Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Προβλήματα Ελαχιστοποίησης Λόγου Αθροίσματος Υποσυνόλων

Subset Sum Ratio Minimization Problems

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΛΩΝΙΣΤΙΩΤΗΣ ΙΩΑΝΝΗΣ

Επιβλέπων : Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2022

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Προβλήματα Ελαχιστοποίησης Λόγου Αθροίσματος Υποσυνόλων

Subset Sum Ratio Minimization Problems

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΛΩΝΙΣΤΙΩΤΗΣ ΙΩΑΝΝΗΣ

Επιβλέπων : Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Απριλίου 2022.

.....	.....	.....
Αριστείδης Παγουρτζής Καθηγητής Ε.Μ.Π.	Δημήτριος Φωτάκης Αν. Καθηγητής Ε.Μ.Π.	Ευάγγελος Μαρκάκης Αν. Καθηγητής Ο.Π.Α.

Αθήνα, Απρίλιος 2022

Αλωνιστιώτης Ιωάννης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Κάπου, κάτι απίστευτο περιμένει να γίνει γνωστό.
Carl Sagan

Περίληψη

Στην παρούσα διπλωματική εργασία, παρουσιάζουμε το πρόβλημα Subset Sum Ratio το οποίο ορίζεται ως εξής: δεδομένου ενός συνόλου θετικών ακεραίων Z πληθυκότητας n, να προσδιοριστεί εάν υπάρχουν δύο ξένα υποσύνολα του Z, των οποίων ο λόγος των αθροισμάτων τους είναι βέλτιστος. Το Subset Sum Ratio αποτελεί πρόβλημα βελτιστοποίησης και είναι στενά συνδεδεμένο με το Equal Subset Sum, οριζόμενο ως εξής: δεδομένου ενός συνόλου θετικών ακεραίων Z πληθυκότητας n, να προσδιοριστεί εάν υπάρχουν δύο ξένα υποσύνολα του Z, τα στοιχεία των οποίων αθροίζουν στην ίδια τιμή. Ακριβέστερα, στο Subset Sum Ratio θεωρούμε ότι δεν υπάρχουν ξένα υποσύνολα που να αθροίζονται στον ίδιο αριθμό, συνεπώς αναζητούμε αυτά τα ξένα υποσύνολα των οποίων ο λόγος είναι πιο κοντά στη μονάδα.

Εμπνεόμενοι από τεχνικές που χρησιμοποιούνται στις πιο σύγχρονες ερευνητικές δημοσιεύσεις για τα προβλήματα Subset Sum και Equal Subset Sum ,καθώς και προσεγγιστικών σχημάτων που έχουν διαμορφωθεί για το Subset Sum Ratio ,τα οποία περιγράφουμε αναλυτικά στη παρούσα διπλωματική, παρουσιάζουμε ένα νέο προσεγγιστικό σχήμα (FPTAS) το οποίο επιτυγχάνει τη καλύτερη πολυπλοκότητα από τα προσεγγιστικά σχήματα που έχουν παρουσιαστεί μέχρι σήμερα. Πιο αναλυτικά, αποδεικνύουμε ότι για την πολυπλοκότητα του FPTAS για το Subset Sum Ratio εκφραζόμενη στη μορφή $O((n + \frac{1}{\varepsilon})^c)$, γίνεται με $c < 5$, αποτέλεσμα το οποίο αποτελεί βελτίωση συγκριτικά με όλα τα FPTAS's που έχουν δημοσιευτεί.

Επιπροσθέτως, μέσω του προσεγγιστικού σχήματος που παρουσιάζεται σε αυτή τη διπλωματική, αποδεικνύεται μια στενότερη σύνδεση μεταξύ της πολυπλοκότητας του Subset Sum Ratio και του Approximate Subset Sum .Ειδικότερα,στο προτεινόμενο FPTAS χρησιμοποιείται ως υπορουτίνα η προσεγγιστική επίλυση πολλών Subset Sum υποπροβλημάτων. Με αυτό τον τρόπο, οποιαδήποτε βελτίωση στην πολυπλοκότητα σε προσεγγιστικό σχήμα για το Subset Sum μπορεί να μεταφερθεί απευθείας και στο προσεγγιστικό σχήμα για το Subset Sum Ratio

Καταλήγουμε, ότι με το FPTAS που παρουσιάζουμε αποτυπώνεται η ανάγκη για περαιτέρω έρευνα στη προσεγγισιμότητα του Subset Sum και ειδικότερα η μελέτη μιας πιο αδύναμης μορφής προσέγγισης η οποία χαλαρώνει την ανισότητα για το άθροισμα στόχο t. Πιο αναλυτικά, για το σύνολο λύση Y για το οποίο πρέπει να ισχύει $\Sigma(Y) \leq t$, η ανισότητα μετατρέπεται σε $\Sigma(Y) \leq (1+\varepsilon) \cdot t$, συνθήκη η οποία δύναται να βοηθήσει στην διαμόρφωση κάποιου βελτιωμένου προσεγγιστικού σχήματος για το Subset Sum και επομένως για το Subset Sum Ratio που είναι το υπό μελέτη πρόβλημα.

Λέξεις κλειδιά

subset sum, equal subset sum, meet-in-the-middle algorithm, approximate subset sum, color-coding, pseudopolynomial time algorithms.

Abstract

In this diploma dissertation, we present the Subset Sum Ratio problem which is defined as follows: Given a set of positive integers Z of cardinality n, find two disjoint subsets of Z whose ratio of sums is optimal. Subset Sum Ratio constitutes the optimization problem closely related to the Equal Subset Sum problem, which is defined in the following way: Given a set Z of positive integers of cardinality n, find two disjoint subsets of Z whose sums of elements are equal. In particular, in the Subset Sum Ratio problem we assume that there are no disjoint subsets with the same sum, thus we search the subsets that their ratio of sums is as close to 1 as possible.

Inspired from algorithmic techniques that are utilized in the most recent publications for Subset Sum and Equal Subset Sum problems and also approximation schemes which have been formulated for the Subset Sum Ratio problem, which we describe and analyze in detail, we present a new approximation scheme (FPTAS) that achieves an improved time complexity in comparison with the fastest scheme that was presented up until now. In particular, we prove that while the complexity of the current state of the art approximation scheme expressed in the form $O((n + 1/\varepsilon)^c)$ has an exponent $c = 5$, the presented FPTAS achieves complexity with constant $c < 5$. Furthermore, through this approximation scheme, a connection between Subset Sum Ratio and Approximate Subset Sum is established. This happens, because many Subset Sum calculations are used in our FPTAS in order to approximate Subset Sum Ratio .Thus, any improvement in the complexity of approximating Subset Sum could be immediately transferred in our FPTAS and improve its complexity.

We conclude that the presented approximation scheme highlights the necessity of further research on approximation algorithms for Subset Sum and specifically a weak notion of approximation which relaxes the constrain $\Sigma(Y) \leq t$, where t is the target sum, and it is sufficient to satisfy $\Sigma(Y) \leq (1 + \varepsilon) \cdot t$. This relaxation seems to open a way for further improvements in approximating Subset Sum and thus open the way for many improvements in problems that Subset Sum and Subset Sum Ratio are applied.

Key words

subset sum, equal subset sum, meet-in-the-middle algorithm, approximate subset sum, color-coding, pseudopolynomial time algorithms.

Ευχαριστίες

Σε αυτές τις πρώτες σελίδες της διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω κάποιους ανθρώπους οι οποίοι αποτέλεσαν βασικό πυλώνα στήριξης και πηγή έμπνευσης για την ολοκλήρωση αυτής της εργασίας και στους οποίους την αφιερώνω.

Αρχικά, οφείλω ένα μεγάλο ευχαριστώ στον επιβλέποντα καθηγητή της παρούσας διπλωματικής εργασίας, κ. Αριστείδη Παγουρτζή, ο οποίος με βοήθησε πολύ να δημιουργήσω μια πιο καθαρή εικόνα για το κόσμο της έρευνας στη θεωρητική πληροφορική κάτι που από πάντα με γοητεύει και για τις καίριες, ειλικρινείς και αντικειμενικές συμβουλές του τόσο για την ολοκλήρωση της εργασίας όσο και για τρόπους και επιλογές που έχω για να κυνηγήσω μελλοντικά το όνειρό μου.

Επιπλέον, θα ήθελα να ευχαριστήσω τους διδακτορικούς φοιτητές Μανώλη Βασιλάκη, Νίκο Μελλισινό, Αντώνη Αντωνόπουλο και Σταύρο Πετσαλάκη, για την ευχάριστη και αποδοτική συνεργασία μας. Από την αρχή, ένιωσα κομμάτι μιας "καλολαδωμένης μηχανής" όπου με την ευφυία τους και την εμπειρία τους ήρθα σε επαφή με διαφορετικές οπτικές για τη ζωή ενός νεαρού και ταλαντούχου ερευνητή.

Πάνω από όλα όμως, δε μπορώ παρά να εκφράσω την βαθύτατη ευγνωμοσύνη και αγάπη μου στην οικογένειά μου, τους γονείς μου Βάγια και Χρήστο, τα αδέρφια μου Κωνσταντίνα και Στέργιο, που είναι πάντα δίπλα μου, με αποδέχονται όπως είμαι και με καθοδηγούν δίνοντας μου, την ειλικρινή άποψή τους, ακόμα και αν αυτό είναι δύσκολο για μένα. Ελπίζω να τους κάνω υπερήφανους.

Επιπλέον, θα ήθελα να ευχαριστήσω την κοπέλα μου, Μαρίνα, η οποία σε όλο αυτό το "ταξίδι" της διπλωματικής, στις δύσκολες στιγμές και στις χαρούμενες, ήταν πάντα εκεί με το κοφτερό μυαλό, το χαμόγελο και την αλήθεια της, να μου δίνει έμπνευση, δύναμη να συνεχίσω και να παλεύω για τα όνειρά μου.

Τέλος, οφείλω να ευχαριστήσω τους κολλητούς μου φίλους με τους οποίους έχω μεγαλώσει και έχω ζήσει πολλές ξεχωριστές στιγμές, καθώς με έχουν διαμορφώσει ως ένα δοτικό και χαρούμενο άνθρωπο και έκαναν τα φοιτητικά μου χρόνια αξέχαστα.

Αλωνιστιώτης Ιωάννης,

Αθήνα, 10η Απριλίου 2022

Contents

List of Figures

Κεφάλαιο 0

Εκτεταμένη Ελληνική Περίληψη

Σε αυτό το κεφάλαιο θα παρουσιάσουμε συνοπτικά την παρούσα διπλωματική εργασία μέσα από μία εκτεταμένη περίληψη στα Ελληνικά. Θα αποφευχθούν οι τεχνικές λεπτομέρειες και οι αποδείξεις, οι οποίες παρουσιάζονται ενδελεχώς στο αγγλικό κείμενο, και θα εστιάσουμε στην ουσία των αποτελεσμάτων μας.

0.1 Εισαγωγή

'Ενα από τα πιο θεμελιώδη προβλήματα της θεωρίας πολυπλοκότητας, από το οποίο έχουν δημιουργηθεί ποικίλες παραλλαγές μία εκ των οποίων και το υπό μελέτη πρόβλημα, είναι το Subset Sum και διατυπώνεται ως εξής: Δοθέντος ενός συνόλου S θετικών ακεραίων και ενός στόχου t, εξετάστε εάν υπάρχει κάποιο υποσύνολο $A \subseteq S$, όπου το άθροισμα των στοιχείων του ισούται με t. Το παραπάνω πρόβλημα αποτελεί ένα από τα αρχικά NP-πλήρη προβλήματα, όπως παρουσιάστηκαν από τον Karp στο [Karp72], και θεωρείται ένας από τους δομικούς λίθους της θεωρίας πολυπλοκότητας, μαζί με τα προβλήματα Knapsack, SAT και άλλα. Παγκοσμίως στα πανεπιστημιακά ιδρύματα, σε μαθήματα αλγορίθμων παρουσιάζεται η έννοια της ψευδοπολυωνυμικής πολυπλοκότητας μέσω του κλασσικού αλγορίθμου του Bellman για το Subset Sum όπως αποδίδεται στο [Bell57]. Παρότι πρόκειται για ένα εκτενώς μελετημένο πρόβλημα, τα τελευταία χρόνια έχουν γίνει σημαντικές βελτιώσεις όσον αφορά την επιλυσιμότητά του σε ψευδοπολυωνυμικό χρόνο. Νέοι ντετερμινιστικοί [Koil19] και τυχαιοκρατικοί [Brin17, Jin19] αλγόριθμοι έχουν προταθεί, αποτελώντας τις πρώτες ουσιώδεις βελτιώσεις σε σχέση με την καθιερωμένη κλασσική προσέγγιση του Bellman [Bell57] καθώς και την βελτίωση του Pisinger [Pisi99]. Επιπλέον, σε μία πρόσφατη δημοσίευση των Bringmann και Nakos [Brin21] παρουσιάζεται ένα νέο προσεγγιστικό σχήμα για το Subset Sum το οποίο οδηγεί στην εύρεση ενός κατώτερου ορίου στη πολυπλοκότητα των προσεγγιστικών σχημάτων για το Subset Sum αλλά και την εύρεση ενός ταχύτερου προσεγγιστικού σχήματος για την ειδική περίπτωση του Subset Sum που ονομάζεται PARTITION.

Το πρόβλημα Subset Sum Ratio ,το οποίο όπως αναφέρθηκε και προηγουμένως, αποτελεί την εκδοχή βελτιστοποίησης του Equal Subset Sum ,είναι ένα λιγότερο μελετημένο, εντούτοις αξιοσημείωτο πρόβλημα που έχει προσελκύσει την προσοχή αρκετών ερευνητών, λόγω των ενδιαφέρουσων εφαρμογών που βρίσκει σε διάφορα πεδία, όπως λόγου χάρη η υπολογιστική βιολογία [Ciel03b, Ciel04], η υπολογιστική κοινωνική επιλογή [Lipt04] και η κρυπτογραφία [Volo17]. Επιπλέον, σχετίζεται με σημαντικές θεωρητικές έννοιες, όπως παραδείγματος χάρη την πολυπλοκότητα των προβλημάτων αναζήτησης στην κλάση TFNP [Papa94].

Σχετική Βιβλιογραφία

Το πρόβλημα Equal Subset Sum και η εκδοχή βελτιστοποίησής του την οποία μελετάμε, που στη βιβλιογραφία καλείται Subset Sum Ratio [Bazg02], σχετίζονται με διάφορα προβλήματα που παρουσιάζονται σε μία πληθώρα επιστημονικών περιοχών. Μερικά παραδείγματα αποτελούν το πρόβλημα Partial Digest, που εμφανίζεται στην περιοχή της υπολογιστικής βιολογίας [Ciel03b, Ciel04], η εκχώρηση ατομικών αγαθών [Lipt04], η κατασκευή τουρνουά [Khan17],

και μία παραλλαγή του Subset Sum, το Multiple Integrated Sets SSP, που βρίσκει εφαρμογές στο πεδίο της κρυπτογραφίας [Volo17]. Επιπλέον, σχετίζονται και με σημαντικές έννοιες της θεωρητικής επιστήμης των υπολογιστών. Για παράδειγμα, μία περιορισμένη εκδοχή του Equal Subset Sum ανήκει σε μία υποκλάση της κλάσης πολυπλοκότητας TFNP, συγκεκριμένα στην PPP [Papa94], μία κλάση που απαρτίζεται από προβλήματα αναζήτησης που πάντοτε έχουν λύση εξαιτίας κάποιου επιχειρήματος περιστερώνα, και κανένας πολυωνυμικός αλγόριθμος δεν είναι γνωστός για αυτήν την περιορισμένη εκδοχή.

Το πρόβλημα Equal Subset Sum έχει αποδειχθεί ότι είναι NP-hard από τους Woeginger και Yu [Woeg92] και αρκετές παραλλαγές του έχουν αποδειχθεί ότι είναι NP-hard από τους Cieliebak *et al.* στο [Ciel03a, Ciel08]. Έχει μελετηθεί κυρίως σε σχέση με το Subset Sum Ratio, όπου ζητάτε η εύρεση δύο ξένων υποσυνόλων με όσο το δυνατόν μικρότερο λόγο αθροισμάτων. Ένας 1.324-προσεγγιστικός αλγόριθμος έχει προταθεί για το Subset Sum Ratio στο [Woeg92] και αρκετά FPTASs εμφανίστηκαν στα [Bazg02, Nano13, Meli18], με το γρηγορότερο έως τώρα να είναι αυτό που παρουσιάστηκε στο [Meli18] πολυπλοκότητας $\mathcal{O}(n^4/\varepsilon)$, φράγμα το οποίο βελτιώνεται με το αποτέλεσμα της παρούσας διπλωματικής.

Αυτά τα προβλήματα είναι άρρηκτα συνδεδεμένα με το Subset Sum. Το τελευταίο, πρόσφατα παρουσίασε εντυπωσιακή πρόοδο, εξαιτίας των Koiliaris και Xu [Koil19] που παρουσίασαν έναν αλγόριθμο πολυπλοκότητας $\tilde{\mathcal{O}}(\sqrt{n}t)$, όπου n είναι το πλήθος των στοιχείων εισόδου και t είναι ο στόχος, και από τον Bringmann [Brin17] που παρουσίασε έναν πιθανοκρατικό αλγόριθμο πολυπλοκότητας $\tilde{\mathcal{O}}(n+t)$. Οι Jin και Wu πρότειναν έναν απλούστερο πιθανοκρατικό αλγόριθμο [Jin19] πετυχαίνοντας τα ίδια όρια με το [Brin17], που όμως δεν δείχνει να επεκτείνεται αποδοτικά ώστε να γίνεται ανακατασκευή των συνόλων της λύσης.[1] Πολύ πρόσφατα, οι Bringmann και Nakos [Brin20] παρουσίασαν έναν αλγόριθμο πολυπλοκότητας $\mathcal{O}(|\mathcal{S}_t(Z)|^{4/3}\mathrm{poly}(\log t))$, όπου $\mathcal{S}_t(Z)$ είναι το σύνολο όλων των πιθανών αθροισμάτων υποσυνόλων του συνόλου εισόδου Z που είναι μικρότερα του t, βασιζόμενοι στο top-k convolution.

Το πρόβλημα Multiple Subset Sum στην πραγματικότητα αποτελεί μία ειδική περίπτωση του προβλήματος Multiple Knapsack, όπου και τα δύο εκ των οποίων έχουν προσελκύσει ιδιαίτερο ενδιαφέρον. Σχετικά με το Multiple Subset Sum, οι Caprara *et al.* παρουσίασαν ένα PTAS για την περίπτωση όπου όλοι οι στόχοι είναι ίδιοι στο [Capr00], και στη συνέχεια στο [Capr03] παρουσιάζουν έναν 3/4 προσεγγιστικό αλγόριθμο. Το Multiple Knapsack έχει εξεταστεί εκτενέστερα τα τελευταία χρόνια, αφού εφαρμογές του συναντώνται σε διάφορα πεδία, όπως τα οικονομικά ή οι μετακινήσεις. Μερικές αξιοσημείωτες έρευνες πάνω σε παραλλαγές του προβλήματος παρέχονται από τους Lahyani *et al.* [Lahy19] και Dell'Amico *et al.* [Dell19]. Ειδικές περιπτώσεις και παραλλαγές του Multiple Subset Sum, όπως λόγου χάρη το πρόβλημα k-Subset Sum, έχουν μελετηθεί στα [Ciel03a, Ciel08], όπου προτάθηκαν απλοί ψευδοπολυωνυμικοί αλγόριθμοι.

