in S_1} a_i'}{\sum_{j \in S_2} b_j'}, \frac{\sum_{j \in S_2} b_j'}{\sum_{i \in S_1} a_i'}\} \geq 1$$

*Απόδειξη.* Προφανές αφού ο Sub-Algorithm 7.5 επιστρέφει την βέλτιστη λύση για το A' και τα $S_{1Opt}$, $S_{2Opt}$ είναι πιθανές λύσεις για αυτές τις εισόδους ($A'$,$n_0$) □

Τέλος να παρατηρήσουμε ότι στην $n_0$ επανάληψη πληρούνται οι προϋποθέσεις του θεωρήματος 4.1 αφού για τα $a'_i$ και $b'_i$ έχουμε:

$$a'_i = \lfloor \frac{a_i}{\delta} \rfloor, b'_i = \lfloor \frac{b_i}{\delta} \rfloor \text{όπου} \delta = \frac{a_{n_0} \cdot \varepsilon}{3 \cdot n}$$

με το $a_{n_0}$ να είναι μικρότερο από οποιοδήποτε άθροισμα είτε λόγο υπόθεσης είτε λόγο κατασκευής και το n μεγαλύτερο από οποιαδήποτε πληθικότητα αφού όλα τα σύνολα είναι υποσύνολα του $\{1, 2, ..., n\}$. ενώ για τους λόγους των αθροισμάτων έχουμε το λήμμα 7.6. Τέλος από το θεώρημα 4.1 και αφού ο Algorithm 7.4 επιστρέφει τα σύνολα $S_1$, $S_2$ που δίνουν τον καλύτερο λόγο από όλες τις επαναλήψεις έχουμε:

$$\max\{\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} b_j}, \frac{\sum_{j \in S_2} b_j}{\sum_{i \in S_1} a_i}\} \in [1, (1+\varepsilon) \cdot \max\{\frac{\sum_{i \in S_{1Opt}} a_i}{\sum_{j \in S_{2Opt}} b_j}, \frac{\sum_{j \in S_{2Opt}} b_j}{\sum_{i \in S_{1Opt}} a_i}\}]$$

□

Τέλος όσων αφορά την πολυπλοκότητα μπορούμε να πούμε ότι εξαρτάτε κυρίως από το μέγεθος του πίνακα ($2 \times n \times 2 \cdot B$) και ότι τον κατασκευάζουμε $2 \cdot n$ φορές. Άρα η τελική πολυπλοκότητα είναι πάλι $O(\frac{n^4}{\varepsilon})$ και άρα ο Algorithm 7.4 είναι FPTAS.

## 7.3 Αλγόριθμοι για το Factor-r SSR

Για την λύση του Factor-r SSR δεν θα κατασκευάσουμε κάποιον ψευδο-πολυωνυμικό αλγόριθμο αλλά θα προσεγγίσουμε κατευθείαν κάνοντας κάποιες προσαρμογές στο FPTAS του Two Sets SSR.

### 7.3.1 FPTAS

Η μια αλλαγή που θα κάνουμε στο FPTAS του Two Sets SSR είναι στην προσαρμογή των τιμών καθώς δεν μπορούμε να θεωρήσουμε ζεύγη της μορφής $(a_i, r \cdot a_i)$ αφού το $r \cdot a_i$ δεν είναι απαραίτητα ακέραιος. Ακριβέστερα ο Sub-Algorithm 7.6 θα αντικατασταθεί με τον ακόλουθο:

**Sub-Algorithm 7.7** Make the "new" values

**Require:** a sorted set $A$ of $n$ integers $A = \{a_1, a_2, ..., a_n\}$ a integer $l < n$ real r and a number $\varepsilon \in (0, 1)$

1: $\delta_1 \leftarrow \frac{\varepsilon \cdot a_l}{3 \cdot n}$

2: $\delta_2 \leftarrow \frac{\varepsilon \cdot r \cdot a_l}{3 \cdot n}$
3: $A' \leftarrow \emptyset$, $B' \leftarrow \emptyset$
4: **for** $i = 1$ to $n$ **do**
5: $a'_i \leftarrow \lfloor \frac{a_i}{\delta_1} \rfloor$
6: $b'_i \leftarrow \lfloor \frac{r \cdot a_i}{\delta_1} \rfloor$
7: $A' \leftarrow A' \cup (a'_i, b'_i)$
8: $a''_i \leftarrow \lfloor \frac{a_i}{\delta_2} \rfloor$
9: $b''_i \leftarrow \lfloor \frac{r \cdot a_i}{\delta_2} \rfloor$
10: $B' \leftarrow B' \cup (b''_i, a''_i)$
11: **end for**
12: **return** $A'$, $B'$

Για τον υπολογισμό των βέλτιστων εντός της επανάληψης θα τρέξουμε και πάλι τους Sub-Algorithm 7.3, 7.4 και 7.5 αλλά θα αλλάξουμε τον συνολικό αλγόριθμο.