0.1.1 Ερευνητική Συνεισφορά

Ερευνητική Συνεισφορά

Τα τελευταία χρόνια έχει αυξηθεί το ενδιαφέρον της ερευνητικής κοινότητας για προσεγγιστικούς αλγορίθμους και έχει υπάρξει εντυπωσιακή πρόοδος σχετικά με το πρόβλημα Subset Sum καθώς νέοι, πιο αποδοτικοί ψευδοπολυωνυμικοί αλγόριθμοι συνεχώς προτείνονται. Αυτή η πρόοδος μπορεί να επηρεάσει έντονα άλλα σχετιζόμενα προβλήματα.
Στην παρούσα εργασία, εξετάζουμε κυρίως το Subset Sum Ratio πρόβλημα, καθώς και πώς βελτιώσεις στην προσεγγιστική επίλυση του Subset Sum δύναται να επιταχύνουν την προσεγγιστική επίλυση του Subset Sum Ratio και αντίστροφα. Με άλλα λόγια, αποδεικνύουμε

[1] Παρατηρήστε ότι ούτε ο αλγόριθμος του Bringmann [Brin17] λύνει την εκδοχή αναζήτησης του Subset Sum, όμως σε αυτή την εργασία θα δείξουμε πώς μπορούμε να επεκτείνουμε κατάλληλα τον αλγόριθμο ώστε να επιστρέφονται τα σύνολα της λύσης του Equal Subset Sum, χωρίς να αυξηθεί η πολυπλοκότητά του (αγνοώντας πολυλογαριθμικούς παράγοντες).

τη σύνδεση της προσεγγισιμότητας του Subset Sum με αυτή του Subset Sum Ratio κατασκευάζοντας ένα προσεγγιστικό σχήμα για το Subset Sum Ratio που χρησιμοποιεί την επίλυση στιγμιοτύπων του Subset Sum προβλήματος. Ταυτοχρόνως, επιτυγχάνεται η βελτίωση της χρονικής πολυπλοκότητας σε σύγκριση με το ταχύτερο εώς τώρα FPTAS για το Subset Sum Ratio .
Μία πληθώρα προβλημάτων μπορούν να επηρεαστούν και να εκμεταλλευτούν αυτήν την πρόοδο, με αποτέλεσμα νέους, πιο αποδοτικούς αλγορίθμους για πολλά αλγοριθμικά προβλήματα, που χρησιμοποιούν ως υπορουτίνα το Subset Sum Ratio πρόβλημα και αναδεικνύοντας έτσι την στενή σχέση που έχουν με το Subset Sum .

0.2 Προαπαιτούμενα

Σε αυτό το υποκεφάλαιο, θα παρουσιαστούν συνοπτικά κάποιες θεωρητικές έννοιες που αποτελέσουν το θεωρητικό υπόβαθρο για την αρτιότερη κατανόηση του ελληνικού κειμένου. Επιπρόσθετα, γίνεται αναλυτικότερη παρουσίαση των εννοιών και των συμβολισμών της παρούσας εργασίας σε μεταγενέστερο κεφάλαιο που περιέχεται στο αγγλικό κείμενο (παραπομπή στο κεφάλαιο 2 του αγγλικού κειμένου).

Συμβολισμός

Αρχικά, παρουσιάζουμε τον συμβολισμό που χρησιμοποιείται σε αυτήν την ελληνική περίληψη. Σε μεγάλο βαθμό ακολουθούμε τον συμβολισμό που χρησιμοποιείται στα [Brin17] και [Koil19].

- Δεδομένου ενός συνόλου $A \subseteq \mathbb{N}$, συμβολίζουμε
  - ▷ το άθροισμα των στοιχείων του με $\Sigma(A) = \sum_{a \in A}$.
  - ▷ το σύνολο όλων των δυνατών αθροισμάτων υποσυνόλων του A με $\mathcal{S}(A) = \{\Sigma(X) \mid X \subseteq A\}$.
  - ▷ το σύνολο όλων των δυνατών αθροισμάτων υποσυνόλων του μέχρι t με $\mathcal{S}_t() = \mathcal{S}() \cap [t]$.
  - ▷ το σύνολο όλων των δυνατών αθροισμάτων υποσυνόλων του A μαζί με την εκάστοτε πληθυκότητά τους με $\mathcal{SC}(A) = \{(\Sigma(X), |X|) \mid X \subseteq A\}$.
  - ▷ το σύνολο όλων των δυνατών αθροισμάτων υποσυνόλων του A μέχρι t μαζί με την εκάστοτε πληθυκότητά τους με $\mathcal{SC}_t(A) = \mathcal{SC}(A) \cap ([t] \times \mathbb{N})$.
- Δεδομένου δύο συνόλων $X, Y \subseteq \mathbb{N}$, συμβολίζουμε το σύνολο των δυνατών αθροισμάτων που προέρχονται από στοιχεία των δύο σύνολων ως $X \oplus Y = \{x + y \mid x \in X \cup \{0\}, y \in Y \cup \{0\}\}$. Επιπλέον, ορίζουμε $X \oplus_t Y = (X \oplus Y) \cap [t]$.
- $\tilde{\mathcal{O}}$ – Ευρέως χρησιμοποιούμενος συμβολισμός στο κομμάτι της υπολογιστικής πολυπλοκότητας, $f(n) \in \tilde{\mathcal{O}}(g(n))$ είναι συντομογραφία για το $\exists k : f(n) \in \mathcal{O}(g(n) \log^k g(n))$, δηλαδή χρησιμοποιείται για την απόκρυψη πολυλογαριθμικών παραγόντων.

Θεωρητικό Υπόβαθρο

Σε αυτό το σημείο θα παρουσιάσουμε κάποιες βασικές θεωρητικές έννοιες που κρίνονται απαραίτητες για την κατανόηση της παρούσας εργασίας.

NP-completeness Η κλάση πολυπλοκότητας NP εμπεριέχει τα προβλήματα απόφασης για τα οποία, τα input instances με καταφατική απόκριση έχουν αποδείξεις που πιστοποιούνται σε πολυωνυμικό χρόνο από μία ντετερμινιστική μηχανή Turing. Με άλλα λόγια, δεδομένου ενός στιγμιότυπου εισόδου του προβλήματος, μπορούμε να πιστοποιήσουμε ότι η έξοδος είναι όντως καταφατική σε πολυωνυμικό χρόνο.

Εάν ένα πρόβλημα Π_1 ανάγεται σε ένα πρόβλημα Π_2 σε πολυωνυμικό χρόνο ($\Pi_1 \leq_P \Pi_2$), τότε:

- Υπάρχει μία συνάρτηση $T : \Sigma^* \to \Sigma^*$ η οποία υπολογίζεται σε πολυωνυμικό χρόνο, τέτοια ώστε $\forall x \in \Sigma^*$, ισχύει ότι $x \in \Pi_1 \iff T(x) \in \Pi_2$, όπου με Σ^* αναπαριστούμε το σύνολο όλων των πιθανών εισόδων και $x \in \Pi$ αν και μόνο αν η είσοδος x έχει ως αποτέλεσμα καταφατική έξοδο για το πρόβλημα Π.
- Η συνάρτηση T καλείται *πολυωνυμική αναγωγή.*
- Ουσιαστικά, μετασχηματίζουμε (αποδοτικά) κάθε στιγμιότυπο εισόδου του προβλήματος Π_1 σε ένα στιγμιότυπο εισόδου του προβλήματος Π_2 και στη συνέχεια επιλύουμε το πρόβλημα Π_2.

Ένα πρόβλημα απόφασης Π είναι *NP-complete* εάν:

1. Το Π ανήκει στην κλάση NP, και
2. Κάθε πρόβλημα στην κλάση NP μπορεί να αναχθεί στο Π σε πολυωνυμικό χρόνο.

Τα NP-complete προβλήματα αναπαριστούν τα δυσκολότερα προβλήματα της κλάσης NP και συνοψίζουν την υπολογιστική δυσκολία της ίδιας της κλάσης.

Ψευδοπολυωνυμικοί Αλγόριθμοι Υποσύνολο των NP-complete αποτελούν τα weakly NP-complete προβλήματα. Ένα πρόβλημα αποκαλείται *weakly NP-complete* εάν υπάρχει αλγόριθμος που το επιλύει, του οποίου ο χρόνος εκτέλεσης είναι πολυωνυμικός ως προς την αριθμητική τιμή της εισόδου (ο μεγαλύτερος ακέραιος που υπάρχει στην είσοδο) αλλά όχι απαραίτητα στο μήκος της εισόδου (τον αριθμό των bits που χρειάζονται για να την αναπαραστήσουν), που συμβαίνει για τους αλγορίθμους πολυωνυμικού χρόνου. Ένας τέτοιος αλγόριθμος καλείται *ψευδοπολυωνυμικός*, αφού η αριθμητική τιμή της εισόδου είναι εκθετική ως προς το μήκος της εισόδου.

Τυχαιοκρατικοί Αλγόριθμοι Ένας αλγόριθμος αποκαλείται *τυχαιοκρατικός* εάν εμπεριέχει έναν βαθμό τυχαιότητας ως μέρος της λογικής του. Σε αντίθεση με έναν *ντετερμινιστικό* (ή αλλιώς *αιτιοκρατικό*) αλγόριθμο, η έξοδος ενός τυχαιοκρατικού αλγορίθμου μπορεί να διαφέρει ανάμεσα σε ξεχωριστές κλήσεις στο ίδιο στιγμιότυπο εισόδου. Επομένως, υπάρχει η πιθανότητα παραγωγής λανθασμένου αποτελέσματος όπου συνήθως συμβολίζεται με δ. Αυτοί οι αλγόριθμοι αποκαλούνται επίσης *πιθανοτικοί ή πιθανοκρατικοί* και η πολυπλοκότητά τους επηρεάζεται από τις τιμές της πιθανότητας λάθους δ.

Προσεγγιστικά σχήματα PTAS Ένα προσεγγιστικό σχήμα PTAS (polynomial time approximation scheme) είναι ένας αλγόριθμος ο οποίος δέχεται σαν είσοδο instance ενός προβλήματος βελτιστοποίησης και μία παράμετρο σφάλματος $\varepsilon > 0$ (error), σε πολυωνυμικό χρόνο, και παράγει ένα αποτέλεσμα το οποίο απέχει κατά ένα παράγοντα $(1 + \varepsilon)$ από την βέλτιστη (ή $(1 - \varepsilon)$ για προβλήματα μεγιστοποίησης). Επιπλέον, ένας αλγόριθμος για να θεωρείται PTAS πρέπει να έχει χρονική πολυπλοκότητα πολυωνυμική ως προς το μέγεθος της εισόδου n, για κάθε σταθερό ε. Δεν υπάρχει απαίτηση για την εξάρτηση από το ε, συνεπώς ένας αλγόριθμος που τρέχει σε χρόνο $O(n^{1/\varepsilon})$ ή ακόμα και $O(n^{exp(1/\varepsilon)})$, μπορεί να αποτελεί ένα PTAS.

Προσεγγιστικά σχήματα FPTAS Ένα προσεγγιστικό σχήμα καλείται FPTAS (fully polynomial time approximation scheme) όταν τρέχει σε χρόνο πολυωνυμικό και ως προς το μέγεθος της εισόδου n, αλλά και ως προς το $1/\varepsilon$. Πιο αναλυτικά, όλα τα προβλήματα που επιδέχονται FPTAS, έχουν επίσης και PTAS. Αποτελούν, λοιπόν, μια ειδικότερη και αυστηρότερη περίπτωση από τα PTAS και ισχύει FPTAS $\subsetneq$ PTAS.

Συνδυασμός Συνόλων Ένα κοινό χαρακτηριστικό των αλγορίθμων που εξετάζονται σε αυτήν την διπλωματική εργασία είναι η διαίρεση του αρχικού συνόλου εισόδου Z σε υποσύνολα $Z_1, \ldots, Z_k$ και ο αναδρομικός υπολογισμός των αθροισμάτων υποσυνόλων τους. Μετέπειτα, αυτά τα αθροίσματα υποσυνόλων συνδυάζονται με σκοπό τον υπολογισμό των αθροισμάτων υποσυνόλων του αρχικού συνόλου Z. Επομένως, είναι πολύ σημαντικό να πραγματοποιείται αποδοτικά αυτός ο συνδυασμός, αφού έχει κεντρικό ρόλο στον καθορισμό της πολυπλοκότητας των αλγορίθμων. Αυτό γίνεται με χρήση του αλγορίθμου FFT, η λογική και η χρησιμότητα του οποίου αναλύεται διεξοδικά στο αγγλικό κείμενο. Προσέξτε πως ο αλγόριθμος FFT μπορεί να επεκταθεί και σε σύνολα k-διάστατων σημείων, όπου μπορεί να πραγματοποιηθεί αποδοτικά το άθροισμα των τιμών των εκάστοτε διαστάσεων.

Κλάσεις Ισοτιμίας Δοθέντος ενός ακεραίου $n > 1$, δύο ακέραιοι a, b είναι *ισότιμοι modulo* n, αν το n είναι διαιρέτης της διαφοράς τους (με άλλα λόγια, υπάρχει ακέραιος $k \in \mathbb{Z}$ τέτοιος ώστε $a-b = kn$). Η ισοτιμία modulo n είναι σχέση ισοδυναμίας και συμβολίζεται με $a \equiv b \pmod{n}$.

Το σύνολο όλων των κλάσεων ισοτιμίας modulo n συμβολίζεται με $\mathbb{Z}_n = \{\overline{0}, \ldots, \overline{n-1}\}$. Καθένα από τα στοιχεία του $\overline{i}$ αναπαριστά το σύνολο των ακεραίων που είναι ισότιμοι modulo n με το i, με άλλα λόγια $\forall x \in \mathbb{Z}, x \in \overline{i} \iff x \equiv i \pmod{n}$.

Εύρεση Μαρτύρων μέσω Peeling Κάθε φορά που αναζητούμε τα δυνατά αθροίσματα που μπορούν να σχηματιστούν από τον συνδυασμό δύο συνόλων, είναι καίριας σημασίας ο αποδοτικός προσδιορισμός των επιμέρους αθροισμάτων που αθροίστηκαν για τον υπολογισμό κάθε αθροίσματος. Αυτό είναι απαραίτητο για την ανακατασκευή των συνόλων λύσης και επομένως της επίλυσης της *εκδοχής αναζήτησης* του Equal Subset Sum. Οι Koiliaris και Xu [Koil19] ανάγουν αυτό το πρόβλημα στο *πρόβλημα ανακατασκευής*, όπως αυτό αναφέρεται στο [Auma11] και επιχειρηματολογούν σχετικά με την πολυπλοκότητά του, καταλήγοντας στο συμπέρασμα ότι υπάρχει μόνο πολυλογαριθμική επιβάρυνση στον υπολογισμό των σχετιζόμενων μαρτύρων[2]. Αυτό είναι ένα πολύ σημαντικό αποτέλεσμα, που υποδεικνύει ότι ο υπολογισμός των μαρτύρων δεν προκαλεί κάποιο 'άλμα' στην χρονική πολυπλοκότητα.

Closest set Για ένα σύνολο $S_i \subseteq A$, ορίζουμε ως *closest set*, ένα σύνολο $S_{i,opt}$, τέτοιο ώστε $S_{i,opt} \subseteq A \setminus S_i$ και $\Sigma(S_i) \geq \Sigma(S_{i,opt}) \geq \Sigma(S')$, για κάθε $S' \subseteq A \setminus S_i$.

ε-close set Για ένα σύνολο $S_i \subseteq A$ ορίζουμε ως *ε-close set* ένα σύνολο $S_{i,\varepsilon}$ τέτοιο ώστε $S_{i,\varepsilon} \subseteq A \setminus S_i$ και $\Sigma(S_i) \geq \Sigma(S_{i,\varepsilon}) \geq (1-\varepsilon)\Sigma(S_{i,opt})$.

0.3 Προσεγγιστικά σχήματα για το Subset Sum

Σε αυτό το υποκεφάλαιο θα παρουσιάσουμε συνοπτικά τη διαίσθηση δύο προσεγγιστικών σχημάτων FPTAS, τα οποία έχουν παρουσιαστεί στις εργασίες των Kelleler [Kell03] και Bringmann, Nakos [Brin21] για το Subset Sum πρόβλημα. Οι ιδέες και οι τεχνικές που παρουσιάζονται σε αυτές τις ερευνητικές δημοσιεύσεις, αποτελούν κομμάτι και του προσεγγιστικού σχήματος

[2] Ως *μάρτυρες* ενός αθροίσματος s, ορίζουμε τα επιμέρους αθροίσματα s' και $s - s'$, από τα οποία σχηματίστηκε το άθροισμα s.

που θα παρουσιάσουμε αργότερα για το Subset Sum Ratio . Εκτενέστερη παρουσίαση των αλγορίθμων θα γίνει στο αγγλικό κείμενο.

Στο υπόλοιπο του κεφαλαίου θα θεωρούμε ως σύνολο εισόδου το σύνολο $A = \{a_1, a_2, ..., a_n\}$ με $|A| = n$ και θεωρούμε ότι τα στοιχεία του συνόλου A είναι ταξινομημένα σε αύξουσα σειρά, δηλαδή $a_1 < a_2 < ... < a_n$. Συνεπώς, όπως θα γίνει εμφανές, στόχος των αλγορίθμων είναι να παράξουν λύση για το Subset Sum με κάποιο περιθώριο σφάλματος ε, σε χρόνο πολυωνυμικό τόσο ως προς το μέγεθος του κωδικοποιημένου στιγμιοτύπου της εισόδου n, όσο και ως προς τη ποσότητα $1/\varepsilon$.

0.3.1 Αποδοτικό FPTAS σε χρόνο $O(min\{\frac{n}{\varepsilon}, n + \frac{1}{\varepsilon^2}log(\frac{1}{\varepsilon})\})$ και χώρο $O(n + \frac{1}{\varepsilon})$

Το FPTAS που θα παρουσιαστεί, δημοσιεύθηκε από τους Hans Kellerer, Renata Mansini, Ulrich Pferschy και Maria Grazia Speranza. Όπως ανεφέρθηκε και στην εισαγωγή, το Subset Sum είναι μια ειδική περίπτωση του Knapsack προβλήματος, όπου το βάρος των αντικειμένων ισούται με το κόστος τους. Συνεπώς, η δομή του αλγορίθμου ξεκινάει από τον γνωστό ψευδωπολυωνυμικό αλγόριθμο του Bellman και θα παρουσιαστούν οι αλλαγές και οι περιορισμοί που χρειάζονται, ώστε ο αλγόριθμος να αποτελεί FPTAS για το υπό μελέτη πρόβλημα. Ο αλγόριθμος του Bellman λύνει βέλτιστα το Subset Sum με τον εξής τρόπο:
Σε ένα σύνολο R που καλείται σύνολο εφικτών τιμών (reachable values) αποθηκεύονται ακέραιοι i μικρότετοι ή ίσοι της χωρητικότητας c, για τους οποίους υπάρχει υποσύνολο του A που το άθροισμα των στοιχείων (βαρών) ισούται με i. Για να κατασκευαστεί το σύνολο R, εκκινούμε από το κενό σύνολο και σε κάθε επανάληψη j προσθέτουμε βάρος w_j σε όλα τα υπάρχοντα στοιχεία του R και ταυτόχρονα αποκλείουμε τα στοιχεία που ξεπερνούν την τιμή χωρητικότητας c. Έτσι, καταλήγουμε με το σύνολο των εφικτών αθροισμάτων που μπορούμε να λάβουμε από υποσύνολα των στοιχείων εισόδου A, και επομένως έχουμε έναν ψευδοπολυωνυμικό αλγόριθμο για το Subset Sum που τρέχει σε χρόνο και χώρο $O(nc)$.
Στο συγκεκριμένο προσεγγιστικό σχήμα με στόχο να καταστεί FPTAS, σε πρώτο στάδιο τα αντικείμενα του συνόλου εισόδου χωρίζονται σε μικρά και μεγάλα στοιχεία. Μικρά θεωρούνται αυτά των οποίων η τιμή είναι $\leq \varepsilon c$ και τα υπόλοιπα τοποθετούνται στο σύνολο των μεγάλων στοιχείων. Είναι εμφανές, λοιπόν, πως οποιαδήποτε $(1-\varepsilon)-$προσεγγιστική λύση που αποτελείται μόνο από μεγάλα στοιχεία, αποτελεί $(1-\varepsilon)-$προσεγγιστική λύση και για το συνολικό πρόβλημα, καθώς η προσθήκη των μικρών στοιχείων δεν μπορεί να επηρρεάσει αρκετά το άθροισμα των μεγάλων. Επομένως, εάν βρεθεί μια $(1-\varepsilon)-$προσεγγιστική λύση με μεγάλα στοιχεία είναι εφικτό με άπληστο τρόπο (greedy way) να προστεθούν και τα μικρά στοιχεία για να πλησιάσει κι άλλο το άθροισμα στόχο t. Επιπρόσθετα, το διάστημα των μεγάλων στοιχείων $[\varepsilon c, c]$ τμηματοποιείται επιπλέον σε $O(1/\varepsilon)$ υποδιαστήματα ίσου μεγέθους εc. Από αυτά τα υποδιαστήματα $I_j = [j\varepsilon c, (j+1)\varepsilon c]$, διατηρούνται το πολύ τα $\lceil \frac{1}{\varepsilon j} \rceil - 1$ μικρότερα και μεγαλύτερα στοιχεία τα οποία τοποθετούνται στο επονομαζόμενο σύνολο σχετικών αντικειμένων (relevant items set) $\mathcal{K}$ και τα υπόλοιπα αποκλείονται. Σε λήμμα που περιέχεται στην δημοσίευση, αποδεικνύται πως κάθε βέλτιστη λύση με στοιχεία μόνο από το σύνολο $\mathcal{K}$ είναι το πολύ εc μικρότερη από μία βέλτιστη λύση που χρησιμοποιεί όλα τα μεγάλα στοιχεία. Συνεπώς, με αυτή τη μεταβολή η χρήση του αλγορίθμου Bellman με χρήση των στοιχείων του συνόλου $\mathcal{K}$, για το οποίο ισχύει $|\mathcal{K}| \leq O(min\{n, 1/\varepsilon log(1/\varepsilon)\})$, μας δίνει αποτέλεσμα σε χρόνο $O(min\{nc, n + 1/\varepsilon \cdot log(1/\varepsilon)c\})$ και χώρο $O(n + (1/\varepsilon)c)$, με ακρίβεια εc. Οπότε, στόχος είναι να ληφθεί προσεγγιστικός αλγόριθμος που η χρονική και χωρική πολυπλοκότητά του δε θα εξαρτάται από την χωρητικότητα c.
Στο επόμενο βήμα κατασκευής του προσεγγιστικού σχήματος, για να αφαιρεθεί η εξάρτηση από την ποσότητα c, σε κάθε υποδιάστημα μεγάλων στοιχείων I_j διατηρούνται μόνο η μεγαλύτερη τιμή $\delta^+(j)$ και η μικρότερη τιμή $\delta^-(j)$ του υποδιαστήματος αυτού. Με αυτό τον τρόπο, αντικαθιστούμε τις c πιθανές εφικτές τιμές με $1/\varepsilon$ εφικτά διαστήματα, συνεπώς δεν επηρρεάζεται η χρονική πολυπλοκότητα από το c. Επακόλουθα, εφαρμόζεται αντίστοιχη τεχνική με τον αλγόριθμο του Bringmann την οποία ονομάζουν relaxed dynamic programming, την οποία θα

παρουσιάσουμε πιο αναλυτικά στο μεταγενέστερο αγγλικό κείμενο. Τελικά, με χρήση αυτής της μεθόδου και με εφαρμογή διαίρει και βασίλευε καθώς και backtracking για την ανακατασκευή των συνόλων εξόδου, εξασφαλίζεται η παραγωγή επιθυμητού ως προς το σφάλμα, αποτελέσματος σε χρόνο $O(min\{n \cdot 1/\varepsilon, n + 1/\varepsilon^2 log(1/\varepsilon)\})$ και χώρο $O(n + 1/\varepsilon)$, που ήταν ο στόχος, και συμβαίνει γιατί πλέον δεν είναι απαραίτητο να αποθηκεύονται τα σύνολα λύσεων κατά τη διάρκεια του αλγορίθμου, αλλά δύναται να ανακατασκευαστούν, αποθηκεύοντας μόνο δύο αριθμούς σε κάθε επανάληψη.

0.3.2 Σύνδεση του Subset Sum με το Min-Plus Convolution πρόβλημα και παραγωγή FPTAS χρονικής πολυπλοκότητας $\tilde{O}(n + \frac{1/\varepsilon^2}{2^{\Omega(\sqrt{(log(1/\varepsilon))})}})$

Σε αυτή τη παράγραφο θα παρουσιαστεί μία πρόσφατη δημοσίευση των Karl Bringmann και Vasileios Nakos [Brin21], στην οποία παρουσιάστηκαν κάποια πρωτοποριακά αποτελέσματα, καθώς παρουσιάστηκε ένα νέο προσεγγιστικό σχήμα για το Subset Sum μέσω της αναγωγής του προβλήματος στο Min-Plus Convolution πρόβλημα. Με αυτήν την αναγωγή, οι ερευνητές είχαν τη δυνατότητα να παρέχουν άμεσα lower bounds για την πολυπλοκότητα του Approximating Subset Sum και ταυτόχρονα κατασκεύασαν ένα προσεγγιστικό αλγόριθμο για το Partition πρόβλημα που είναι ειδική περίπτωση του Subset Sum στην οποία $t = \Sigma(A)/2$. Στόχος είναι, δηλαδή, ο αλγόριθμος να αποφανθεί για την ύπαρξη υποσυνόλου που να έχει άθροισμα στοιχείων ίσο με το μισό του αθροίσματος του συνόλου εισόδου. Ωστόσο, στη περίπτωση προσέγγισης είμαστε ικανοποιημένοι με μία ακρίβεια ε ως προς το βέλτιστο, συνεπώς δεν απαιτείται η ακριβής επίλυση του προβλήματος.
Επιπροσθέτως, τα πιο ουσιαστικά σημεία σε αυτή τη δημοσίευση είναι η κατασκευή ενός ταχύτερου προσεγγιστικού σχήματος για το Subset Sum μετά από 20 χρόνια και πως το FPTAS για το Partition είναι ντετερμινιστικό και τρέχει σε χρόνο υποτετραγωνικό, σε αντίθεση με τα μέχρι τώρα δημοσιευμένα ερευνητικά προσεγγιστικά σχήματα που ήταν τυχαιοκρατικά, αποτελέσματα τα οποία καθιστούν αυτή την ερευνητική δημοσίευση πρωτοποριακή. Επίσης, οι ερευνητές κατόρθωσαν να απαντήσουν αρνητικά ένα ανοιχτό ερώτημα για το εάν μπορεί να υπάρξει προσεγγιστκό σχήμα για το Subset Sum που να έχει πολυπλοκότητα $O(n + 1/\varepsilon)^{2-\delta}$, για κάποιο $\delta > 0$, δηλαδή να τρέχει σε υποτετραγωνικό χρόνο. Στη συνέχεια, θα παρατεθούν συνοπτικά τα θεωρητικά βήματα της απόδειξης που παρουσιάζονται στο paper και η εκτενέστερη ανάλυση τους θα γίνει σε μεταγενέστερο κεφάλαιο. Τα αποτελέσματα και τα θεωρητικά εργαλεία που χρησιμοποιούνται στο προσεγγιστικό σχήμα της δημοσίευσης συνοψίζονται στις παρακάτω προτάσεις:

Ορισμός 0.1 Λέμε ότι ένα γεγονός συμβαίνει με μεγάλη πιθανότητα, έαν η πιθανότητα πραγματοποίησής του είναι τουλάχιστον $1 - min\{1/n, \Delta/t\}^c$, για κάποια σταθερά $c > 0$ όσο μεγάλη επιθυμούμε.

Θεώρημα 0.2 (Αναγωγή από το Subset Sum στο MinConv)
Εάν το MinConv μπορεί να λυθεί σε χρόνο $T(n)$, τότε το Subset Sum επιδέχεται πιθανοτικό προσεγγιστικό σχήμα, το οποίο είναι ορθό με μεγάλη πιθανότητα, σε χρόνο $\tilde{O}(n + T(1/\varepsilon))$.

Πόρισμα 0.3 (Προσεγγιστικό σχήμα για το Subset Sum)
Υπάρχει πιθανοτικό προσεγγιστικό σχήμα για το Subset Sum που απαντάει ορθά με μεγάλη πιθανότητα, και τρέχει σε χρόνο:

$$\tilde{O}(n + \frac{(1/\varepsilon)^2}{2^{\Omega(\sqrt{(log(1/\varepsilon))})}})$$

Θεώρημα 0.4 (Αναγωγή από το MinConv στο Subset Sum)
Εάν υπάρχει προσεγγιστικό σχήμα για το Subset Sum που τρέχει σε αυστηρά υποτετραγωνικό χρόνο $O((n+1/\varepsilon)^{2-\delta})$, για κάθε $\delta > 0$, τότε το MinConv μπορεί να λυθεί σε χρόνο $O(n^{2-\delta'})$, για κάποιο $\delta' > 0$.

Ο αλγόριθμος που αποτελεί το προσεγγιστικό σχήμα, χρησιμοποιεί μία μορφή προσέγγισης που διατυπώνεται ως εξής:

Δοσμένου ενός συνόλου A και στόχου t και $\varepsilon > 0$, επέστρεψε ένα οποιοδήποτε υποσύνολο $A_i \subseteq A$, το οποίο ικανοποιεί τις σχέσεις: $\Sigma(A_i) \leq t$ και $\Sigma(A_i) \geq min\{(1-\varepsilon)t, OPT\}$

Είναι σημαντικό να παρατηρηθεί ότι αυτή η μορφή προσεγγισιμότητας είναι αρκετά αυστηρή, καθώς σε περίπτωση που $OPT \leq (1-\varepsilon)t$, τότε το πρόβλημα πρέπει να λυθεί ακριβώς.
Στη συνέχεια, χρησιμοποιούν τεχνικές από ήδη υπάρχοντα προσεγγιστικά σχήματα, ωστόσο παρουσιάζουν μία νέα τεχνική για τον υπολογισμό του συνόλου εφικτών αθροισμάτων (sumset) του συνόλου εισόδου A, χρησιμοποιώντας ως μαύρο κουτί τον αλγόριθμο του MinConv. Επιπλέον, χρησιμοποιούνται οι τεχνικές RecursiveSplitting και ColorCoding για να χωρίζονται αναδρομικά τα στοιχεία σε μικρά και μεγάλα και για να λυθεί το Subset Sum για τα μεγάλα στοιχεία, τεχνικές που θα παρουσιασθούν αναλυτικά αργότερα. Τελικά, αποδεικνύεται ότι ο αλγόριθμος εντοπίζει ένα σύνολο εφικτών αθροισμάτων A_f για το οποίο ισχύει:

Ισχυρισμός 0.5 Με μεγάλη πιθανότητα ισχύει $max(A_f) \geq min\{(1-\varepsilon)t, OPT\}$

Συνοψίζοντας, δημιουργείται ένα FPTAS για το Subset Sum , το οποίο απαντάει ορθά με μεγάλη πιθανότητα (όπως ορίζεται Ορισμός 0.1) και τρέχει σε χρόνο $O((n+T(1/\varepsilon))log^8(n/\varepsilon))$.

0.4 Προσεγγιστικά σχήματα για το Subset Sum Ratio

Στο παρών υποκεφάλαιο, θα παρουσιαστούν δύο FPTAS's για το Subset Sum Ratio .Το πρώτο αποτελεί το μέχρι προσφάτως, ταχύτερο προσεγγιστικό σχήμα που είχε δημοσιευθεί από τους Aris Pagourtzis και Nikos Mellisinos [Meli18] με χρονική πολυπλοκότητα $O(n^4/\varepsilon)$ και στην συνέχεια θα παρουσιάσουμε το FPTAS που προτείνεται ως νέο ταχύτερο προσεγγιστικό σχήμα και είναι η ερευνητική συνεισφορά της παρούσας διπλωματικής εργασίας.
Υπενθυμίζεται, ότι το Subset Sum Ratio ως πρόβλημα βελτιστοποίησης διατυπώνεται ως εξής:

Subset Sum Ratio(SSR) Δοσμένου ενός ταξινομημένου συνόλου θετικών ακεραίων $A = \{a_1, a_2, ..., a_n\}$, να βρεθούν δύο ξένα μεταξύ τους υποσύνολα $S_1, S_2 \subseteq \{1, 2, ..., n\}$ τα οποία ελαχιστοποιούν την εξής ποσότητα:

$$\frac{max\{\Sigma_{i\in S_1} a_i, \Sigma_{j\in S_2} a_j\}}{min\{\Sigma_{i\in S_1} a_i, \Sigma_{j\in S_2} a_j\}}$$

Επομένως, στόχος είναι να εντοπισθούν δύο υποσύνολα ξένα μεταξύ τους τα οποίων ο λόγος είναι ο ελάχιστος δυνατός (πιο κοντά στη μονάδα). Επιπροσθέτως, αναφέρεται ότι στον αριθμητή μπαίνει το υποσύνολο με το μεγαλύτερο άθροισμα στοιχείων.

0.4.1 FPTAS για το SSR σε χρόνο $O(n^4/\varepsilon)$ με χρήση scaling

Το προσεγγιστικό σχήμα που παρουσιάζεται σε αυτό το paper, είναι το ταχύτερο που υπάρχει στην βιβλιογραφία και γίνεται με χρήση ενός ψευδοπολυωνυμικού αλγορίθμου, όπου εκμεταλλευόμενοι κάποιες κρίσιμες ιδιότητες των συνόλων λύσεων επιτυγχάνουν μεγάλη βελτίωση στη πολυπλοκότητα σε σχέση με τα προηγούμενα FPTAS's. Αρχικά, ορίζονται οι εξής ποσότητες:

$$\mathcal{R}(S_1, S_2, A) = \begin{cases} \frac{\Sigma_{i\in S_1} a_i}{\Sigma_{j\in S_2} a_j} & if S_1 \cup S_2 \neq \emptyset \\ +\infty & otherwise \end{cases}$$

$$\mathcal{MR}(S_1, S_2, A) = max\{\mathcal{R}(S_1, S_2, A), \mathcal{R}(S_2, S_1, A)\}$$

οι οποίες αντιπροσωπεύουν τον λόγο αθροισμάτων δύο υποσυνόλων και τον μέγιστο λόγο αθροισμάτων δύο υποσυνόλων αντίστοιχα. Για την κατασκευή αυτού του FPTAS οι ερευνητές εμπνεύστηκαν από το προσεγγιστικό σχήμα που προτάθηκε από τον Nanonghai στο [Nano13] όπου για την αντιμετώπιση του προβλήματος έλυσαν μία πιο αυστηρή εκδοχή του προβλήματος, που διατυπώνεται ως εξής:

Restricted Subset Sum Ratio problem. Δοσμένου ενός ταξινομημένου συνόλου θετικών ακεραίων $A = \{a_1, a_2, ..., a_n\}$ και δύο ακεραίων $1 \leq p < q \leq n$, να βρεθούν δύο υποσύνολα $S_1, S_2 \subseteq \{1, 2, ..., n\}$ ξένα μεταξύ τους, τέτοια ώστε $\{max(S_1), max(S_2)\} = \{p, q\}$ και ο λόγος των αθροισμάτων τους ελαχιστοποιείται.

Πιο διαισθητικά, απαιτείται η επίλυση ενός δυσκολότερου προβλήματος, καθώς τα μεγαλύτερα στοιχεία και των δύο συνόλων που απαρτίζουν τη λύση είναι γνωστά. Στην ερευνητική δημοσίευση, ωστόσο, που παρουσιάζεται στο παρών υποκεφάλαιο, οι ερευνητές παρατήρησαν ότι δεν χρειάζεται να είναι γνωστά τα μέγιστα στοιχεία και των δύο συνόλων, παρά μόνο του ενός. Συνεπώς, διατύπωσαν και έλυσαν το εξής πρόβλημα:

Semi Restricted Subset Sum Ratio problem. Δοσμένου ενός ταξινομημένου συνόλου θετικών ακεραίων $A = \{a_1, a_2, ..., a_n\}$ και ενός ακεραίου $1 \leq p \leq n$ να βρεθούν δύο υποσύνολα $S_1, S_2 \subseteq \{1, 2, ..., n\}$ ξένα μεταξύ τους, τέτοια ώστε $max(S_1) = p < max(S_2)$ και ο λόγος αθροισμάτων τους ελαχιστοποιείται.

Συνεπώς, είναι εύκολο κανείς να παρατηρήσεις το εξής: Έστω ότι, το ζεύγος συνόλων S_1^*, S_2^* αποτελεί την βέλτιστη λύση του Subset Sum Ratio προβλήματος ενός στιγμιοτύπου εισόδου και S_1^p, S_2^p του Semi Restricted Subset Sum Ratio προβλήματος με είσοδο A, p, τότε ισχύει:

$$\mathcal{MR}(S_1^*, S_2^*, A) = \min_{p \in \{1,2,...,n-1\}} \mathcal{MR}(S_1^p, S_2^p, A)$$

Συνεπώς, καταλήγουμε ότι μπορούμε εντοπίσουμε τη βέλτιστη λύση του Subset Sum Ratio προβλήματος, λύνοντας τη Semi Restricted εκδοχή του για $p \in \{1, 2, ..., n\}$.
Ύστερα, οι ερευνητές κάνουν διερεύνηση για τη τιμή του q, δηλαδή που βρίσκεται το μέγιστο στοιχείο του δεύτερου συνόλου της βέλτιστης λύσης, κι έτσι χωρίζουν το σύνολο λύσεων σε 3 διαφορετικά υποσύνολα με τον εξής τρόπο:

$$A = \{a_1, ..., a_p\} \cup B \cup C$$

όπου $B = \{a_i | i > p, a_i < \Sigma_{j=1}^{p} a_j\}$ και $C = \{a_i | a_i \geq \Sigma_{j=1}^{p} a_j\}$. Επομένως, είναι προφανές ότι το q ανήκει είτε στο σύνολο B είτε στο C αφού εξ ορισμού της αυστηρότερης εκδοχής του προβλήματος ισχύει $p < q$. Τελικά, με αυτό τον τον διαχωρισμό κατασκευάζουν ένα ψευδοπολυωνυμικό αλγόριθμο, τον οποίο θα δούμε αναλυτικά σε επόμενο κεφάλαιο, πολυπλοκότητας $O(n \cdot Q)$, όπου $Q = \Sigma_{i=1}^{p} a_i$ ποσότητα που αποτελεί άνω όριο στο άθροισμα του πρώτου υποσυνόλου της βέλτιστης λύσης.
Συνεχίζοντας, προκειμένου να κατασκευαστεί ένα FPTAS με χρήση του αλγορίθμου που παρουσίασαν, κάνουν χρήση scaling ορίζοντας μία παράμετρο scaling $\delta = \frac{\varepsilon \cdot a_p}{3 \cdot n}$ με την οποία δημιουργείται ένα νέο σύνολο εισόδου $A' = \{a'_1, ..., a'_n\}$ με $a'_i = \lfloor \frac{a_i}{\delta} \rfloor$. Έπειτα, χρησιμοποιείται ο παραπάνω αλγόριθμος με είσοδο $\{A', p\}$ ο οποίος λύνει βέλτιστα το Subset Sum Ratio για το scaled σύνολο εισόδο και για τη σχέση αυτής της λύσης (S_A, S_B) με τη βέλτιστη λύση του αρχικού προβλήματος (S_1^*, S_2^*) ισχύει:

$$\mathcal{MR}(S_A, S_B, A') \leq (1 + \varepsilon) \cdot \mathcal{MR}(S_1^*, S_2^*, A)$$

δηλαδή η παραγόμενη λύση αποτελεί $(1+\varepsilon)$-προσέγγιση της βέλτιστης λύσης του αρχικού προβλήματος. Για την πολυπλοκότητα του προσεγγιστικού σχήματος ισχύει:

$$Q = \Sigma_{i=1}^{p} a'_i \leq n \cdot a'_p \leq \frac{n \cdot a_p}{\delta} = \frac{3 \cdot n^2}{\varepsilon}$$

επομένως ο ψευδοπολυωνυμικός αλγόριθμος που έχει πολυπλοκότητα $O(n \cdot Q)$ τρέχει σε χρόνο $O(n \cdot \frac{3 \cdot n^2}{\varepsilon}) = O(\frac{n^3}{\varepsilon})$. Τελικά, επειδή τρέχουμε τον αλγόριθμο n φορές για την διερεύνηση στη παράμετρο p, η συνολική πολυπλοκότητα του FPTAS είναι $O(\frac{n^4}{\varepsilon})$.