**Algorithm 7.5** FPTAS for Factor - r

**Require:** a set $A$ of n positive integers $A = \{(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)\}$ a positive parameter $r \geq 0$ and a parameter $\varepsilon \in (0, 1)$

1: **sort** $A$
2: **if** r < 1 **then**
3: $r \leftarrow \frac{1}{r}$
4: **end if**
5: $Opt \leftarrow \frac{r \cdot a_n}{a_1}$
6: $set \leftarrow (\{1\}, \{n\})$
7: **for** $i = 1$ to $n$ **do**
8: **run** Sub-Algorithm 7.7 with **input** $A, i, r, \varepsilon$ and **output** two sets $A'$, $B'$
9: **run** Sub-Algorithm 7.5 with **input** $A'$, $i$ and **output** $set_a$= ($S_1^a$, $S_2^a$)
10: **if** $set_a \neq \vec{\emptyset}$ **and** $Opt > max\{\frac{\sum_{k \in S_1^a} a_k}{\sum_{l \in S_2^a} b_l}, \frac{\sum_{l \in S_2^a} b_l}{\sum_{k \in S_1^a} a_k}\}$ **then**
11: $Opt \leftarrow max\{\frac{\sum_{k \in S_1^a} a_k}{\sum_{l \in S_2^a} b_l}, \frac{\sum_{l \in S_2^a} b_l}{\sum_{k \in S_1^a} a_k}\}$
12: $set \leftarrow set_a$
13: **end if**
14: **run** Sub-Algorithm 7.5 with **input** $B'$, $i$ and **output** $set_b$= ($S_2^b$, $S_1^b$)
15: **if** $set_b \neq \vec{\emptyset}$ **and** $Opt > max\{\frac{\sum_{k \in S_1^b} a_k}{\sum_{l \in S_2^b} b_l}, \frac{\sum_{l \in S_2^b} b_l}{\sum_{k \in S_1^b} a_k}\}$ **then**
16: $Opt \leftarrow max\{\frac{\sum_{k \in S_1^b} a_k}{\sum_{l \in S_2^b} b_l}, \frac{\sum_{l \in S_2^b} b_l}{\sum_{k \in S_1^b} a_k}\}$
17: $set \leftarrow set_b$
18: **end if**
19: **end for**
20: **return** $Opt, set$

Εδώ δεν χρειάζεται να κάνουμε κάποια απόδειξη ορθότητας καθώς η απόδειξη για την ορθότητα του FPTAS του TwoSets-SSR δεν εξαρτάτε από το αν οι αρχικές τιμές είναι ακέραιες και άρα μπορούμε να θεωρήσουμε το Factor-r SSR ως υποπερίπτωση του TwoSets-SSR (με πραγματικές τιμές).

# Κεφάλαιο 8

# Ύπαρξη και κατασκευή FPTAS

## 8.1 Γενίκευση Προβλημάτων

Εδώ, επηρεασμένοι από την δουλειά στο [19], θα παρουσιάσουμε μια αντίστοιχη προσέγγιση για προβλήματα επιλογής τουλάχιστον δύο υποσυνόλων. Η κλάση αυτόν των προβλημάτων περιγράφεται παρακάτω:

**Πρόβλημα 14** (Πρόβλημα επιλογής πολλαπλών υποσυνόλων)**.** *Ένα πρόβλημα επιλογής πολλαπλών υποσυνόλων $\mathcal{P}$ είναι ένα πρόβλημα βελτιστοποίησης του οποίου τα στιγμιότυπα I = (A,w,k,Chk) αποτελούνται από:*

- *ένα σύνολο από n αντικείμενα* $A = \{a_1, a_2, ..., a_n\}$
- *ένα θετικό βάρους* $w(a_i) \forall a_i \in A$
- *το πλήθος k των συνόλων που ψάχνουμε*
- *έναν αλγόριθμο απόφασης "Chk"*