0.4.2 Σύνδεση του SSR με το approximate Subset Sum και FPTAS σε χρόνο $\tilde{O}(min\{\frac{n^{2.3}}{\varepsilon^{2.6}}, \frac{n^2}{\varepsilon^3}\})$

Σε αυτή την παράγραφο θα παρουσιάσουμε συνοπτικά την συνεισφορά αυτής της διπλωματικής εργασίας όπου είναι ένα νέο FPTAS για το Subset Sum Ratio πρόβλημα το οποίο τρέχει σε χρόνο $O(\min\{\frac{n^{2.3}}{\varepsilon^{2.6}} \cdot \log(n/\varepsilon^2), \frac{n^2}{\varepsilon^3} \cdot \log(n/\varepsilon^2), \frac{n^2}{\varepsilon^2}(\log(n/\varepsilon^2) + \frac{1}{\varepsilon^2} \cdot \log(1/\varepsilon)))$. Πιο αναλυτικά, αν εκφράσουμε την πολυπλοκότητα των υπάρχοντων προσεγγιστικών σχημάτων για το Subset Sum Ratio στη μορφή $O((n+1/\varepsilon)^c)$, τότε το σχήμα που θα παρουσιάσουμε είναι το πρώτο που επιτυγχάνει τιμή $c < 5$ κι έτσι αποτελεί βελτίωση στη πολυπλοκότητα του ταχύτερου μέχρι τώρα FPTAS για το Subset Sum Ratio . Επιπροσθέτως, ο εκθέτης στη παράμετρο n μπορεί να μειωθεί μέχρι την τιμή 2, η οποία αποτελεί μεγάλη βελτίωση σε σχέση με την μέχρι τώρα καλύτερη πολυπλοκότητα. Επίσης, δείχνει να είναι και η βέλτιστη τιμή στον εκθέτη του n, εξαιτίας των lower bounds που έχουν τεθεί για το Subset Sum πρόβλημα.
Σε αυτό το προσεγγιστικό σχήμα, αρχικά γίνεται διαχωρισμός μεταξύ μεγάλων και μικρών αντικειμένων και ύστερα διατηρώντας μόνο τα μεγάλα αντικείμενα, εφαρμόζονται αλγόριθμοι που έχουν υλοποιηθεί για το Subset Sum , ώστε να εντοπισθούν καλές προσεγγιστικές λύσεις που αποτελούνται με σύνολα αποκλειστικά μεγάλων αντικειμένων. Στην περίπτωση που δεν εντοπισθούν, τότε αποδεικνύεται ότι είναι εφικτό να εντοπισθεί καλή προσεγγιστική λύση προσθέτοντας μικρά στοιχεία με αποτελεσματικό τρόπο.
Συμπερασματικά, εκμεταλλευόμενοι υπολογισμούς και γρήγορους αλγορίθμους για το Subset Sum έχουμε δύο βασικά ωφέλη:

- Βελτιώνεται σε μεγάλο βαθμό η χρονική πολυπλοκότητα του FPTAS για το Subset Sum Ratio
- Συνδέεται η πολυπλοκότητα του Subset Sum με το Subset Sum Ratio , έτσι ώστε οποιαδήποτε βελτίωση στη προσεγγισιμότητα του πρώτου δύναται να μεταφερθεί άμεσα στο δεύτερο.

Στη συνέχεια δίνεται μια περιγραφή των βημάτων του προσεγγιστικού σχήματος και των θεωρητικών βάσεων του αλγορίθμου:

Μικρά και μεγάλα στοιχεία Το σύνολο εισόδου A χωρίζεται σε δύο υποσύνολα το σύνολο μικρών στοιχείων και στο σύνολο μεγάλων στοιχείων που ορίζονται ως εξής:

- Το σύνολο μικρών στοιχείων $M(A, \varepsilon) = s \in A, s < \varepsilon \cdot \max(A)$.
- Το σύνολο μεγάλων στοιχείων $L(A, \varepsilon) = s \in A, s \geq \varepsilon \cdot \max(A)$.

Closest set Για ένα σύνολο $S_i \subseteq A$, ορίζουμε ως *closest set*, ένα σύνολο $S_{i,opt}$, τέτοιο ώστε $S_{i,opt} \subseteq A \setminus S_i$ και $\Sigma(S_i) \geq \Sigma(S_{i,opt}) \geq \Sigma(S')$, για κάθε $S' \subseteq A \setminus S_i$.

ε-close set Για ένα σύνολο $S_i \subseteq A$ ορίζουμε ως *ε-close set* ένα σύνολο $S_{i,\varepsilon}$ τέτοιο ώστε $S_{i,\varepsilon} \subseteq A \setminus S_i$ και $\Sigma(S_i) \geq \Sigma(S_{i,\varepsilon}) \geq (1-\varepsilon)\Sigma(S_{i,opt})$.

Συνοπτική παρουσίαση του αλγορίθμου

- Αρχικά, αναζητούμε προσεγγιστικές λύσεις που αποτελούνται μόνο από μεγάλα στοιχεία, δηλαδή τα υποσύνολα που τις αποτελούν περιέχουν στοιχεία μόνο από το σύνολο $L(A, \varepsilon)$.
- Ύστερα, παράγουμε όλα τα subset sums που αποτελούνται μόνο από μεγάλα στοιχεία. Εάν το πλήθος τους ξεπερνάει το n/ε^2, τότε μπορούμε εύκολα να βρούμε μια καλή προσεγγιστική λύση.
- Σε αντίθετη περίπτωση, για κάθε ένα από τα παραγόμενα υποσύνολα, εντοπίζουμε το αντίστοιχο ε'-close set, για κάποιο ε' που ορίζουμε αργότερα.
- Τότε αρκεί να μελετήσουμε μόνο αυτά τα ζεύγη υποσυνόλων για πιθανή προσεγγιστική λύση.
- Στην περίπτωση που η βέλτιστη λύση περιέχει μικρά στοιχεία, τότε μπορούμε να εντοπίσουμε μια ε-προσέγγιση της προσθέτοντας μικρά στοιχεία με άπληστο τρόπο.

0.5 Συμπεράσματα και μελλοντικές προεκτάσεις

Αρχικά, το Subset Sum Ratio πρόβλημα έχει αρκετές σημαντικές εφαρμογές στην κρυπτογραφία και στην υπολογιστική βιολογία. Επιπλέον, συνδέεται όπως είδαμε και με το Subset Sum το οποίο έχει εξέχουσα θέση στην θεωρία πολυπλοκότητας και ποικίλες εφαρμογές. Συνεπώς, η βελτίωση στην προσεγγισιμότητα αυτών των δύο προβλημάτων αποτελεί πολύ σημαντική εξέλιξη.
Καταλήγουμε, λοιπόν, σε αυτή την διπλωματική εργασία έχουμε δύο σημαντικές προσθήκες. Βελτίωση της χρονικής πολυπλοκότητας και παραγωγή νέου FPTAS για το Subset Sum Ratio και το σημαντικότερο είναι ότι αποδεικνύεται η σύνδεση του Subset Sum Ratio με την προσεγγιστική εκδοχή του Subset Sum. Συνεπώς, οποιαδήποτε βελτίωση στην ακριβή είτε και στην προσεγγιστική επίλυση του Subset Sum οδηγεί και σε αντίστοιχη βελτίωση στο Subset Sum Ratio πρόβλημα.
Ως μια μελλοντική προέκταση, όπως αναφέρθηκε παραπάνω, θα μπορούσε να αποτελέσει η μελέτη της Weak Approximation of Subset Sum όπως διατυπώνεται στο [Brin21] όπου χρησιμοποιείται για την προσέγγιση του ευκολότερου PARTITION προβλήματος και ίσως με την χρήση αυτής της μορφής προσέγγισης να επιταχυνθεί η εύρεση των ε-close συνόλων.
Μια άλλη κατεύθυνση θα μπορούσε να είναι η χρήση αλγορίθμων που επιτελούν ακριβή επίλυση του Subset Sum και όχι προσεγγιστικά, οι οποίοι είναι παραμετροποιημένοι με τη παράμετρο συγκέντρωσης β, όπως φαίνεται και στα [Aust15], [Aust16], όπου αναλύεται η εκδοχή απόφασης του Subset Sum .Συνεπώς, θα ήταν ενδιαφέρον, να ερευνηθεί πως ανάλογα επιχειρήματα και τεχνικές μπορούν να εφαρμοσθούν στην επίλυση της εκδοχής βελτιστοποίησης.

Chapter 1

Introduction

1.1 Motivation

In complexity theory one of the fundamental problems is the so called SUBSET SUM problem, from which a variety of problems has been produced and is stated in the following way: Given a set A of positive integers and a target t, determine whether there exists a subset $Z \subseteq A$, such that the sum of its elements equals t. For the first time, it was formulated by Karp as one of the fundamental NP-complete problems in [Karp72], alongside with KNAPSACK, SAT and more. In many universities around the world, undergraduate computer science classes introduce the very concept of pseudopolynomial complexity and dynamic programming via the classical algorithm of Bellman, as presented in [Bell57]. SUBSET SUM has been studied for many years, however there have been many vital improvements in the past few years with respect to its pseudopolynomial time solvability. At first, an innovative deterministic [Koil19] and a randomised [Brin17, Jin19] algorithm have been proposed, representing the first substantial improvements over the long-standing standard approach of Bellman [Bell57] and the improvement by Pisinger [Pisi99].

SUBSET SUM RATIO is a less studied, nevertheless noteworthy problem, which is the optimization version of the EQUAL SUBSET SUM problem. SSR has attracted the attention of several researchers, as it finds interesting applications in computational biology [Ciel03b, Ciel04], computational social choice [Lipt04] and cryptography [Volo17]. Moreover, it is related to important theoretical concepts such as the complexity of search problems in the class TFNP [Papa94]. The definition of the SUBSET SUM RATIO problem is stated below:

SUBSET SUM RATIO(SSR) Given a sorted set of positive integers $A = \{a_1, a_2, ..., a_n\}$, detect two disjoint subsets $S_1, S_2 \subseteq \{1, 2, ..., n\}$ such that they minimize the ratio:

$$\frac{max\{\Sigma_{i \in S_1} a_i, \Sigma_{j \in S_2} a_j\}}{min\{\Sigma_{i \in S_1} a_i, \Sigma_{j \in S_2} a_j\}}$$

Consequently, the target is to find two disjoint subsets whose ratio of sums is minimum. It is important to notice, that on the nominator goes larger sum, thus in other words, we try to reach a ratio as close to 1 as possible.
In this dissertation, we construct and formulate an FPTAS ($(1+\varepsilon)$-approximation algorithm). Formally, we are solving the following problem:

Given a *sorted* set of n positive integers $A = a_1, \ldots, a_n$ and a fixed parameter $\varepsilon \in (0, 1)$, find two disjoint subsets $S'_1, S'_2 \subseteq A$ where $\Sigma(S'_2) \leq \Sigma(S'_1)$ which constitute an $(1 + \varepsilon)$-approximation solution. In other words, the ratio of their sums is in ε-distance from the optimal ratio i.e. $\frac{\Sigma(S'_1)}{\Sigma(S'_2)} \leq (1 + \varepsilon)\frac{\Sigma(S^*_1)}{\Sigma(S^*_2)}$.
Let $S^*_1, S^*_2 \subseteq A$ be an optimal solution. Two disjoint subsets $S_1, S_2 \subseteq A$ constitute an $(1+\varepsilon)$-*approximate solution* if and only if $1 \leq \frac{\Sigma(S_1)}{\Sigma(S_2)} \leq (1 + \varepsilon)\frac{\Sigma(S^*_1)}{\Sigma(S^*_2)}$. Notice that since $\frac{\Sigma(S_1)}{\Sigma(S_2)} \geq 1$ it is sufficient, for a pair of disjoint sets S_1, S_2 to be an $(1 + \varepsilon)$-approximation solution, to prove

that $\frac{\Sigma(S_1)}{\Sigma(S_2)} \leq (1+\varepsilon)$, because $1+\varepsilon \leq (1+\varepsilon)\frac{\Sigma(S_1^*)}{\Sigma(S_2^*)}$, so the initial inequality is still satisfied.

1.2 Related Work

Subset Sum Ratio is the optimization version of the so called Equal Subset Sum problem, which both of them are very well studied problems and have many application in a variety of scientific areas such as computational biology, cryptography and computational social choice. Some examples are presented, such as the Partial Digest problem, which comes from computational biology [Ciel03b, Ciel04], the allocation of individual goods [Lipt04], tournament construction [Khan17], and a variation of Subset Sum, namely the Multiple Integrated Sets SSP, which finds applications in the field of cryptography [Volo17]. Moreover, it is related to important concepts in theoretical computer science; for example, a restricted version of Equal Subset Sum namely when the sum of input values is stricly less than $2^n - 1$, then there exists an argument from Dirichlet Principle (Pigheonhole Principle) that ensures that there is a solution, however there is no way to find it. This problem lies in a subclass of the complexity class TFNP, namely in PPP [Papa94], a class consisting of search problems that always have a solution due to some pigeonhole argument, and no polynomial time algorithm is known for this restricted version. The main subject of this thesis is the Subset Sum Ratio problem, and more specifically approximation schemes that have been proposed for this problem. The first FPTAS was proposed by Bazgan et al. in [Bazg02] and more recently a simpler but slower FPTAS was introduced in [Nano13] and a faster one in [Meli18] the latter is the fastest known so far for the problem. Variations of ESS were studied and shown NP-hard in [5,6,7], where also pseudopolynomial time algorithms were presented for some of them and it was left open whether the corresponding optimization problems admit an FPTAS.

1.3 Research Objectives & Contribution

The main contribution of this dissertation, apart from the introduction of a new FPTAS for the Subset Sum Ratio problem, is the establishment of a connection between Subset Sum and approximating Subset Sum Ratio. In particular, we showed that any improvement over the classic meet in the middle algorithm [Horo74] for Subset Sum, or over the approximation scheme for Subset Sum will result in an improved FPTAS for Subset Sum Ratio.

Additionally, it is important to note that there is a distinct limit to the complexity that one may achieve for the Subset Sum Ratio problem using the techniques discussed in this paper.

Finally, we establish that the complexity of approximating Subset Sum Ratio, expressed in the form $O((n + 1/\varepsilon)^c)$ has an exponent $c < 5$, which is an improvement over all the previously presented FPTASs for the problem.

As a direction for future research, we consider the notion of the *weak* approximation of Subset Sum, as discussed in [Much19, Brin21], which was used in order to approximate the slightly easier Partition problem, and may be able to replace the approximate Subset Sum algorithm in the computation of the ε'-close sets.

Another possible direction could be the use of exact Subset Sum algorithms parameterized by a *concentration* parameter β, as described in [Aust15, Aust16], where they solve the decision version of Subset Sum. See also [Dutt21] for a use of this parameter (defined as k in their paper) under a pseudopolynomial setting. It would be interesting to investigate whether analogous arguments could be used to solve the optimization version.

1.4 Thesis Outline

The thesis is structured as follows:

- ▷ We start by presenting some necessary theoretical background in Chapter 2.1. Here, we provide some necessary lemmas used throughout the rest of the thesis. We also explain the notation used as well as some of the used techniques.
- ▷ Afterwards, in the Chapter2 we present and thoroughly analyse two approximation algorithms for Subset Sum problem, as introduced by Bringmann and Nakos in [Brin21] and Kelleler [Kell03]. These are the algorithms we extend in order to efficiently approximate Subset Sum Ratio .
- ▷ Next, in Chapter3 we introduce and analyse two schemes that efficiently approximate the *optimization version* of Equal Subset Sum i.e. Subset Sum Ratio and successfully returning the solution sets. Additionally, we present a faster FPTAS than the was previously the faster one.
- ▷ Finally, in Chapter4 we summarise our results, as well as provide some intuition regarding possible extensions and research areas.

Chapter 2

Preliminaries

In this subsection, we will present the theoretical background, through some definitions, lemmas and theorems that are necessary for a better understanding of the results and algorithms in this dissertation.

Notation

At first, we describe the notation that is used throughout this thesis and the notation from [Brin17] and [Koil19] is largely followed.

- Given a set $A \subseteq \mathbb{N}$, we denote as
  - ▷ the sum of its elements $\Sigma(A) = \sum_{a \in A} a$.
  - ▷ the set of all possible subset sums of A with $\mathcal{S}(A) = \{\Sigma(X) \mid X \subseteq A\}$.
  - ▷ the set of all possible subset sums of A with upper bound t with $\mathcal{S}_t() = \mathcal{S}() \cap [t]$.
  - ▷ the set of all possible subset sums of A with their respective cardinality $\mathcal{SC}() = \{(\Sigma(X), |X|) \mid X \subseteq \}$.
  - ▷ the set of all possible subset sums of A with upper bound t with their respective cardinality $\mathcal{SC}_t(A) = \mathcal{SC}(A) \cap ([t] \times \mathbb{N})$.
- Given two sets $X, Y \subseteq \mathbb{N}$, we denote the pairwise sum from elements of the two sets $X \oplus Y = \{x+y \mid x \in X \cup \{0\}, y \in Y \cup \{0\}\}$. Furthermore, we define $X \oplus_t Y = (X \oplus Y) \cap [t]$.
- $\tilde{\mathcal{O}}$ – Standard complexity notation, used to ignore polylogarithmic factors. Thus, it holds that, $f(n) \in \tilde{\mathcal{O}}(g(n))$ is an abbreviation for $\exists k : f(n) \in \mathcal{O}(g(n) \log^k g(n))$.

Theoretical Background

In this subsection we define some important theoretical concepts.

NP-completeness Complexity class NP includes decision problems, whose input instances have positive answer, they have proofs that can be validated in polynomial time from a deterministic Turing machine. In other words, we can verify if the answer is actually positive in polynomial time. Furthermore, it holds that $P \subseteq NP$, that is if a problem is solvable in polynomial time (class P), then it is also verifiable in polynomial time.