Ο αλγόριθμος απόφασης "Chk" προσδιορίζει για κάθε k-αδα συνόλων $S_1, S_2, ..., S_k$ με $S_i \subseteq \{1, 2, ..., n\}\, \forall i = 1, 2, ..., k$ αν αποτελεί πιθανή λύση του $\mathcal{P}$ ή όχι και επιστέφει τιμές true ή fulse αντίστοιχα. Εδώ θα υποθέσουμε ότι για κάθε σύνολο A και για κάθε αλγόριθμο απόφασης "Chk" υπάρχει τουλάχιστον μια k-αδα που είναι πιθανή λύση. Αν το $\mathcal{P}$ είναι πρόβλημα ελαχιστοποίησης τότε σκοπός είναι να βρούμε μια k-αδα συνόλων $S_1, S_2, ..., S_k$ τέτοια ώστε αν:

$$\sum_{i \in S_1} w(a_i) \leq \sum_{i \in S_j} w(a_i) \leq \sum_{i \in S_k} w(a_i) \text{ , για όλα τα } j = 1, 2, ..., k$$

ο λόγος $\frac{\sum_{i \in S_k} w(a_i)}{\sum_{j \in S_1} w(a_j)}$ να είναι ο ελάχιστος δυνατός. Αντίστοιχα αν το $\mathcal{P}$ είναι πρόβλημα μεγιστοποίησης τότε σκοπός είναι να βρούμε μια k-αδα συνόλων $S_1, S_2, ...,$ $S_k$ τέτοια ώστε:

$$\sum_{i \in S_1} w(a_i) \leq \sum_{i \in S_j} w(a_i) \leq \sum_{i \in S_k} w(a_i) \text{ , για όλα τα } j = 1, 2, ..., k$$

και ο λόγος $\frac{\sum_{i \in S_1} w(a_i)}{\sum_{j \in S_k} w(a_j)}$ να είναι ο μέγιστος δυνατός.

Η πρώτη παρατήρηση αμέσως μετά τον ορισμό είναι ότι έτσι όπως έχουμε ορίσει τα προβλήματα ελαχιστοποίησης και μεγιστοποίησης έχουν ακριβώς την ίδια λύση.

## 8.2 Ύπαρξη-Κατασκευή FPTAS

Πριν διατυπώσουμε το θεώρημα που περιγράφει τις συνθήκες που θέλουμε να πληρούνται για να υπάρχει FPTAS σε ένα τέτοιο πρόβλημα θα περιγράψουμε μια σημαντική συνθήκη η οποία θα θέλουμε να ισχύει. Η συνθήκη αυτή είναι ο καθορισμός εξ αρχής του μικρότερου μέγιστου βάρους των συνόλων. Ακριβέστερα ένα στοιχείο $n_0$ θα λέμε ότι είναι το στοιχείο με το μικρότερο μέγιστο βάρος ή ότι πληρεί την συνθήκη ($C_0$) αν ισχύει ότι $w(a_{n_0}) = min_{S \in \{S_1, S_2, ..., S_k\}} \{max\{w(a_i) | i \in S\}\}$.

**Θεώρημα 8.1.** *Έστω ότι έχουμε ένα πρόβλημα επιλογής πολλαπλών υποσυνόλων $\mathcal{P}$ με στιγμιότυπα I = (A,w,k,Chk). Αν μπορούμε να κατασκευάσουμε αλγόριθμο ο οποίος λύνει το $\mathcal{P}$ όταν:*

- *οι τιμές $w(a_i)$ είναι ακέραιες για κάθε i=1,2,...,n*
- *για δοθέν στοιχείο $n_0 \in \{1, 2, ..., n\}$ πληρείται η συνθήκη $(C_0)$*

*και είναι πολυωνυμικά φραγμένος ως προς το n και το $w(a_{n_0})$ τότε υπάρχει FPTAS για το πρόβλημα $\mathcal{P}$.*

Θα μπορούσε κάποιος να αναρωτηθεί για ποιο λόγο ενώ στο θεώρημα ζητάμε ύπαρξη αλγόριθμου που να λύνει προβλήματα για ακεραίους ως είσοδο εμείς στον ορισμό που δώσαμε δεν απαρτίσαμε κάτι τέτοιο. Αυτό το κάναμε αφενός γιατί ο τρόπος που λύνουμε τα προβλήματα απαιτεί ακέραιες τιμές στο A και αφετέρου γιατί υπάρχουν προβλήματα όπως το Factor-r Sum Subsets optimization problem τα οποία, με τον τρόπο που θα τα αντιστοιχίσουμε στην συγκεκριμένη οικογένεια προβλημάτων, απαιτούν ύπαρξη πραγματικών τιμών στο A.

Θα δούμε πως θα μπορούσε να περιγραφεί, εν μέρη τουλάχιστον, το Factor-r Sum Subsets optimization problem με είσοδο ένα σύνολο ακεραίων $A = \{a_1, ..., a_n\}$ και ένα πραγματικό r. Θα θεωρήσουμε πρόβλημα ελαχιστοποίησης τις παραπάνω οικογένειας με στιγμιότυπο I=(E,w,2,"Chk") όπου $E = \{e_1, ..., e_{2n}\}$ με $w(e_i) = a_i$ για i=1,2,...,n και $w(e_i) = r \cdot a_{i-n}$ για $i = n+1, ..., 2 \cdot n$ και ο "Chk" επιστρέφει true όταν ισχύουν τα ακόλουθα:

1. $S_1 \subseteq \{1, 2, ..., n\}$
2. $S_2 \subseteq \{n+1, n+2, ..., 2 \cdot n\}$
3. $\forall i \in S_1 \nexists j \in S_2$ τέτοιο ώστε $i + n = j$

τα οποία μπορούν να ελεγχθούν σε πολυωνυμικό χρόνο. Αυτό είναι ένα πρόβλημα ελαχιστοποίησης του Factor-r. Ο ορισμός αυτός περιγραφεί το ίδιο ακριβώς πρόβλημα με τον ορισμό 11 πού είδαμε στο κεφάλαιο 5.4. Να παρατηρήσουμε ότι επειδή η τιμή r δεν είναι ακέραια αναγκαζόμαστε να έχουμε βάρη της μορφής $r \cdot a_i$, τα οποία δεν είναι ακέραια. Τώρα θα δούμε την απόδειξη του θεωρήματος:

*Απόδειξη.* Έστω ένα $\varepsilon \in (0,1)$ και ένα πρόβλημα ελαχιστοποίησης $\mathcal{P}$ της οικογένειας του ορισμού 14 με στιγμιότυπο I = (A,w,k,Chk) το οποίο πληρεί τις υποθέσεις του θεωρήματος 8.1. Τότε για m=1,2,...,n κατασκευάζουμε μια παράμετρο μεγέθους(scaling parameter):

$$\delta^{(m)} = \frac{\varepsilon \cdot w(a_m)}{3 \cdot n}$$

Με βάση αυτή την σταθερά κατασκευάζουμε, n το πλήθος, καινούρια στιγμιότυπα $I^{(1)}, I^{(2)}, ..., I^{(n)}$ τα οποία έχουν ίδιο A, k και αλγόριθμο "Chk" αλλά διαφορετικά βάρη. Ακριβέστερα για κάθε m=1,2,...,n το στιγμιότυπο $I^{(m)}$ ισχύει:

$$w^{(m)}(a_i) = \lfloor \frac{w(a_i)}{\delta^{(m)}} \rfloor \forall i = 1, 2, ..., n$$

Αφού για κάθε m=1,2,...,n τα βάρη είναι μόνο ακέραιοι(από κατασκευή) τότε από υπόθεση υπάρχει αλγόριθμος που λύνει το $\mathcal{P}$ για το στιγμιότυπο $I^{(m)}$ με την επιπλέον συνθήκη ότι το στοιχείο m πληρεί την $(C_0)$. Έστω $S_1^{(m)}$, $S_2^{(m)}$,..., $S_k^{(m)}$ οι k-αδες συνόλων που παίρνουμε ως λύσεις για τα αντίστοιχα στιγμιότυπα $I^{(m)}$ και $S_1, S_2, ..., S_k$ τα σύνολα της βέλτιστης λύσης. Θα συμβολίσουμε:

$$Opt^{(m)} = \max \Big\{ \frac{\sum_{i \in S_l^m} w(a_i)}{\sum_{j \in S_{l'}^m} w(a_j)} | l, l' \in \{1, 2, ..., k\}, l \neq l' \Big\}$$

και το λόγο της βέλτιστης λύσης $\frac{\sum_{i \in S_k} w(a_i)}{\sum_{j \in S_1} w(a_j)} = Opt$. Έτσι έχουμε τον ακόλουθο ισχυρισμό:

**Ισχυρισμός.**
Για τους λόγους που προκύπτουν από τις αρχικές τιμές και τα σύνολα των λύσεων των διαφόρων στιγμιοτύπων ισχύει ότι

$$\min_{m=1,2,...n} \{Opt^{(m)}\} \in [1, (1+\varepsilon) \cdot Opt]$$

*Απόδειξη.* Αν $S_1, S_2, ..., S_k$ τα σύνολα της βέλτιστης λύσης και $n_0$ ο ακέραιος για τον οποίο ισχύει $w(a_{n_0}) = min_{S \in \{S_1, S_2, ..., S_k\}} \big\{ max\{w(a_i) | i \in S\} \big\}$ τότε θα δείξουμε ότι ο λόγος $Opt^{(n_0)}$ θα είναι εντός του διαστήματος που μας ενδιαφέρει και άρα θα ισχύει ο ισχυρισμός. Αρχικά να παρατηρήσουμε ότι τα $S_1^{(n_0)}, S_2^{(n_0)}, ..., S_k^{(n_0)}$ είναι η βέλτιστη λύση του $\mathcal{P}$ με στιγμιότυπο το $I^{(n_0)}$ και τον επιπλέον περιορισμό