If a problem Π_1 is reducible to a problem Π_2 in polynomial time ($\Pi_1 \leq_P \Pi_2$), then:

- There exists a function $T : \Sigma^* \to \Sigma^*$ which can be computed in polynomial time, such that $\forall x \in \Sigma^*$, it holds that $x \in \Pi_1 \iff T(x) \in \Pi_2$, where with Σ^* we denote the sum of all possible inputs and $x \in \Pi$ if and only if the input x has a positive answer for the problem Π.
- Function T is called *polynomial reduction*.

- Essentially, we efficiently transform every input instance of problem Π_1 into an input instance for problem Π_2 and afterwards we solve problem Π_2.

A decision problem Π is *NP-complete* iff:

1. Π belongs to the NP class, and
2. Every NP problem can be reduced to Π in polynomial time.

NP-complete problems are the most difficult to solve problems in NP class and they represent the total complexity of NP complexity class.

Pseudopolynomial Algorithms Weakly NP-complete problems is a subset of NP-complete problem. An algorithmic problem is called *weakly NP-complete* if there is an algorithm that solves it, whose runtime is polynomial in the numeric value of the input instance (the largest integer in the input) but not necessarily in the length of input (the number of bits that are required for its representation), which is the case for polynomial time algorithms. An algorithm is called *pseudopolynomial*, as the numerical value of the input is exponential in the input length.

Randomized Algorithms A *randomized* algorithm is an algorithm that employs a degree of randomness as part of its logic. In contrast to a *deterministic* algorithm, the output of a randomized one may be different among distinct calls with the same input. Hence, there is a chance of producing an incorrect result (commonly symbolised by δ). These algorithms are also referred to as *probabilistic* and their complexity is affected by the allowed values of error probability δ.

Polynomial Time Approximation Schemes An approximation scheme PTAS (polynomial time approximation scheme) is an algorithm that takes as input in polynomial time an instance of an optimization problem and an error parameter $\varepsilon > 0$ (error), and produces an output that is in an $(1+\varepsilon)$ distance from the optimal ($\acute{\eta}$ $(1-\varepsilon)$ for maximization problems). In addition, for an algorithm to constitute a PTAS, its complexity must be polynomial in the input size n and for every fixed ε. There is no special requirement for the dependence on ε, thus an algorithm which runs in time $O(n^{1/\varepsilon})$ or even in $O(n^{exp(1/\varepsilon)})$, could be a PTAS.

Fully polynomial time approximation schemens An approximation scheme is called FPTAS (fully polynomial time approximation scheme), when it runs in polynomial time both in the input size n, and in $1/\varepsilon$. Specifically, every problem that has an FPTAS, has also a PTAS. Consequently, FPTAS's is a more strict form of PTAS and it holds FPTAS $\subsetneq$ PTAS.

Congruence Classes Given an integer $n > 1$, two integers a, b are said to be *congruent modulo* n, if n is a divisor of their difference (i.e., there exists an integer $k \in \mathbb{Z}$ such that $a-b = kn$). Congruence modulo n is an equivalence relation[1] that is compatible with the operations of addition, subtraction, and multiplication and is denoted by $a \equiv b \pmod{n}$.

The set of all congruence classes of the integers for a modulus n is called the ring of integers modulo n and is denoted by $\mathbb{Z}_n = \{\overline{0}, \ldots, \overline{n-1}\}$. Each of its elements $\bar{i}$ represents the set of integers that are congruent modulo n with i, i.e. $\forall x \in \mathbb{Z}, x \in \bar{i} \iff x \equiv i \pmod{n}$.

Closest set For a given set $S_i \subseteq A$ we define as its *closest set* a set $S_{i,opt}$ such that $S_{i,opt} \subseteq A \setminus S_i$ and $\Sigma(S_i) \geq \Sigma(S_{i,opt}) \geq \Sigma(S')$ for all $S' \subseteq A \setminus S_i$.

[1] An equivalence relation has the following properties: a) Reflexivity, b) Symmetry and c) Transitivity. In our case, that translates to a) $a \equiv a \pmod{n}$, b) $a \equiv b \pmod{n}$ if $b \equiv a \pmod{n}, \forall a, b, n$ and c) If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$.

ε-close set For a given set $S_i \subseteq A$ we define as its *ε-close set* a set $S_{i,\varepsilon}$ such that $S_{i,\varepsilon} \subseteq A \setminus S_i$ and $\Sigma(S_i) \geq \Sigma(S_{i,\varepsilon}) \geq (1-\varepsilon)\Sigma(S_{i,opt})$.

Closest Set and ε-close set computation By solving the following problem, one can compute an ε-close set of a given subset $S_i \subseteq A$.

Definition 1. *(Approximate* Subset Sum*) Given a set $A \setminus S_i$, target $\Sigma(S_i)$ and error margin ε, return a subset $S_{i,\varepsilon} \subseteq A \setminus S_i$ such that $\Sigma(S_i) \geq \Sigma(S_{i,\varepsilon}) \geq (1-\varepsilon)\Sigma(S')$ for all $S' \subseteq A \setminus S_i$ with $\Sigma(S') \leq \Sigma(S_i)$.*

We present two different ways to compute an ε-close set for some set $S_i \subseteq A$. Note that the first way actually returns the closest set, which is by definition an ε-close set.

1. **Closest set ($S_{i,opt}$) computation**
   Compute the subset sums of set $A \setminus S_i$ with target $\Sigma(S_i)$ and keep the largest non exceeding. This can be achieved by a standard meet in the middle [Horo74] algorithm.

2. **ε-close set ($S_{i,\varepsilon}$) computation**
   Run an approximate Subset Sum algorithm [Kell03, Brin21] with error margin ε on set $A \setminus S_i$ with target $\Sigma(S_i)$.

Notation Given a set of n positive integers $S = s_1, \ldots, s_n \subseteq \mathbb{N}$, let

- $\max(S)$ denote its largest element.
- $\Sigma(S) = s_1 + \cdots + s_n$ denote the sum of its elements.

Distinction between large and small elements Given a finite set of positive integers $S \subseteq \mathbb{N}$ as well as a value $\varepsilon \in (0,1)$, define the following *partition* of its elements:

- The set of its *large* elements as $L(S,\varepsilon) = \{s \in S, s \geq \varepsilon \cdot \max(S)\}$.
- The set of its *small* elements as $M(S,\varepsilon) = \{s \in S, s < \varepsilon \cdot \max(S)\}$.

Note that $\max(S) \in L(S,\varepsilon)$, for any $\varepsilon \in (0,1)$.

2.0.1 Relationship between Subset Sum and Equal Subset Sum

Mucha *et al.* present a reduction Subset Sum $\leq$ Equal Subset Sum in the full version of their paper [Much19]. This reduction is slightly sharper than the one presented in [Woeg92] and shows the intrinsic relationship between the two algorithmic problems.

Theorem. *If* Equal Subset Sum *can be solved in time $\mathcal{O}^*((2-\varepsilon)^{0.25n})$ for some $\varepsilon > 0$, then* Subset Sum *can be solved in time $\mathcal{O}^*((2-\varepsilon')^{0.5n})$ for some constant $\varepsilon' > 0$.*

Proof. Assume that we have a black-box access to the Equal Subset Sum algorithm running in time $\mathcal{O}^*((2-\varepsilon)^{0.25n})$ for some $\varepsilon > 0$. We will show how to use this algorithm to obtain an algorithm for Subset Sum running in time $\mathcal{O}^*((2-\varepsilon)^{0.5n})$.

Given an instance (S,t) of Subset Sum such that $S = \{s_1, \ldots, s_n\}$, we will construct an equivalent instance Z of Equal Subset Sum such that $Z = \{z_1, \ldots, z_{2n+1}\}$. The construction is as follows:

- for $1 \leq i \leq n$, let $z_i = s_i \cdot 10^{n+1} + 2 \cdot 10^i$,
- for $1 \leq i \leq n$, let $z_{i+n} = 1 \cdot 10^i$,
- let $z_{2n+1} = t \cdot 10^{n+1} + \sum_{i=1}^{n} 1 \cdot 10^i$.

Now, we will show that if (S, t) is a YES instance for SUBSET SUM, then Z is a YES instance for EQUAL SUBSET SUM. Let $X \subseteq [n]$, such that $\sum_{i \in X} s_i = t$, be the set of the indexes of the elements of the solution set. Then, sets $A = \{z_i \mid i \in X\} \cup \{z_{i+n} \mid i \notin X\}$ and $B = \{z_{i+n} \mid i \in X\} \cup \{z_{2n+1}\}$ are a valid solution to EQUAL SUBSET SUM on instance Z, since $\Sigma(A) = \Sigma(B)$ and $A \cap B = \emptyset$.

For the other direction, we will prove that if Z is a YES instance of EQUAL SUBSET SUM, then (S, t) is a YES instance of SUBSET SUM. Assume that Z is a YES instance and sets $A, B \subseteq Z$ is a correct solution pair. Observe that if for some $i \leq n$ element $z_i \in A$, then $z_{2n+1} \in B$. That is due to the fact that the sets A, B have an equal sum and only the elements z_i, z_{i+n} and z_{2n+1} have something nonzero at the i-th decimal place. Moreover, all smaller decimal places of all numbers sum up to something smaller than 10^i and therefore cannot interfere with the i-th decimal.

Finally, observe that numbers z_{i+n}, for $i \in [n]$, cannot produce a YES instance on their own. Hence, sets $A \cup B$ contain at least one number z_i for $i \in [n]$. Without loss of generality, let A be the set that contains such an z_i. Then, set B has to contain the element z_{2n+1}. That means that set B cannot contain any z_i for $i \in [n]$.

In particular, $\Sigma(A)/10^{n+1} = \Sigma(B)/10^{n+1}$. Only numbers z_i for $i \in [n]$ contribute to $\Sigma(A)/10^{n+1}$ and only number z_{2n+1} contributes to $\Sigma(B)/10^{n+1}$. Hence, there exists a subset $S' \subseteq S$, such that $\Sigma(S') = t$.

□

2.0.2 Subset Sum Ratio

We formally present three different definitions of the Subset Sums Ratio problem, that are used in this diploma dissertation.

SUBSET SUM RATIO problem (SSR) (equivalent definition). Given a set $A = \{a_1, ..., a_n\}$ of n positive integers, find two disjoint sets $S_1, S_2 \subseteq \{1, ..., n\}$ such that the value $\mathcal{MR}(S_1, S_2, A)$ is minimized. In addition, from now on, whenever we have a set $A = \{a_1, ..., a_n\}$ we will assume that $0 < a_1 < a_2 < ... < a_n$ (clearly, if the input contains two equal numbers then the problem has a trivial solution). The FPTAS proposed by Nanonghai approximates the SSR problem by solving a restricted version.

Restricted Subset-Sums Ratio problem . Given a set $A = \{a_1, ..., a_n\}$ of n positive integers and two integers $1 \leq p < q \leq n$, find two disjoint sets $S_1, S_2 \subseteq \{1, ..., n\}$ such that $\{maxS_1, maxS_2\} = \{p, q\}$ and the value $\mathcal{MR}(S_1, S_2, A)$ is minimized. Inspired by this idea, they define a less restricted version. The new problem requires one additional input integer, instead of two, which represents the smallest of the two maximum elements of the sought optimal solution.

Semi-Restricted Subset-Sums Ratio problem . Given a set $A = \{a_1, ..., a_n\}$ of n positive integers and an integer $1 \leq p < n$, find two disjoint sets $S_1, S_2 \subseteq \{1, ..., n\}$ such that $maxS_1 = p < maxS_2$ and the value $\mathcal{MR}(S_1, S_2, A)$ is minimized. Let $A = \{a_1, ..., a_n\}$ be a set of n positive integers and $p \in \{1, ..., n-1\}$. Observe that, if S_1^*, S_2^* is the optimal solution of SSR problem of instance A and S_1^p, S_2^p the optimal solution of Semi-Restricted SSR problem of instance A, p then: $\mathcal{MR}(S_1, S_2^*, A) = min_{p \in \{1,...,n-1\}} \mathcal{MR}(S_1^p, S_2^p, A)$ .Thus, we can find the optimal solution of SSR problem by solving the SSR Semi-Restricted SSR problem for all $p \in \{1, ..., n-1\}$.

Chapter 3

Subset Sum Approximation Algorithms

In this chapter we are going to present two approximation schemes for Subset Sum one from Hans Kellerer, Renata Mansini, Ulrich Pferschy and Maria Grazia Speranza [Kell03] and one new approximation algorithm from Karl Bringmann and Vasileios Nakos [Brin21].
The first fully polynomial approximation scheme for the Subset-Sum Problem was suggested by Ibarra and Kim [Ibar75]. They partition the items into small and large items. The weights of the large items are scaled and then the problem with scaled weights and capacity is solved optimally through dynamic programming. The small items are added afterwards using a greedy-type algorithm. Their approach has time complexity $O(n \cdot 1/\varepsilon^2)$ and space complexity $O(n+1/\varepsilon)$: Lawler [Lawl79] improved the scheme of Ibarra and Kim by a direct transfer of a scheme for the knapsack problem which uses a more efficient method of scaling. His algorithm has only $O(n+1/\varepsilon^4)$ time and $O(n+1/\varepsilon^3)$ memory requirement. Note that the special algorithm proposed in his paper for subset-sum does not work, since he makes the erroneous proposal to round up the item values.
As an improvement, Lawler claims in his paper that a combination of his approach (which is not correct) with a result by Karp [Karp72] would give a running time of $O(n+1/\varepsilon^2 log(\frac{1}{\varepsilon}))$: Karp in this paper presented an algorithm for subset sum with running time $n(\frac{1+\varepsilon}{\varepsilon})log_{1+\varepsilon}2$ which is $O(n \cdot 1/\varepsilon^2)$. Lawler states that replacing n by the number of large items, would give a running time of $O(n+1/\varepsilon^2 log(1/\varepsilon))$. It can be easily checked that a factor of $1/\varepsilon$ is missing in the second term of the expression. Possibly, this mistake originates from the fact that there is a misprint in Karp's paper, giving a false running time.
The approach by Gens and Levner is based on a different idea. They use a dynamic programming procedure where at each iteration solution values are eliminated which are different from each other by at least a threshold value depending on ε. The corresponding solution set is then determined by standard backtracking. Their algorithm solves the Subset-Sum Problem in $O(n \cdot 1/\varepsilon)$ time and space. In 1994 Gens and Levner [Gens94] presented an improved fully polynomial approximation scheme based on the same idea. The algorithm finds an approximate solution with relative error less than ε in time $O(min\{n/\varepsilon, n+1/\varepsilon^3\})$ and space $O(min\{n/\varepsilon, n+1/\varepsilon^2\})$.
The first FPTAS that we will present it was the first improvement over all the previously presented FPTAS's, and it was based on the ideas, which were implemented in the papers we talked about in the previous paragraph, however they exploited some properties of the numbers, thus improving time and spatial complexity.

3.0.1 Efficient FPTAS $O(min\{\frac{n}{\varepsilon}, n+\frac{log(1/\varepsilon)}{\varepsilon^2}\})$ and space $O(n+\frac{1}{\varepsilon})$

Firstly, we give a more intuitive explanation of the algorithm in order to explain better the ideas and theoretical concepts behind it. As a starting point we have the famous dynamic programming algorithm from Bellman, which solves the Subset Sum problem optimally in the following way: A set R of *reachable values* consists of integers i less than or equal to the capacity c for which a subset of items exists with total weight equal to i. Starting from the empty set, R is constructed iteratively in n iterations by adding in iteration j weight w_j

to all elements from R and keeping only partial sums not exceeding the capacity. For each value $i \in R$ a corresponding solution set with total weight equal to i is stored. This gives a pseudopolynomial algorithm with time $O(nc)$ and space $O(nc)$.
In order to formulate an FPTAS at first the input items are split into large and small. Small are the items with weight $\leq \varepsilon c$ and large are the items with weight $> \varepsilon c$. It can be easily seen, that any $(1-\varepsilon)-$approximation algorithm using only large items, constitutes an $(1-\varepsilon)-$approximation solution for the whole item set, by adding small items greedily afterwards.
Consecutively, the small items are discarded and the interval containing large items $[\varepsilon c, c]$ is partitioned in $O(1/\varepsilon)$ subintervals. Then, from each subinterval $I_j = [j\varepsilon c, (j+1)\varepsilon c]$, at most $\lceil \frac{1}{\varepsilon j} \rceil - 1$ smallest and $\lceil \frac{1}{\varepsilon j} \rceil - 1$ largest items are being selected and then stored in the so called *relevant items* set $\mathcal{K}$. The rest items are discarded. Consequently, the corresponding modification of the Bellman algorithm requires time $O(min\{nc, n+1/\varepsilon log(1/\varepsilon)c\})$ and space $O(n+(1/\varepsilon)c)$, but approximates the optimal solution still with accuracy εc: (For each partial sum we have to store $O(1/\varepsilon)$ large items).
The next step, is to formulate an algorithm so that there is no dependence on c. For this purpose, instead of storing the $\lceil \frac{1}{\varepsilon j} \rceil - 1$ smallest and $\lceil \frac{1}{\varepsilon j} \rceil - 1$ largest items, we store only the smallest item $\delta^-(j)$ and the largest item $\delta^+(j)$, for each subinterval I_j. In principle, the c possible *reachable values* have been replace with $1/\varepsilon$ *reachable intervals*. Afterwards, a ***relaxed dynamoic programming*** technique is being performed, which returns an array $\Delta = \{\delta[1], \delta[2], ..., \delta[k']\}$. It is shown, later in the paper, that $\delta[k']$ is at least $(1-\varepsilon)c$ or even equal to the optimal solution value. Thus, we have an $(1-\varepsilon)$-approximation algorithm which runs in time $O(min\{n \cdot 1/\varepsilon, n + 1/\varepsilon^2 log(1/\varepsilon)\})$ and space $O(n + 1/\varepsilon^2)$.
Already, the described algorithm achieves an FPTAS with the claimed running time. However, the space complexity is bigger. In order to improve the space complexity, the researchers use the techniques of **backtracking** and **divide and conquer**. In detail, the space complexity is larger by a factor of $1/\varepsilon$. This happens due to the fact that we store for each reduced reachable value i the corresponding solution set. Thus, if we would be satisfied with calculating only the maximal solution value and not be interested in the corresponding solution set, we could finish the algorithm after this step.
In order to reconstruct the solution set, we start from the maximal solution value and we apply backtracking. The first problem to be resolved in the backtracking procedure is that the partial sum I_i, which remains after each step of backtracking, may not be stored in Δ anymore , thus we select one from the updated values $\{\delta^-(i), \delta^+(i)\}$. Although, we may not have the original values, this is not a big problem because it is shown that if y^B is the total weight of the current solution set determined by backtracking, there exists $y^R \in \{\delta^-(i), \delta^+(i)\}$ such that $(1-\varepsilon)c \leq y^R + y^B \leq c$, thus constituting a good approximation or even exact computation of the optimal. However, the actual problem with backtracking is that the series of indices from which we reconstruct the solution set may have been updated (increased actually) in some step. In this case, we may accept twice the same item in the solution set. A straightforward solution is to run the procedure only when the item values are decreasing. In the worst case, backtracking will always stop after detecting only one correct item of the solution set.
For this reason, the researchers use divide and conquer combined with backtracking in order to reconstruct the approximate solution set. More analytically, after performing backtracking and the values $\delta(j)$ are increased, then the procedure divide and conquer is called. The set $\mathcal{K}$ is partitioned in $\mathcal{K}_1$ and $\mathcal{K}_2$, on which *relaxed dynamic programming* procedure is performed recursively and two o arrays of reduced reachable arrays Δ_1, Δ_2 are returned. By lemma proved in this paper, it is known that there exist $u_1 \in \Delta_1$ and $u_2 \in \Delta_2$ such that $c^* - \varepsilon c \leq u_1 + u_2 \leq c^*$, where $c^* = c - y^B$.
To find the solution sets corresponding to value u_1 and u_2 they first perform backtracking

for item set $\mathcal{K}_1$ with capacity $c^* - u_2$ which reconstructs a part of the solution contributed by $\mathcal{K}_1$ with value y_1^B. If y_1^B is not close enough to $c^* - u_2$ and hence does not fully represent the solution value generated by items in K_1 we perform a recursive execution of divide and conquer for item set K_1 with capacity $c^* - u_2 - y_1^B$ which finally produces y_1^{DC} such that $y_1^B + y_1^{DC}$ is close to u_1.
The same strategy is carried out for K_2 producing a partial solution value y_2^B by backtracking and possibly performing recursively divide and conquer which again returns a value y_2^{DC}. Finally, we derive the solution contributed by item set K from this formula $y^{DC} = y_1^B + y_1^{DC} + y_2^B + y_2^{DC}$.
In every recursive execution of divide and conquer, two runs of relaxed dynamic programming and backtracking are performed for both subsets. If backtracking completely produces the solution for the desired capacity we have completely solved one subproblem, otherwise we continue the splitting process of the item set recursively for the remaining subproblem. As each execution of divide and conquer returns at least one item of the solution through backtracking (usually more than one), the depth of the recursion is bounded by $O(log(1/\varepsilon))$. Furthermore, it is possible to represent the recursive structure of the algorithm (specifically divide and conquer calls) as a binary rooted tree. The root indicates the first call of divide and conquer procedure and every child node represents one call. Each node has up to two child nodes, the left representing the recursive call for K_1 and the right one for $\mathcal{K}_2$. Afterwards, a depth-first search (DFS) is used in order to traverse the tree nodes. Every node returns a part of the solution set computed directly through backtracking and returns as another part the results of its child nodes. Finally, it is guaranteed that the final solution value $y^L = y^B + y^{DC}$ returned by the first backtracking phase and the first application of divide and conquer, either satisfies $(1-\varepsilon)c \leq y^L \leq c$ or that y^L is optimal for the large items. In this way, the algorithm is still a FPTAS, however it is not necessary to store solution sets of items thus requiring only $O(n + 1/\varepsilon)$ space. Finally, Theorem 8 in the paper shows the rather surprising fact that introducing divide and conquer does not increase the running time. This can be intuitively explained by the fact that the size of the subproblems is decreasing systematically both with respect to the number of items and the required subset capacity.In the following, we give a technical description of the algorithm, by presenting its pseudocode.