ότι το στοιχείο $n_0$ πληρεί την συνθήκη ($C_0$). Να παρατηρήσουμε ότι μια πιθανή λύση είναι η $S_1, S_2, ..., S_k$ καθώς πληρεί τις προϋποθέσεις που θέσαμε. Από τον τρόπο που ορίσαμε την λύση προκύπτει ότι:

$$\sum_{i \in S_1^{(n_0)}} w^{(n_0)}(a_i) \leq \sum_{i \in S_j^{(n_0)}} w^{(n_0)}(a_i) \leq \sum_{i \in S_k^{(n_0)}} w^{(n_0)}(a_i), \forall j = 1, 2, ..., k$$

ενώ για τα αρχικά βάρη:

$$\frac{\sum_{i \in S_k^{(n_0)}} w(a_i)}{\sum_{j \in S_1^{(n_0)}} w(a_j)} \leq \max_{l,l' \in \{1,2,...,k\}, l \neq l'} \{\frac{\sum_{i \in S_l^{(n_0)}} w(a_i)}{\sum_{j \in S_{l'}^{(n_0)}} w(a_j)}\}$$

ενώ αντίθετα για τα $S_1, S_2, ..., S_k$ ισχύει:

$$\sum_{i \in S_1} w(a_i) \leq \sum_{i \in S_j} w(a_i) \leq \sum_{i \in S_k} w(a_i), \forall j = 1, 2, ..., k \text{ και}$$

$$\frac{\sum_{i \in S_k} w^{(n_0)}(a_i)}{\sum_{j \in S_1} w^{(n_0)}(a_j)} \leq \max_{l,l' \in \{1,2,...,k\}, l \neq l'} \{\frac{\sum_{i \in S_l} w^{(n_0)}(a_i)}{\sum_{j \in S_{l'}} w^{(n_0)}(a_j)}\}$$

Να παρατηρήσουμε τώρα ότι για οποιοδήποτε συνδυασμό συνόλων από τα $\{S_1, S_2,$ $..., S_k\}$ και $\{S_1^{(n_0)}, S_2^{(n_0)}, ..., S_k^{(n_0)}\}$ έχουμε ότι για το $\delta^{(n_0)}$ ισχύουν οι προϋποθέσεις του θεωρήματος 4.1. Ακριβέστερα όλα τα σύνολα έχουν πληθικότητα μικρότερη του n και τα αθροίσματα που χρησιμοποιούν τις αρχικές τιμές $w(a_i)$ είναι όλα μεγαλύτερα του $w(a_{n_0})$ είτε εξ υποθέσεως ($\{S_1, S_2, ..., S_k\}$) είτε εκ κατασκευής ($\{S_1^{(n_0)}, S_2^{(n_0)}, ..., S_k^{(n_0)}\}$) άρα αν βάλω στο θεώρημα m=n και w=$w(a_{n_0})$ ισχύουν οι σχέσεις για το $\delta$. Αν θέσουμε ως $S_A^{(n_0)}$ και $S_B^{(n_0)}$ τα δυο σύνολα τις k-αδας $\{S_1^{(n_0)}, S_2^{(n_0)}, ..., S_k^{(n_0)}\}$ για τα οποία ισχύει:

$$\frac{\sum_{i \in S_A^{(n_0)}} w(a_i)}{\sum_{j \in S_B^{(n_0)}} w(a_j)} = \max_{l,l' \in \{1,2,...,k\}, l \neq l'} \{\frac{\sum_{i \in S_l^{(n_0)}} w(a_i)}{\sum_{j \in S_{l'}^{(n_0)}} w(a_j)}\}$$

και ως $S_A$ και $S_B$ τα δυο σύνολα τις k-αδας $S_1, S_2, ..., S_k$ για τα οποία ισχύει:

$$\frac{\sum_{i \in S_A} w^{(n_0)}(a_i)}{\sum_{j \in S_B} w^{(n_0)}(a_j)} = \max_{l,l' \in \{1,2,...,k\}, l \neq l'} \{\frac{\sum_{i \in S_l} w^{(n_0)}(a_i)}{\sum_{j \in S_{l'}} w^{(n_0)}(a_j)}\}$$

τότε μπορούμε να πούμε ότι:

$$\frac{\sum_{i \in S_A^{(n_0)}} w^{(n_0)}(a_i)}{\sum_{j \in S_B^{(n_0)}} w^{(n_0)}(a_j)} \leq \frac{\sum_{i \in S_k^{(n_0)}} w^{(n_0)}(a_i)}{\sum_{j \in S_1^{(n_0)}} w^{(n_0)}(a_j)} \leq \frac{\sum_{i \in S_A} w^{(n_0)}(a_i)}{\sum_{j \in S_B} w^{(n_0)}(a_j)} \leq$$