Algorithm (A)

Input: n, w_j $(j = 1, \ldots, n)$, c, ε.

Output: z^A, X^A.

Step 1: *Partition into intervals.*

Compute the number of intervals $k := \lceil \frac{1}{\varepsilon} \rceil$. Set the interval length $t := \varepsilon c$.

Partition the interval $[0, c]$ into the interval $[0, t]$, into $k-2$ intervals $I_j :=]jt, (j+1)t]$ $(j = 1, \ldots, k-2)$ of length t and the (possibly smaller) interval $I_{k-1} :=](k-1)t, c]$.

Denote the items in $[0, t]$ by S and call them *small items.*

Denote the items in I_j by L_j with $n_j := |L_j|$.

Set $L := \bigcup_{j=1}^{k-1} L_j$ and call the elements of L *large items.*

If $L = \emptyset$ **then go to** Step 4.

Step 2: *Determination of the relevant item set* Λ.

For every $j = 1, \ldots, k-1$ **do**

If $n_j > 2(\lceil \frac{k}{j} \rceil - 1)$ **then**

Let K_j consist of the $\lceil \frac{k}{j} \rceil - 1$ smallest and the $\lceil \frac{k}{j} \rceil - 1$ biggest items in L_j.

Else let K_j consist of all items in L_j.

Define the set of *relevant items* Λ by $\Lambda := \bigcup_{j=1}^{k-1} K_j$. Set $\lambda := |\Lambda|$.

Discard the remaining items $L \backslash \Lambda$.

Step 3: *Dynamic programming recursion.*

$P^L := \emptyset$ (current solution set of large items)

$\Lambda^E := \emptyset$ (set of relevant items excluded from further consideration)

These two sets are updated only by procedure **backtracking**.

Perform procedure **relaxed dynamic programming** (Λ, c) returning the dynamic programming arrays $\delta^-(), \delta^+()$ and $d()$ with entries $\delta^-(j), \delta^+(j)$ $(j = 1, \ldots, k-1)$ and $d(i)$ $(i = 1, \ldots, k', k' \leqslant 2k-1)$. Let the array $\Delta := \{\delta[1] \leqslant \delta[2] \leqslant \cdots \leqslant \delta[k']\}$ of *reduced reachable values* represent the values $\delta^-(j)$, $\delta^+(j)$ (unequal to zero) sorted in non-increasing order.

If $\delta[k'] < (1-\varepsilon)c$, **then** set $c := \delta[k'] + \varepsilon c$.

Perform procedure **backtracking** $(\delta^-(), \delta^+(), d(), \Lambda, c)$ returning y^B.

If $c - y^B > \varepsilon c$ **then** perform procedure **divide and conquer** $(\Lambda \backslash \Lambda^E, c - y^B)$ returning y^{DC}

$y^L := y^B + y^{DC}$.

Step 4: *Assignment of the small items.*

Apply a greedy-type algorithm to S and a knapsack with capacity $c - y^L$, i.e. examine the small items in any order and insert each new item into the knapsack if it fits.

Let y^S be the greedy solution value and P^S be the corresponding solution set.

Finish with $z^A := y^L + y^S$ and $X^A := P^L \cup P^S$.

Comment. It will be clear from Corollary 3 that the possible redefinition of c in Step 3 of the algorithm is used to find the *exact* solution in case of $z^* < (1-\varepsilon)c$.

Step 4: *Assignment of the small items.*

Apply a greedy-type algorithm to S and a knapsack with capacity $c - y^{L}$, i.e. examine the small items in any order and insert each new item into the knapsack if it fits.

Let y^{S} be the greedy solution value and P^{S} be the corresponding solution set.

Finish with $z^{A} := y^{L} + y^{S}$ and $X^{A} := P^{L} \cup P^{S}$.

Comment. It will be clear from Corollary 3 that the possible redefinition of c in Step 3 of the algorithm is used to find the *exact* solution in case of $z^{*} < (1 - \varepsilon)c$.

Procedure relaxed dynamic programming $(\tilde{\Lambda}, \tilde{c})$

Input: $\tilde{\Lambda}$: subset of items, $\tilde{c}$: subset capacity.

Output: $\delta^{-}(), \delta^{+}(), d()$: dynamic programming arrays.

(Forward recursion)

Let $\tilde{\Lambda} = v_1, v_2, \ldots, v_{\tilde{\lambda}}$ and $\tilde{\lambda} := |\tilde{\Lambda}|$.

Compute $\tilde{k}$ with $\tilde{c} \in I_{\tilde{k}}$.

$\delta^{-}(j) := \delta^{+}(j) := 0 \ (j = 1, \ldots, \tilde{k})$.

For $i := 1$ **to** $\tilde{\lambda}$ **do**
begin
Form the set $\Delta_i := \{\delta^{+}(j) + v_i \mid \delta^{+}(j) + v_i \leqslant \tilde{c},\ j = 1, \ldots, \tilde{k}\} \cup$
$\{\delta^{-}(j) + v_i \mid \delta^{-}(j) + v_i \leqslant \tilde{c},\ j = 1, \ldots, \tilde{k}\} \cup \{v_i\}$.
For all $u \in \Delta_i$ **do**
begin
Compute j with $u \in I_j$.
If $\delta^{-}(j) = 0$ (and therefore also $\delta^{+}(j) = 0$) **then** $\delta^{-}(j) := \delta^{+}(j) := u$
and $d(\delta^{-}(j)) := d(\delta^{+}(j)) := i$.
If $u < \delta^{-}(j)$ **then** $\delta^{-}(j) := u$ and $d(\delta^{-}(j)) := i$.
If $u > \delta^{+}(j)$ **then** $\delta^{+}(j) := u$ and $d(\delta^{+}(j)) := i$.
end
end
return $\delta^{-}(), \delta^{+}(), d()$.

Comment. In each interval I_j we keep only the current biggest iteration value $\delta^{+}(j)$ and the current smallest iteration value $\delta^{-}(j)$, respectively. The value $d(\delta)$ represents the index of the last item which is used to compute the iteration value δ. It is stored for further use in procedure backtracking. Note that the last interval $I_{\tilde{k}}$ contains only values smaller than or equal to $\tilde{c}$.

Procedure backtracking $(\delta^-(), \delta^+(), d(), \tilde{\Lambda}, y^T)$
Input: $\delta^-(), \delta^+(), d()$: dynamic programming arrays,
$\tilde{\Lambda} = \{v_1, v_2, \ldots, v_{\tilde{\lambda}}\}$: subset of items as in **relaxed dynamic programming**;
y^T: target point for **backtracking**.
Output: y^B: collected partial solution value.
This is the only procedure where items are added to P^L and deleted from Λ^E.

(Backward recursion)
$u := \max_j\{u'_j \mid u'_j = \delta^+(j) \text{ and } u'_j \leqslant y^T\}$
$y^B := 0$; *stop* := false.

Repeat
$i := d(u)$
$P^L := P^L \cup \{v_i\}$
$y^B := y^B + v_i$
$u := u - v_i$
If $u > 0$ **then**
Compute j with $u \in I_j$.
If $\delta^+(j) + y^B \leqslant y^T$ and $d(\delta^+(j)) < i$ **then** $u := \delta^+(j)$
else if $\delta^-(j) + y^B \geqslant y^T - \varepsilon c$ and $d(\delta^-(j)) < i$ **then** $u := \delta^-(j)$
else *stop* := true.
until $u = 0$ or *stop*
$\Lambda^E := \Lambda^E \cup \{v_j \in \tilde{\Lambda} \mid j \geqslant i\}$
return (y^B).

Comment. A part of the sequence of items which led to a value within εc of y^T is reconstructed. The backtracking stops in particular if, in the dynamic programming arrays, an entry is found which, meeting the condition on the solution value, was however updated after the generation of the "forward arc" v_i. Such an updated entry must not be used because it may originate from a smaller entry which was generated by an item already used in the partial solution and hence this item would appear twice in the solution vector.

Procedure divide and conquer $(\hat{\Lambda}, \hat{c})$
Input: $\hat{\Lambda}$: subset of items, $\hat{c}$: subset capacity.
Output: y^{DC}: part of the solution value contained in $\hat{\Lambda}$.
(Divide)
Partition $\hat{\Lambda}$ into two disjoint subsets Λ_1, Λ_2 with cardinalities as equal as possible.
Perform procedure **relaxed dynamic programming** $(\Lambda_1, \hat{c})$ returning $\delta_1^-(), \delta_1^+(), d_1()$.
Perform procedure **relaxed dynamic programming** $(\Lambda_2, \hat{c})$ returning $\delta_2^-(), \delta_2^+(), d_2()$.
(Conquer)
Find entries u_1 and u_2 of the dynamic programming arrays $\delta_1^-(), \delta_1^+()$ and $\delta_2^-(), \delta_2^+()$, respectively, with $u_1 \geqslant u_2$ such that

$$\hat{c} - \varepsilon c \leqslant u_1 + u_2 \leqslant \hat{c}. \tag{2}$$

$y_1^{DC} := 0$; $y_2^B := 0$; $y_2^{DC} := 0$. *local variables*

(Resolve Λ_1)
Perform procedure **backtracking** $(\delta_1^-(), \delta_1^+(), d_1(), \Lambda_1, \hat{c} - u_2)$ returning y_1^B.
If $\hat{c} - u_2 - y_1^B > \varepsilon c$ **then**
perform procedure **divide and conquer** $(\Lambda_1 \backslash \Lambda^E, \hat{c} - u_2 - y_1^B)$ returning y_1^{DC}.

(Resolve Λ_2)
If $u_2 > 0$ **then begin**
If $\hat{c} - u_2 - y_1^B > \varepsilon c$ **then** perform procedure **relaxed dynamic programming** $(\Lambda_2, \hat{c} - y_1^B - y_1^{DC})$ returning $\delta_2^-(), \delta_2^+(), d_2()$. $(*)$
Perform procedure **backtracking** $(\delta_2^-(), \delta_2^+(), d_2(), \Lambda_2, \hat{c} - y_1^B - y_1^{DC})$ returning y_2^B.
If $\hat{c} - y_1^B - y_1^{DC} - y_2^B > \varepsilon c$ **then** perform procedure **divide and conquer** $(\Lambda_2 \backslash \Lambda^E, \hat{c} - y_1^B - y_1^{DC} - y_2^B)$ returning y_2^{DC}.
end
$y^{DC} = y_1^B + y_1^{DC} + y_2^B + y_2^{DC}$
return (y^{DC}).

3.0.2 Connection establishment between Subset Sum and Min-Plus Convolution problem and new FPTAS in time $\tilde{O}(n + \frac{1/\varepsilon^2}{2^{\Omega(\sqrt{(log(1/\varepsilon))})}})$

In this subsection, a new faster FPTAS was presented. After the approximation scheme of Kelleler et al. that we analyzed in the previous subsection, with time complexity $O(min\{n \cdot 1/\varepsilon, n + 1/\varepsilon^2 log(1/\varepsilon)\})$ and space $O(n + 1/\varepsilon)$, which was the fastest for 20 years the following question about approximating Subset Sum remained elusive.

Does Subset Sum *admit an approximation scheme in time* $O((n + 1/\varepsilon)^{2-\delta})$ *for some* $\delta > 0$*?*

After, the improvement of solving Subset Sum with a pseudopolynomial algorithm from Karl Bringmann in time $\tilde{O}(n + t)$, which is proved to be optimal in lower order factors, it seemed very challenging to find any higher lower bounds than $n + 1/\varepsilon$, up to lower order factors. However, in this paper the researches managed to find a better lower bound through establishing a connection with the MinPlus-Convolution problem. In detail, they proved that computing a $(1-1/n)$-approximation for Subset Sum is equivalent to the MinPlus-Convolution problem, thus adding the first approximation problem to the list of known MinConv-equivalent problems. This negatively answers the main question for Subset Sum approximation schemes running in strongly subquadratic time, conditional on the MinConv conjecture. Moreover, their reductions allow to transfer the known lower order improvements from MinConv to approximating Subset Sum which yields the first algorithmic improvement in over 20 years.

MinPlus-Convolution Problem In this problem we are given two sets $A, B \in \mathbb{Z}^n$ and the goal is to compute the sequence $C \in \mathbb{Z}^{2n}$ with $C[k] = min_{i+j=k} A[i] + B[j]$. The naive running time of $O(n^2)$ can be improved to $O(\frac{n^2}{2^{\Omega(\sqrt{logn})}})$ by a reduction to All Pairs Shortest by using Williams' algorithm. Despite, considerable attention has been shown for this problem no subquadratic algorithm has been detected. This was the reason for a hardness result that

was formulated for this problem. In particular, a Knapsack instance with weight budget W can be solved in time $O((n+W)^{2-\delta})$ if and only if MinPlus-Convolution can be solved in time $O((n+W)^{2-\delta'})$ for some $\delta' > 0$.

Contribution of this paper In this paper the researchers prove that $(1-\frac{1}{n})$-approximation for Subset Sum is equivalent to the MinConv problem, thus adding the first approximation problem to the list of known MinConv-equivalent problems. This negatively answers our main question for Subset Sum approximation schemes running in strongly subquadratic time, conditional on the MinConv conjecture. Moreover, our reductions allow us to transfer the known lower order improvements from MinConv to approximating Subset Sum which yields the first algorithmic improvement in over 20 years.

Formal statement of the results

In this section, we present the theorems that their results are based on. Actually, the researchers prove a subquadratic equivalence between Subset Sum and MinConv through the following reductions.

Theorem 3.1. *(Reduction from* Subset Sum *to MinConv)If MinConv can be solved in time $T(n)$, then* Subset Sum *has a randomized approximation scheme that is correct with high probability and runs in time $\tilde{O}(n+T(1/\varepsilon))$.*

This reduction even transfers the lower order improvements of the MinConv algorithm that runs in time $n^2/2^{\Omega(\sqrt{logn})}$. This yields the first improved approximation scheme for SubsetSum in over 20 years.

Corollary. *1.2 (Approximation Scheme for* Subset Sum*).* Subset Sum *has a randomized approximation scheme that is correct with high probability and runs in time*

$$\tilde{O}(n+\frac{(1/\varepsilon)^2}{2^{\Omega(\sqrt{(log(1/\varepsilon))}}})$$

'The second reduction is as follows:

Theorem 3.2. *(Reduction from MinConv to* Subset Sum *) If* Subset Sum *has an approximation scheme running in time $O((n+1/\varepsilon)^{2-\delta})$ for some $\delta > 0$ then MinConv can be solved in time $O(n^{2-\delta'})$ for some $\delta' > 0$.*

Under the MinConv conjecture this rules out the existence of approximation schemes for SubsetSum running in strongly subquadratic time $O((n+1/\varepsilon)^{2-\delta})$ for any $\delta > 0$. Taken together, the two reductions prove that SubsetSum has an approximation scheme in time $O((n+1/\varepsilon)^{2-\delta})$ if and only if MinConv can be solved in time $O(n^{2-\delta'})$ for some $\delta' > 0$.

Algorithm for Approximating Subset Sum

In this subsection we are going to describe analytically the reduction from approximating Subset Sum to exactly solving MinConv, thus proving Theorem 1.1. This implies the approximation of Corollary 1.2. In the paper, they state a different formulation of approximating Subset Sum:

Given X, t and $\varepsilon > 0$, return any subset $Y \subseteq X$ satisfying $\Sigma(Y) \leq t$ and $\Sigma(Y) \geq min\{OPT, (1-\varepsilon)t\}$.

Note that this is quite a strong problem variant, since for $OPT \leq (1-\varepsilon)t$ it requires to solve the problem exactly! In particular, an algorithm for the above problem variant implies a standard approximation scheme for Subset Sum.
From high level perspective, we adapt the exact pseudopolynomial algorithm for Subset Sum from [Brin20] by replacing its standard sumset subroutine by a novel approximate sumset computation, in addition to several further changes to make it work in the approximate setting. This yields an approximation scheme for Subset Sum , with black-box access to a MinConv algorithm. Throughout this section we assume to have access to an algorithm for MinConv running in time $T_{MinConv}(n)$ on sequences of length n. Since this is an exact algorithm, we can assume that $T_{MinConv}(n) = (n)$.

Definitions and Lemmas

In this subsection we present some definitions and lemmas which are utilized in this paper in order to support the correctness of the produced FPTAS.
(Approximation). Let $t, \Delta \in \mathcal{N}$. For any $A \subseteq [t]$ and $b \in \mathcal{N}$, we define the lower and upper approximation of b in A as:

$$apx_t^-(b, A) := max\{a \in A \cup \{t+1\} | a \leq b\}$$

$$apx_t^+(b, A) := min\{a \in A \cup \{t+1\} | a \geq b\}$$

We use the convention max := -∞ and min := ∞
For $A, B \subseteq N$, we say that A (t, Δ)-approximates B if $A \subseteq B \subseteq [t]$ and for any $b \in B$ we have:

$$apx_t^+(b, A) - apx_t^-(b, A) \leq \Delta$$

It is important to note that the approximations of b in A are not necessarily elements of A, since we add $t+1$. Throughout the analysis, we may use "b has good approximations in A", with the meaning that $apx_t^+(b, A) - apx_t^-(b, A) \leq \Delta$ holds.