και οπότε για τα ζεύγη συνόλων $S_A^{(n_0)}$, $S_B^{(n_0)}$ και $S_A$, $S_B$ ισχύει η ανίσωση του θεωρήματος 4.1 το οποίο σημαίνει ότι έχουμε:

$$\frac{\sum_{i \in S_A^{(n_0)}} w(a_i)}{\sum_{j \in S_B^{(n_0)}} w(a_j)} \leq (1+\varepsilon) \frac{\sum_{i \in S_A} w(a_i)}{\sum_{j \in S_B} w(a_j)} \leq (1+\varepsilon) \frac{\sum_{i \in S_k} w(a_i)}{\sum_{j \in S_1} w(a_j)}$$

το οποίο σημαίνει ότι έχουμε την ακρίβεια που μας ενδιαφέρει.

□

Αυτό που μας μένει είναι να δείξουμε ότι η όλη διαδικασία είναι πολυωνυμικά φραγμένη ως προς την είσοδο και το $\frac{1}{\varepsilon}$. Ο αλγόριθμος που τρέχουμε για το τυχαίο στιγμιότυπο $I^{(m)}$ έχουμε υποθέσει ότι είναι φραγμένος ως προς το n και την τιμή $w^{(m)}(a_m)$, υπόθεση η οποία λόγο της προσαρμογής των τιμών μας δίνει:

$$w^{(m)}(a_m) = \lfloor \frac{w(a_m)}{\delta^{(m)}} \rfloor \leq \frac{w(a_m)}{\delta^{(m)}} = \frac{3 \cdot n}{\varepsilon}$$

και αφού τρέχουμε για n στιγμιότυπα η συνολική διαδικασία είναι φραγμένη από το n και το $\frac{1}{\varepsilon}$ που σημαίνει ότι έχουμε FPTAS.

□

Μια πολύ σημαντική παρατήρηση είναι ότι η παραπάνω απόδειξη μας παρέχει όχι μόνο ύπαρξη FPTAS υπό αυτές τις συνθήκες αλλά και μια διαδικασία κατασκευής αυτού.

# Κεφάλαιο 9

# Επίλογος

## 9.1 Αποτελέσματα

Κατά την ενασχόληση μας με τα προβλήματα που είδαμε στα προηγούμενα κεφάλαια, πέραν της εξοικείωσης που αποκτήσαμε με τον δυναμικό προγραμματισμό και τις τεχνικές για την δημιουργία FPTAS, καταφέραμε να αναπτύξουμε αλγόριθμο ο οποίος βελτιώνει την πολυπλοκότητα των ήδη υπαρχόντων (στα [2], [18]) για το SSR. Επιπλέον αναπτύξαμε στο κεφάλαιο 7 αλγορίθμους που λύνουν τα Two Sets SSR (το οποίο παρουσιάσαμε στο κεφάλαιο 6) και το Factor-r SSR (λύση με χρήση του αλγορίθμου για το Two Sets SSR). Τα Αποτελέσματα μας εμφανίζονται στους παρακάτω πίνακες.

Αλγόριθμοι για το Subset-sums Ratio

| Αλγόριθμος | Πολυπλοκότητα |
|---|---|
| Ψευδοπολυωνυμικός [2] | $O(nB^2)$, όπου $B = \sum_{i=1}^{n} a_i$ |
| Ψευδοπολυωνυμικός [18] | $O(n^3B^2)$, όπου $B = \sum_{i=1}^{n} a_i$ |
| Ψευδοπολυωνυμικός (κεφ. 2) | $O(n^2B^2)$, όπου $B = \sum_{i=1}^{n} a_i$ |
| Ψευδοπολυωνυμικός (κεφ. 5) | $O(nB^2)$, όπου $B = \sum_{i=1}^{n} a_i$ |
| Ψευδοπολυωνυμικός (κεφ. 7) | $O(n^2B)$, όπου $B = \sum_{i=1}^{l} a_i$ |
| FPTAS [2] | $O(n^6/\varepsilon^4)$ |
| FPTAS [18] | $O(n^7/\varepsilon^2)$ |
| FPTAS (κεφ. 2) | $O(n^7/\varepsilon^2)$ |
| FPTAS (κεφ. 5) | $O(n^6/\varepsilon^2)$ |
| FPTAS (κεφ. 7) | $O(n^4/\varepsilon)$ |