Lemma 3.3. *(Transitivity). If A (t, Δ)-approximates B and B (t, Δ)-approximates C, then A (t, Δ)-approximates C*

Lemma 3.4. *If $A \subseteq B \subseteq C$ and A (t, Δ)-approximates C, then B (t, Δ)-approximates C.*

Lemma 3.5. *(Union Property). If A_1 (t, Δ)-approximates B_1 and A_2 (t, Δ)-approximates B_2, then $A_1 \cup A_2$ (t, Δ)-approximates $B_1 \cup B_2$.*

Lemma 3.6. *(Sumset Property). If A_1 (t, Δ)-approximates B_1 and A_2 (t, Δ)-approximates B_2, then $A_1 +_t A_2$ (t, Δ)-approximates $B_1 +_t B_2$.*

(Sparsity). Let $A \in N$ and $\Delta \in N$. We say that A is Δ-sparse if $|A \cap [x, x+\Delta]| \leq 2$ holds for any $x \in N$. If A is Δ-sparse and A (t, Δ)-approximates B, we say that A sparsely (t, Δ)-approximates B. The previous definition shows that is always valid to assume approximation sets have small size, or more precisely to be locally sparse.

Lemma 3.7. *(Sparsification). Given $t, \Delta \in N$ and a set $B \subseteq [t]$, in time $O(|B|)$ we can compute a set A such that A sparsely (t, Δ)-approximates B.*

It is easy to compute A in time $O(|B|)$ by one sweep from left to right, assuming that B is given in sorted order. Pseudocode for this is presented in the following Algorithm 1.

Algorithm 1 `Sparsification`(B, t, Δ): Given $t, \Delta > 0$ and a set $B \subseteq [t]$ in sorted order, compute a set A that sparsely (t, Δ)-approximates B. We denote the elements of B by $B[1], \ldots, B[m]$.

Initialize $A := \emptyset$ and $n := 0$
for $i = 1, \ldots, m$ **do**
    $n := n + 1$
    $A[n] := B[i]$
    if $n \geq 3$ and $A[n] - A[n-2] \leq \Delta$ **then**
        $A[n-1] := A[n]$
        $n := n - 1$
return $\{A[1], \ldots, A[n]\}$

Lemma 3.8. *(Down Shifting). Let $t, t', \Delta \in \mathcal{N}$ with $t \geq t'$. If A (t, Δ)-approximates B, then $A \cap [t']$ (t, Δ)-approximates $B \cap [t']$.*

Lemma 3.9. *(Up Shifting). Let $t, t', \Delta \in \mathcal{N}$ with $t \leq t'$. If A (t, Δ)-approximates B, then there exists $u \in [t - \Delta, t]$ such that A (t', Δ)-approximates $B \cap [u]$.*

Algorithm for Approximate Sumset Computation

Now the main connection to MinConv is presented. In this subsection it is shown how to compute for given A_1, A_2 a set A that approximates $A_1 + A_2$, by performing two calls to MinConv. At first, we set $t := \infty$, so that we do not have to worry about the upper end. In this way, we have the following.

Lemma 3.10. *(Unbounded Sumset Computation). Given $t, \Delta \in \mathcal{N}$ with $t \geq \Delta$ and Δ-sparse sets $A_1, A_2 \subseteq [t]$, in time $O(T_{MinConv}(t/\Delta)$*

Proof. To simplify notation, for this proof we introduce the symbol $\perp$ indicating an undefined value. We let $min = max = \perp$. Furthermore, we let $x + \perp = \perp$ and $min\{x, \perp\} = max\{x, \perp\} = x$ .This gives rise to natural generalizations of MinConv and MaxConv to sequences over$Z \cup \{\perp\}$. We call an entry of such a sequence defined if it is not $\perp$. Note that since $\perp$ acts as a neutral element for the min and max operations, we can think of $\perp$ being ∞ for MinConv, and $-\infty$ for MaxConv. The fact that these neutral elements, ∞ and $-\infty$, are different is the reason why we introduce $\perp$.
Observe that if MinConv on sequences over $\{-M, ..., M\}$ is in time $T_{MinConv}(n)$, then also MinConv on sequences over $\{-M/4, ..., M/4\} \cup \{\perp\}$ is in time $O(T_{MinConv}(n))$. Indeed, replacing $\perp$ by M, any output value in $[-M/2, M/2]$ is computed correctly, while any output value in $[3M/4, 2M]$ corresponds to $\perp$. Also observe that MaxConv is equivalent to MinConv by negating all input and output values, and therefore MaxConv is also in time $O(T_{MinConv}(n))$.
The algorithm is as follows. Set $n := 4\lceil t, \Delta \rceil$. We consider intervals $I_i := [i\Delta/2, (i+1)\Delta/2]$ for $0 \leq i < n$. Since A_1, A_2 are Δ-sparse, they contain at most two elements in any interval I_i . We may therefore "unfold" the sets A_1, A_2 into vectors X_1, X_2 of length $2n$ as follows. For $r \in \{1, 2\}$ and $0 \leq i < n$ we set:

$$Xr[2i] := min(I_i \cap A_r),$$

$$Xr[2i+1] := max(I_i \cap A_r).$$

We then compute the sequences

$$C^{-} := MinConv(X_1, X_2),$$

that is,

$$C^-[k] = min_{i+j=k} X_1[i] + X_2[j],$$

and also

$$C^+ := MaxConv(X_1, X_2),$$

that is,

$$C^+[k] = max_{i+j=k} X_1[i] + X_2[j],$$

for $0 \leq k < 4n$. Finally, we return the set A containing all defined entries of C^- and C^+. Clearly, this algorithm runs in $O(T_{MinConv}(t/\Delta))$.

Correctness. We should prove that for any $a_1 \in A_1, a_2 \in A_2$ their sum $a_1 + a_2$ has good approximations in A. Let $0 \leq i^*, j^* < 2n$ be such that $X_1[i^*] = a_1$ and $X_2[j^*] = a_2$ and let $k^* := i^* + j^*$. Then, by definition we have:

$$C^-[k^*] \leq a_1 + a_2 \leq C^+[k^*].$$

It remains to prove that $C^+[k^*] - C^-[k^*] \leq \Delta$. From the construction of $X_r[2i]$ and $X_r[2i+1]$ it follows that any defined entry satisfies $X_r[i] \in [(i-1)\Delta/4, (i+1)\Delta/4]$. In particular, the sum of two defined entries satisfies $X_1[i] + X_2[j] \in [(i+j-2)\Delta/4, (i+j+2)\Delta/4]$. This yiels:

$$C^-[k^*], C^+[k^*] \in [(k^*-2)\Delta/4, (k^*+2)\Delta/4] \cup \{\}.$$

Moreover, at least one summand, $X_1[i^*]+X_2[j^*] = a_1+a_2$, is defined and thus $C^-[k^*], C^+[k^*] \neq$. This yields $C^+[k^*] - C^-[k^*] \leq \Delta$. As a conclusion, we have that $a_1 + a_2 \in A_1 + A_2$ has good approximations in A which finishes the proof. □

Lemma 3.11. *(Capped Sumset Computation). Let $t, \Delta \in \mathcal{N}$ and $B_1, B_2 \subseteq [t]$. Set $B := B_1 +_t B_2$ and suppose that A_1 sparsely (t, Δ)-approximates B_1 and A_2 sparsely (t, Δ)-approximates B_2. In this situation, given A_1, A_2, t, Δ, we can compute a set A that sparsely (t, Δ)-approximates B in time $O(T_{MinConv}(t/\Delta))$. We call this algorithm* CappedSumset(A_1, A_2, t, Δ).

Proof. By Lemma 4.5, $A_1 +_t A_2$ (t, Δ)-approximates B. Using Lemma 4.10, we can compute a set A' that (∞, Δ)-approximates $A_1 + A_2$. By Lemma 4.8 (Down Shifting), $A'' := A' \cap [t]$ (t, Δ)-approximates $(A_1 + A_2) \cap [t] = A_1 +_t A_2$. Using Lemma 4.7, given A'' we can compute a set A that sparsely (t, Δ)-approximates A''. By Lemma 4.2 (Transitivity), these three steps imply that A (t, Δ)-approximates B. Since A is Δ-sparse, A also sparsely (t, Δ)-approximates B.
Each of these steps runs in time $O(t/\Delta)$ or in time $O(T_{MinConv}(t/\Delta))$. Since we can assume $T_{MinConv}(n) = \Omega(n)$, we can bound the total running time by $O(T_{MinConv}(t/\Delta))$. □

Algorithms for Subset Sum

Using the lemmas that we proved above in this subsection we are going to present the approximation algorithm for the Subset Sum of this paper. The main difference is that they use Lemma 4.11 instead of the usual sumset computation by Fast Fourier Transform, but significant changes are required in order to make it work.

Given (X, t, Δ) where X has size n, the goal is to compute a set A that sparsely (t, Δ)-approximates the set $\mathcal{S}(X; t) = \{\Sigma(Y) | Y \subseteq X, \Sigma(Y) \leq t\}$.

. We say that an event happens with *high probability* if its probability is at least $1 - min\{1/n, \Delta/t\}^c$ for some constant $c > 0$ that we are free to choose as any large constant. We say that A w.h.p. (t, Δ)-approximates B if we have:

- $A \subseteq B$, and
- with high probability A (t, Δ)-approximates B.

Color Coding

We present the algorithm that the researchers use in order to solve SUBSET SUM in case all items are large, i.e. $X \subseteq [t/k, t]$ for a parameter k.

Lemma 3.12. *(Color Coding). Given $t, \Delta, k \in N$ with $t \geq \Delta$ and a set $X \subseteq [t/k, t]$ of size n, we can compute a set A that w.h.p. sparsely (t, Δ)-approximates $S(X; t)$, in time*

$$O((n + k^2 \cdot T_{MinConv}(t/\Delta))log(nt/\Delta)).$$

Proof. Denote by $X_1, ..., X_{k^2}$ a random m partitioning of X, that is, for every $x \in X$ we choose a number j uniformly and independently at random and we put x into X_j. For any subset $Y \in X$ with $\Sigma(Y) \leq t$, note that $|Y| \leq k$ since $X \subseteq [t/k, t]$, and consider how the random partitioning acts on Y . We say that the partitioning splits Y if we have $|Y \cap X_j| \leq 1$ for any $1 \leq j \leq k^2$. By the birthday paradox, Y is split with constant probability. More precisely, we can view the partitioning restricted to Y as throwing $|Y| \leq k$ balls into k^2 bins. Thus, the probability that Y is split is equal to the probability that the second ball falls into a different bin than the first, the third ball falls into a different bin than the first two, and so on, which has probability:

$$\frac{k^2-1}{k^2} \cdot \frac{k^2-2}{k^2} \cdots \frac{k^2-(|Y|-1)}{k^2} \geq (\frac{k^2-(|Y|-1)}{k^2})^{|Y|} \geq (1-1/k)^k \geq (1/2)^2 = 1/4.$$

We make use of this splitting property. Let $X_j^i := X_j \cup \{0\}$ and

$$T := X_1' +_t .... +_t X_{\dot{k}^2}^{'}$$

Observe that $T \subseteq S(X; t)$, since each sum appearing in T uses any item $x \in X$ at most once. We claim that if Y is split then T contains $\Sigma(Y)$. Indeed, in any part X_j with $|Y \cap X_j| = 1$ we pick this element of Y, and in any other part we pick $0 \in X_j'$, to form $\Sigma(Y)$ as a sum appearing in $T = X_1' +_t ... +_t X_{k^2}'$. Hence, we have $\Sigma(Y) \in T$ with probability at least $1/4$. To boost the success probability, we repeat the above random experiment several times. More precisely, for $r = 1, ..., Clog(nt/\Delta)$ we sample a random partitioning $X = X_{r,1} \cup ... \cup X_{r,k^2}$, set $X_{r,i}' := X_{r,i} \cup \{0\}$, and consider $T_r := X_{r,1}' +_t ... +_t X_{r,k^2}'$. Since we have $\Sigma(Y) \in T_r$ with probability at least $1/4$, we obtain $\Sigma(Y) \in \bigcup_r T_r$ with high probability. Moreover, we have $\bigcup_r T_r \subseteq S(X; t)$. Let $\mathcal{S}_{\Delta-sp}(X; t)$ be a sparsification of $\mathcal{S}(X; t)$, and note that it has size $|\mathcal{S}_{\Delta-sp}(X; t)| = O(t/\Delta)$ and can be found in linear time by Lemma 4.7. Since we use "with high probability" to denote a probability of at least $1-min\{1/n, \Delta/t\}^c$ for large constant c, we can afford a union bound over the $O(t/\Delta)$ elements of $S_{\Delta-sp}(X; t)$ to infer that with high probability

$$\mathcal{S}_{\Delta-sp}(X; t) \subseteq \bigcup_r T_r \subseteq \mathcal{S}(X; t).$$

Since $\mathcal{S}_{\Delta-sp}(X; t)$ (t, Δ)-approximates $\mathcal{S}(X; t)$, Lemma 4.3 implies that

$$\bigcup_r T_r \text{ w.h.p. } (t, \Delta)\text{-approximates } \mathcal{S}(X; t).$$

We cannot afford to compute any T_r explicitly, but we can compute approximations of these sets. To this end, let $Z_{r,j}$ be the sparsification of $X_{r,j}'$ given by Lemma 4.7. We start with $A_{r,0} := \{0\}$ and repeatedly compute the capped sumset with $Z_{r,j}$, setting $A_{r,j} := CappedSumset(A_{r,j-1}, Z_{r,j}, t, \Delta)$ for $1 \leq j \leq k^2$. It now follows inductively from Lemma 4.11 that $A_{r,j}$ sparsely (t, Δ)-approximates $X_{r,1}' +_t ... +_t X_{r,j}'$. Hence, A_{r,k^2} sparsely (t, Δ)-approximates T_r. Let $A' := \bigcup_r A_{r,k^2}$. By Lemma 4.4, A' (t, Δ)-approximates $\bigcup_r T_r$. With the previous relation and transitivity, A' w.h.p. (t, Δ)-approximates $\mathcal{S}(X; t)$. Finally, we sparsify A' using Lemma 4.7 to obtain a subset A that sparsely (t, Δ)-approximates A'. By transitivity, A w.h.p. sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$. □

The running time is immediate from Lemmas 4.7 and 4.11

Algorithm 2 `ColorCoding`(X, t, Δ, k): Given $t, \Delta \in \mathbb{N}$ and a set $X \subseteq [t/k, t]$ in sorted order, we compute a set A that w.h.p. sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$.

```
for r = 1, ..., C log(nt/Δ) do
    randomly partition X = X_{r,1} ∪ ... ∪ X_{r,k^2}
    A_{r,0} := {0}
    for j = 1, ..., k^2 do
        X'_{r,j} := X_{r,j} ∪ {0}
        Z_{r,j} := Sparsification(X'_{r,j}, t, Δ)
        A_{r,j} := CappedSumset(A_{r,j-1}, Z_{r,j}, t, Δ)
return Sparsification(⋃_r A_{r,k^2}, t, Δ)
```

In the following section we are going to present a greedy algorithm, which is used as a special treatment of the case that all items are small, that is $max(X) \leq \Delta$. In this case, we pick any ordering of $X = \{x_1, ..., x_n\}$ and let P denote the set of all prefix sums $0, x_1, x_1 + x_2, x_1 + x_2 + x_3, ...$ that are bounded by t, i.e. $P = \{\Sigma_{i=1}^{j} x_i | 0 \leq j \leq n\} \cap [t]$. We return a sparsification A of P.

Claim 3.13. *P (t, Δ)-approximates $\mathcal{S}(X; t)$.*

Proof. Clearly $P \subseteq \mathcal{S}(X; t)$. Moreover, any $s \in [0, max(P)]$ falls into some interval between two consecutive prefix sums, and such an interval has length x_i for some i. Hence, we have

$$apx_t^+(s, P) - apx_t^-(s, P) \leq x_i \leq max(X) \leq \Delta.$$

We now do a case distinction on $\Sigma(X)$. If $\Sigma(X) < t$, then observe that $max(P) = \Sigma(X) = max(\mathcal{S}(X; t))$. Therefore, the interval $[0, max(P)]$ already covers all $s \in \mathcal{S}(X; t)$ and we are done.

Otherwise, if $\Sigma(X) \geq t$, then observe that $max(P) > t - \Delta$, as otherwise we could add the next prefix sum to P. In this case, for any $s \in [max(P), t]$,

$$apx_t^+(s, P) - apx_t^-(s, P) \leq t + 1 - max(P) \leq \Delta.$$

In total, every $s \in \mathcal{S}(X; t)$ has good approximations in P.

From Claim 4.13 and transitivity it follows for A = Sparsification(P, t, Δ) that A sparsely (t, Δ)-approximates S(X;t). We thus proved the following lemma. □

Lemma 3.14. *(Greedy). Given integers $t, \Delta > 0$ and a set $X \subseteq [t]$ of size n satisfying $max(X) \leq \Delta$ we can compute a set A that sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$ in time $O(n)$.*

Algorithm 3 `Greedy`(X, t, Δ): Given $t, \Delta \in \mathbb{N}$ and a set $X = \{x_1, \ldots, x_n\} \subseteq [t]$ with $\max(X) \leq \Delta$, we compute a set A that sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$.

```
P := {0}, s := 0, i := 1
while i ≤ n and s + x_i ≤ t do
    s := s + x_i
    P := P ∪ {s}
    i := i + 1
A := Sparsification(P, t, Δ)
return A
```

Recursive Splitting

Afterwards, in the paper they provide a recursive algorithm which makes use of Color Coding and Greedy. This is actually a simplified version of the algorithms FasterSubsetSum and ColorCodingLayer from [Brin21] adapted to the approximation setting.

Given a set $X \subseteq N$ of size n and numbers $t, \Delta > 0$, our goal is to compute a set A that sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$. We assume that initially $t \geq 8\Delta$. We will use parameters k and η, which are set before the first call of the algorithm to $k := max\{8, Clog3(nt/\Delta)\}$ and $\eta := 1/(2log(t/\Delta))$. We can assume that $X \subseteq [t]$, since larger numbers cannot be picked for subset sums in $[t]$.
We partition X into the large numbers $X_L := X \cap [t/k, t]$ and the small numbers $X_S := X \setminus X_L$. On the large numbers we compute $A_L := ColorCoding(X_L, t, \Delta, k)$, so that A_L w.h.p. sparsely (t, Δ)-approximates $\mathcal{S}(X_L; t)$. We then randomly partition the small numbers X_S into subsets X_1, X_2, that is, for any $x \in X_S$ we choose a number $j \in \{1, 2\}$ uniformly at random and we put x into X_j . We recursively call the same algorithm on (X_1, t', Δ) and on (X_2, t', Δ) for the new target bound $t' := (1 + \eta)t/2 + \Delta$. Call the results of these recursive calls A_1, A_2. Finally, we combine $A_1, A_2 to A_S$, and A_S, A_L to A, by capped sumset computations. We return A. The base case happens when $max(X) \leq \Delta$, where we run Greedy. Next we give the pseudocode of this Algorithm.