Αλγόριθμοι για παραλαγές του Subset-sums Ratio

| Πρόβλημα | Αλγόριθμος | Πολυπλοκότητα |
|---|---|---|
| Factor-r SSR | Ψευδοπολυωνυμικός (κεφ. 3) | $O(n^2B^2)$, όπου $B = \sum_{i=1}^{n} a_i$ |
| | FPTAS (κεφ. 3) | $O(r^4n^7/\varepsilon^2)$ |
| | FPTAS (κεφ. 5) | $O(r^4n^6/\varepsilon^2)$ |
| | FPTAS (κεφ. 7) | $O(n^4/\varepsilon)$ |
| Alternating SSR | FPTAS (κεφ. 5) | $O(n^6/\varepsilon^2)$ |
| Two Sets SSR | FPTAS (κεφ. 6) | $O(n^6/\varepsilon^2)$ |
| | Ψευδοπολυωνυμικός (κεφ. 7) | $O(n^2B)$, όπου $B = \sum_{i=1}^{l} a_i$ |
| | FPTAS (κεφ. 7) | $O(n^4/\varepsilon)$ |

Μια σημαντική παρατήρηση που μπορεί να γίνει είναι ότι τα Two Sets SSR και Factor-r SSR καταφέραμε να τα προσεγγίσουμε με FPTAS ίδιας πολυπλοκότητας με τον αντίστοιχο του SSR.

## 9.2 Μελλοντική έρευνα

Γενικά οι παραπάνω αλγόριθμοι θα μπορούσαν να γενικευτούν για προβλήματα εύρεσης υποσυνόλων ίσων πληθικοτήτων αν προστεθούν στους αντίστοιχους πίνακες δύο διαστάσεις, μεγέθους $n$ η κάθε μία, όπου θα κρατάμε τις πληθικότητες των συνόλων. Η τεχνική αυτή έχει εφαρμοστεί και στο [7] για την λύση ESS με ίσες πληθικότητες. Όμοια θα μπορούσαμε να κάνουμε κάποιες γενικεύσεις για περισσότερα από 2 υποσύνολα (στους αλγορίθμους των κεφαλαίων 2 έως 6) προσθέτοντας μία επιπλέον διάσταση για κάθε επιπλέον υποσύνολο. Θα ήταν ενδιαφέρων να ερευνηθεί κατά πόσο οι επεκτάσεις αυτές των διαστάσεων είναι αναγκαίες η μπορούμε να βρούμε τεχνικές για να προσεγγίσουμε χρησιμοποιώντας λιγότερες διαστάσεις από τις προφανείς. Πάντα θα υπάρχει η απορία για το αν μπορεί να μειωθεί περαιτέρω η πολυπλοκότητα των ήδη υπαρχόντων αλγορίθμων, ενώ θα ήταν ενδιαφέρων να δούμε αλγορίθμους που να προσεγγίζουν γενικεύσεις του SSR όπως SSR with Exclusions (πρόβλημα βελτιστοποίησης αντίστοιχο του ESS with Exclusions στο [7]) για τις οποίες, γενικεύσεις, δεν μπορούμε να ακολουθήσουμε ακριβώς τις ίδιες μεθόδους με αυτές που εφαρμόσαμε εδώ. Τέλος έχουμε ενδείξεις ότι η μέθοδος του κεφαλαίου 7 μπορεί να εφαρμοστεί και στο Alternating SSR. Η εφαρμογή της μεθόδου στο Alternating SSR μπορεί να αποτελέσει μέρος μελλοντικής έρευνας.

Σημαντικό είναι να αναφερθεί ότι πολλά προβλήματα φαίνεται να σχετίζονται με το ESS και θα ήταν ενδιαφέρον να δει κάποιος το κατά πόσο μπορούμε να βγάλουμε κάποια αποτελέσματα για τα άλλα προβλήματα με βάσει την υπάρχουσα δουλειά (είτε να προσεγγίσουμε κάποια είτε να αποκλείσουμε την ύπαρξη λύσης είτε να απορρέει κάποιο άλλο συμπέρασμα με βάση τα αποτέλεσμα στο ESS ή κάποια παραλλαγή του). Κάποια από αυτά τα προβλήματα είναι το Double Digest, Partial Digest, και το Reconstructing Sets From Interpoint Distances τα οποία είναι προβλήματα που αναζητούν σημεία πάνω σε ευθείς έτσι ώστε να καλύπτουν όλες

τις αποστάσεις που δίνονται στην είσοδο και με διαφορετικούς σκοπούς όπως την εύρεση του ελαχίστου πλήθους σημείων είτε την κάλυψη όλων των αποστάσεων από συγκεκριμένο το πλήθος σημεία. Τα παραπάνω προβλήματα σχετίζονται με το ESS καθώς οι απαντήσεις τους εξαρτώνται από το αν το ESS με είσοδο το σύνολο των αποστάσεων έχει θετική απάντηση και ακόμα περισσότερο από το πόσοι συνδυασμοί δίνουν θετική απάντηση στο ESS. Επιπλέον σχετίζονται με προβλήματα μοριακής βιολογίας (όπως η ταυτοποίηση πρωτεϊνών και η χαρτογράφηση του DNA) για τα οποία έχει γίνει σχετική εύρυνα (μπορεί κάποιος να ανατρέξει στα [5], [8], [9] και [20] για περισσότερες πληροφορίες). Ένα άλλο πρόβλημα το οποίο φαίνεται να σχετίζεται είναι το $[h]$-Sumset Covers problem το οποίο από ένα δοσμένο σύνολο αναζητά υποσύνολο του τέτοιο ώστε να μπορούμε να παράγουμε το αρχικό σύνολο από τα αθροίσματα $h$ στοιχείων του συνόλου που επιστρέφουμε (περισσότερες πληροφορίες για το [2]-sumset covers υπάρχουν στο [3]). Η μελέτη των παραπάνω προβλημάτων σε σχέση με το ESS αποτελεί ενδιαφέρουσα μελλοντική δουλειά.

# Βιβλιογραφία


[1] S. Arora and C. Lund. Hardness of approximations. In D. Hochbaum, editor, *Approximation Algorithms for* NP*-Hard Problems*, pages 399–446. PWS Publishing Company, 1996.

[2] C. Bazgan, M. Santha, and Z. Tuza. Efficient approximation algorithms for the subset–sum equality problem. In *Proc. of the* $25^{th}$ *International Colloquium on Automata, Languages and Programming (ICALP 1998)*, pages 387–396, 1998.

[3] L. Bulteau, G. Fertin, R. Rizzi, and S. Vialette. Some algorithmic results for [2]-sumset covers. *Inf. Process. Lett.*, 115:1–5, 2015.

[4] C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35:713–728, 2005.

[5] M. Cieliebak and S. Eidenbenz. Measurement errors make the partial digest problem np-hard. In *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, pages 379–390, 2004.

[6] M. Cieliebak, S. Eidenbenz, and A. Pagourtzis. Composing equipotent teams. In *Proc. of the* $14^{th}$ *International Symposium on Fundamentals of Computation Theory (FCT 2003)*, pages 98–108, 2003.

[7] M. Cieliebak, S. Eidenbenz, A. Pagourtzis, and K. Schlude. On the complexity of variations of equal sum subsets. *Nord. J. Comput.*, 14:151–172, 2008.

[8] M. Cieliebak, S. Eidenbenz, and P. Penna. Partial digest is hard to solve for erroneous input data. *Theor. Comput. Sci.*, 349:361–381, 2005.

[9] M. Cieliebak, S. Eidenbenz, and G. J. Woeginger. Complexity and approximability of double digest. *J. Bioinformatics and Computational Biology*, 3:207–224, 2005.

[10] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[11] M. Holzhauser and S. O. Krumke. An FPTAS for the knapsack problem with parametric weights. *CoRR*, abs/1703.06048, 2017.

[12] K. Jansen and S. E. J. Kraft. A faster FPTAS for the unbounded knapsack problem. *CoRR*, abs/1504.04650, 2015.

[13] E. G. C. Jr., C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, and M. Yannakakis. Perfect packing theorems and the average-case behavior of optimal and online bin packing. *SIAM Review*, 44:95–108, 2002.

[14] E. G. C. Jr., M. R. Garey, and D. S. Johnson. Bin packing with divisible item sizes. *J. Complexity*, 3(4):406–428, 1987.

[15] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, 2004.

[16] R. Korf. A complete anytime algorithm for number partitioning. *ARTIFICIAL INTELLIGENCE*, 106:181–203, 1998.

[17] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[18] D. Nanongkai. Simple FPTAS for the subset-sums ratio problem. 113(19–21):750–753, 2013.

[19] K. Pruhs and G. J. Woeginger. Approximation schemes for a class of subset selection problems. In *Proc. of LATIN 2004*, volume 2976 of *LNCS*, pages 203–211, 2004.

[20] S. Skiena, W. D. Smith, and P. Lemke. Reconstructing sets from interpoint distances (extended abstract). In *Proceedings of the Sixth Annual Symposium on Computational Geometry, Berkeley, CA, USA, June 6-8, 1990*, pages 332–339, 1990.

[21] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag New York, Inc., 2001.

[22] G. J. Woeginger and Z. L. Yu. On the equal–subset–sum problem. *Information Processing Letters*, 42:299–302, 1992.