Algorithm 4 `RecursiveSplitting`(X, t, Δ): Given $t, \Delta \in \mathbb{N}$ and a set $X \subseteq [t]$ in sorted order, we compute a set A that sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$. The parameters k, η are set before the first call of the algorithm to $k := \max\{8, C \log^3(nt/\Delta)\}$ and $\eta := 1/(2 \log(t/\Delta))$.

1: **if** $\max(X) \leq \Delta$ **then return** `Greedy`(X, t, Δ)
2: $X_L := X \cap [t/k, t]$, $X_S := X \setminus X_L$
3: randomly partition $X_S = X_1 \cup X_2$
4: $t' := (1 + \eta)t/2 + \Delta$
5: $A_L :=$ `ColorCoding`(X_L, t, Δ, k)
6: $A_1 :=$ `RecursiveSplitting`(X_1, t', Δ)
7: $A_2 :=$ `RecursiveSplitting`(X_2, t', Δ)
8: $A_S :=$ `CappedSumset`(A_1, A_2, t, Δ)
9: $A :=$ `CappedSumset`(A_L, A_S, t, Δ)
10: **return** A

The analysis of this algorithm implies the following lemma.

Lemma 3.15. *(Recursive Splitting). Given integers $t, \Delta > 0$ with $t \geq 8\Delta$ and a set $X \subseteq [t]$ of size n, we can compute a set A that sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$ in time*

$$O((n + T_{MinConv}(t/\Delta))log^8(nt/\Delta)).$$

Correctness

We inductively prove that with high probability for any recursive call of method RecursiveSplitting(X, t, Δ) the output A sparsely (t, Δ)-approximates $\mathcal{S}(X; t)$. Note that, as an output of CappedSumset, A is clearly Δ-sparse, and thus we only need to show that A (t, Δ)-approximates $\mathcal{S}(X; t)$. Since the recursion tree has total size $O(t/\Delta)$, we can afford a union bound over all recursive calls. In particular, if we prove correctness of one recursive call with high probability, then

the whole recursion tree is correct with high probability. Therefore,in the following we consider one recursive call.

In the following we present three more lemmas which complete the proof of correctness of the approximation algorithm and finish the presentation of the algorithm.

Lemma 3.16. *With high probability, there exist* $t_1, t_2 \in [(1+\eta)t/2, (1+\eta)t/2+\Delta]$ *such that* $A(t,\Delta)$*-approximates* $\mathcal{S}(X_L;t) +_t S(X_1;t_1) +_t \mathcal{S}(X_2;t_2)$.

Lemma 3.17. *Let* $\mathcal{S}_{\Delta-sp}(X_S;t)$ *be the sparsification of* $\mathcal{S}(X_S;t)$ *given by Lemma 4.7 and let* $\bar{t} = (1+\eta)t/2$. *With high probability we have* $\mathcal{S}_{\Delta-sp}(X_S;t) \subseteq \mathcal{S}(X_1;\bar{t}) +_t \mathcal{S}(X_2;t)$

Observation 1. *For any partitioning* $Z = Z_1 \cup Z_2$ *we have* $\mathcal{S}(Z_1,t) +_t \mathcal{S}(Z_2,t) = \mathcal{S}(Z;t)$.

Lemma 3.18. *A with high probability* (t,Δ)*-approximates* $\mathcal{S}(X;t)$.

Finishing the proof

We show how to use **RecursiveSplitting** to obtain an approximation scheme for SubsetSum in time $\tilde{O}(n + T_{MinConv}(1/\varepsilon))$. Note that this proves Theorem 1.1 as well as Corollary 1.2. Given X, t and $\varepsilon > 0$, let $OPT := max(\mathcal{S}(X;t))$. Set $\Delta := min\{\varepsilon t, t/8\}$ and call the procedure RecursiveSplitting(X,t,Δ) to obtain a set A that w.h.p. (t,Δ)-approximates $\mathcal{S}(X;t)$.

Claim 3.19. *With high probability, we have* $max(A) \geq min\{OPT, (1-\varepsilon)t\}$.

Proof. Consider $apx_t^+(OPT,A)$ and $apx_t^-(OPT,A)$. Since $\mathcal{S}(X;t)$ does not contain any numbers in $(OPT,t]$, and $A \subseteq \mathcal{S}(X;t)$, we have $apx_t^+(OPT,A) \in \{OPT, t+1\}$. If $apx_t^+(OPT,A) = OPT$, then A contains OPT, so $max(A) \geq OPT$. Otherwise, if $apx_t^+(OPT,A) = t+1$, then $apx_t^-(OPT,A) \geq apx_t^+(OPT,A) - \Delta > t - \varepsilon t$. In particular, $max(A) \geq (1-\varepsilon)t$. □

We have thus shown how to compute a subset sum $max(A)$ with $max(A) \geq min\{OPT, (1-\varepsilon)t\}$. It remains to determine a subset $Y \subseteq X$ summing to $max(A)$. To this end, we retrace the steps of the algorithm, using the following idea. If $a \in$ CappedSumset(A_1, A_2, t, Δ), then $a \in A_1 + A_2$, and thus we can simply iterate over all $a_1 \in A_1$ and check whether $a - a_1 \in A_2$, to reconstruct a pair $a_1 \in A_1, a_2 \in A_2$ with $a = a_1 + a_2$ in linear time. Starting with $max(A)$, we perform this trick in each recursive call of the algorithm, to reconstruct a subset summing to $max(A)$. The total running time of this algorithm is $O((n + T_{MinConv}(1/\varepsilon))log^8(n/\varepsilon))$.

Chapter 4

Subset Sum Ratio Approximation Algorithms

In this chapter, we are going to present two approximation schemes for the Subset Sum Ratio problem. The first is the fastest FPTAS presented up until now with time complexity $O(n^4/\varepsilon)$ and the second FPTAS is the research contribution of this dissertation which improves upon the time complexity of the previously fastest FPTAS, by achieving complexity $\tilde{O}(\min\{\frac{n^{2.3}}{\varepsilon^{2.6}}, \frac{n^2}{\varepsilon^3}, \frac{n^2}{\varepsilon^2}\})$.

4.0.1 FPTAS in $O(n^4/\varepsilon)$

The centerpiece of this paper is the Subset Sum Ratio problem, the optimization version of Equal Subset Sum, which asks, given an input set $S \subseteq \mathbb{N}$, for two disjoint subsets $S_1, S_2 \subseteq S$, such that the following ratio is minimized

$$\frac{\max\{\sum_{s_i \in S_1} s_i, \sum_{s_j \in S_2} s_j\}}{\min\{\sum_{s_i \in S_1} s_i, \sum_{s_j \in S_2} s_j\}}$$

In this paper, the researchers develop a new FPTAS for the Subset Sum Ratio problem which builds on techniques proposed in [D. Nanongkai, Simple FPTAS for the subset-sums ratio problem, Inf. Proc. Lett. 113 (2013)]. One of the key improvements of our scheme is the use of a dynamic programming table in which one dimension represents the difference of the sums of the two subsets. This idea, together with a careful choice of a scaling parameter, yields an FPTAS that is several orders of magnitude faster than the best currently known scheme of [C. Bazgan, M. Santha, Z. Tuza, Efficient approximation algorithms for the Subset-Sums Equality problem.
At first, we present some functions that are defined in the paper which are used throughout their algorithm.

Definition 1 (Ratio of two subsets) Given a set $A = \{a_1, ..., a_n\}$ of n positive integers and two sets $S_1, S_2 \subseteq \{1, ..., n\}$ we define $R(S_1, S_2, A)$ as follows:

$$\mathcal{R}(S_1, S_2, A) = \begin{cases} \frac{\Sigma_{i \in S_1} a_i}{\Sigma_{j \in S_2} a_j} & if S_1 \cup S_2 \neq \emptyset \\ +\infty & otherwise \end{cases}$$

Definition 2 (Max Ratio) Given a set $A = \{a_1, ..., a_n\}$ of n positive integers and two sets $S_1, S_2 \subseteq \{1, ..., n\}$ we define $\mathcal{MR}(S_1, S_2, A)$ as follows:

$$\mathcal{MR}(S_1, S_2, A) = max\{\mathcal{R}(S_1, S_2, A), \mathcal{R}(S_2, S_1, A)\}$$

In the following, it is important to note that in this paper inspired by the restricted version of Subset Sum Ratio that was defined and solved in the paper of D. Nanongkai, they observed

that one of the restrictions in the restricted version is not necessary, thus a semi-restricted version of SUBSET SUM RATIO is defined. All the versions of SUBSET SUM RATIO problem that are we mentioned are defined as follows:

SUBSET SUM RATIO problem (SSR) (equivalent definition). Given a set $A = \{a_1, ..., a_n\}$ of n positive integers, find two disjoint sets $S_1, S_2 \subseteq \{1, ..., n\}$ such that the value $\mathcal{MR}(S_1, S_2, A)$ is minimized. In addition, from now on, whenever we have a set $A = \{a_1, ..., a_n\}$ we will assume that $0 < a_1 < a_2 < ... < a_n$ (clearly, if the input contains two equal numbers then the problem has a trivial solution). The FPTAS proposed by Nanonghai approximates the SSR problem by solving a restricted version.

Restricted Subset-Sums Ratio problem . Given a set $A = \{a_1, ..., a_n\}$ of n positive integers and two integers $1 \leq p < q \leq n$, find two disjoint sets $S_1, S_2 \subseteq \{1, ..., n\}$ such that $\{maxS_1, maxS_2\} = \{p, q\}$ and the value $\mathcal{MR}(S_1, S_2, A)$ is minimized. Inspired by this idea, they define a less restricted version. The new problem requires one additional input integer, instead of two, which represents the smallest of the two maximum elements of the sought optimal solution.

Semi-Restricted Subset-Sums Ratio problem . Given a set $A = \{a_1, ..., a_n\}$ of n positive integers and an integer $1 \leq p < n$, find two disjoint sets $S_1, S_2 \subseteq \{1, ..., n\}$ such that $maxS_1 = p < maxS_2$ and the value $\mathcal{MR}(S_1, S_2, A)$ is minimized. Let $A = \{a_1, ..., a_n\}$ be a set of n positive integers and $p \in \{1, ..., n-1\}$. Observe that, if S_1^*, S_2^* is the optimal solution of SSR problem of instance A and S_1^p, S_2^p the optimal solution of Semi-Restricted SSR problem of instance A, p then: $\mathcal{MR}(S_1, S_2^*, A) = min_{p \in \{1,...,n-1\}} \mathcal{MR}(S_1^p, S_2^p, A)$ .Thus, we can find the optimal solution of SSR problem by solving the SSR Semi-Restricted SSR problem for all $p \in \{1, ..., n-1\}$.

Pseudo-polynomial time algorithm for Semi-Restricted SSR problem Let the A, p be an instance of the Semi-Restricted SSR problem where $A = \{a_1, ..., a_n\}$ and $1 \leq p < n$. For solving the problem we have to check two cases for the maximum element of the optimal solution. Let S_1^*, S_2^* be the optimal solution of this instance and $maxS_2^* = q$. We define $B = \{a_i | i > p, a_i < \Sigma_{j=1}^{p} a_j\}$ and $C = \{a_i | a_i \geq \Sigma_{j=1}^{p} a_j\}$ from which we have that either $a_q \in B$ or $a_q \in C$. Note that $A = \{a_1, ..., a_p\} \cup B \cup C$.

Case 1 ($a_q \in C$). It is easy to see that if $a_q \in C$, then $a_q = minC$ and the optimal solution will be $(S_1 = \{1, ..., p\}, S_2 = \{q\})$. We describe below a function that returns this pair of sets, thus computing the optimal solution if Case 1 holds.

Definition 2. *(Case 1 solution). Given a set $A = \{a_1, ..., a_2\}$ of n positive integers and an integer $1 \leq p < n$ we define the function $\mathcal{SOL}_1(A, p)$ as follows:*

$$\mathcal{SOL}_1 = \begin{cases} (\{1, ...p\}, \{minC\}) & if C \neq \emptyset \\ (\emptyset, \emptyset) & otherwise \end{cases}$$

where $C = \{a_i | a_i > \Sigma_{j=1}^{p} a_j\}$

Case 2($a_q \in B$). This second case is not trivial. Here, we define an integer P and $m = max\{j|a_j \in A\backslash C\}$ and a matrix T, where $T[i,d], 0 \leq i \leq m, -2{\cdot}\Sigma_{k=1}^{p} a_k \leq d \leq \Sigma_{k=1}^{p} a_k$, is a quadruple to be defined below. A cell $T[i,d]$ is nonempty if there exist two disjoint sets S_1, S_2 with sums sum_1, sum_2 such that $sum_1 - sum_2 = d$, $maxS_1 = p$, and $S_1 \cup S_2 \subseteq \{1, ..., i\} \cup \{p\}$; if $i > p$, we require in addition that $p < maxS_2$. In such a case, cell $T[i,d]$ consists of the two sets S_1, S_2, and two integers $max(S_1 \cup S_2)$ and $sum_1 + sum_2$. A crucial point in this algorithm is that if there exist more than one pairs of sets which meet the required conditions, they keep the one that maximize the value $sum_1 + sum_2$ for convenience, they use a function to check this property and select the appropriate sets. The algorithm for this case (Algorithm 1) finally returns the pair S_1, S_2 which, among those that appear in some $T[m,d] \neq \emptyset$, has the smallest ratio $MR(S_1, S_2, A)$.

Definition 3. *(Larger total sum tuple selection). Given two tuples* $v_1 = (S_1, S_2, q, x)$ *and* $v_2 = (S_1', S_2', q', x')$ *we define the function* $\mathcal{LTST}(v_1, v_2)$ *as follows:*

$$\mathcal{LTST}(v_1, v_2) = \begin{cases} v_2 & v_1 = \emptyset || x' > x \\ v_1 & otherwise \end{cases}$$

Algorithm 1 Case 2 solution [$\mathcal{SOL}_2(A,p)$ function]

Input: a strictly sorted set $A = \{a_1, \ldots, a_n\}$, $a_i \in \mathbb{Z}^+$, and an integer p, $1 \leq p < n$.
Output: the sets of an optimal solution for Case 2.

$S_1' \leftarrow \emptyset$, $S_2' \leftarrow \emptyset$
$Q \leftarrow \sum_{i=1}^{p} a_i$, $m \leftarrow \max\{i \mid a_i < Q\}$
if $m > p$ **then**
  for all $i \in \{0, \ldots, m\}$, $d \in \{-2 \cdot Q, \ldots, Q\}$ **do**
    $T[i,d] \leftarrow \emptyset$
  end for
  $T[0, a_p] \leftarrow (\{p\}, \emptyset, p, a_p)$ ▷ $p \in S_1$ by problem definition
  for $i \leftarrow 1$ **to** m **do**
    if $i < p$ **then**
      for all $T[i-1,d] \neq \emptyset$ **do**
        $(S_1, S_2, q, x) \leftarrow T[i-1,d]$
        $T[i,d] \leftarrow \mathcal{LTST}(T[i,d], T[i-1,d])$
        $T[i,d+a_i] \leftarrow \mathcal{LTST}(T[i,d+a_i], (S_1 \cup \{i\}, S_2, q, x+a_i))$
        $T[i,d-a_i] \leftarrow \mathcal{LTST}(T[i,d-a_i], (S_1, S_2 \cup \{i\}, q, x+a_i))$
      end for
    else if $i = p$ **then** ▷ p is already placed in S_1
      for all $T[i-1,d] \neq \emptyset$ **do**
        $T[i,d] \leftarrow T[i-1,d]$
      end for
    else
      for all $T[i-1,d] \neq \emptyset$ **do**
        $(S_1, S_2, q, x) \leftarrow T[i-1,d]$
        if $i > p+1$ **then**
          $T[i,d] \leftarrow \mathcal{LTST}(T[i,d], T[i-1,d])$
        end if
        if $d - a_i \geq -2 \cdot Q$ **then**
          $T[i,d-a_i] \leftarrow \mathcal{LTST}(T[i,d-a_i], (S_1, S_2 \cup \{i\}, i, x+a_i))$
        end if
      end for
      for all $T[p,d] \neq \emptyset$ **do**
        $(S_1, S_2, q, x) \leftarrow T[p,d]$
        if $d - a_i \geq -2 \cdot Q$ **then**
          $T[i,d-a_i] \leftarrow \mathcal{LTST}(T[i,d-a_i], (S_1, S_2 \cup \{i\}, i, x+a_i))$

```
            end if
        end for
    end if
    for d ← −2 · Q to Q do
        (S1, S2, q, x) ← T[m, d]
        if MR(S1, S2, A) < MR(S1', S2', A) then
            S1' ← S1, S2' ← S2
        end if
    end for
end if
return S1', S2'
```

Next, we present the complete algorithm for Semi-Restricted SSR (Algorithm 2) which returns the best of the two solutions obtained by solving the two cases. The complexity of the algorithm 2 is $O(n \cdot Q)$, where $Q = \Sigma_{i=1}^{p} a_i$.

Algorithm 2 Exact solution for Semi-Restricted SSR [$\mathcal{SOL}_{ex}(A, p)$ function]

Input: a strictly sorted set $A = \{a_1, \ldots, a_n\}$, $a_i \in \mathbb{Z}^+$, and an integer p, $1 \leq p < n$.
Output: the sets of an optimal solution of Semi-Restricted SSR.

```
(S1, S2) ← SOL1(A, p)
(S1', S2') ← SOL2(A, p)
if MR(S1, S2, A) ≤ MR(S1', S2', A) then
    return S1, S2
else
    return S1', S2'
end if
```

FPTAS for Semi-Restricted SSR and SSR Algorithm 2, which we presented at Section 3, is an exact pseudo-polynomial time algorithm for the Semi-Restricted SSR problem. In order to derive a $(1+\varepsilon)$-approximation algorithm we will define a scaling parameter $\delta = \frac{\varepsilon \cdot a_p}{3 \cdot n}$ which we will use to make a new set $A' = \{a'_1, ..., a'_n\}$ with $a'_i = \lfloor \frac{a_i}{\delta} \rfloor$. The approximation algorithm solves the problem optimally on input (A', p) and returns the sets of this exact solution. The ratio of those sets is a $(1+\varepsilon)$-approximation of the optimal ratio of the original input.

Algorithm 3 FPTAS for Semi-Restricted SSR [$\mathcal{SOL}_{apx}(A, p, \varepsilon)$ function]

Input: a strictly sorted set $A = \{a_1, \ldots, a_n\}$, $a_i \in \mathbb{Z}^+$, an integer p, $1 \leq p < n$, and an error parameter $\varepsilon \in (0, 1)$.
Output: the sets of a $(1+\varepsilon)$-approximation solution for Semi-Restricted SSR.

```
δ ← ε·ap / 3·n
A' ← ∅
for i ← 1 to n do
    ai' ← ⌊ai / δ⌋
    A' ← A' ∪ {ai'}
end for
(S1, S2) ← SOLex(A', p)
return S1, S2
```

Proof of correctness In this section, we will present the proof that this approximation scheme, actually approximates the optimal solution by a factor $(1+\varepsilon)$.

Let S_A, S_B be the pair that the Algorithm 3 returns, on input $A = \{a_1, ..., a_n\}, p$ and ε and also (S_1^*, S_2^*) the optimal solution of the instance.

Lemma 4.1. *For any* $S \in \{S_A, S_B, S_1^*, S_2^*\}$

$$\sum_{i \in S} a_i - n \cdot \delta \leq \sum_{i \in S} \delta \cdot a_i' \leq \sum_{i \in S} a_i \tag{4.1}$$

$$n * \delta \leq \frac{\varepsilon}{3} \sum_{i \in S} a_i \tag{4.2}$$

Proof. For 4.1 notice that for all $i \in \{1, ..., n\}$ we define $a_i' = \frac{a_i}{\delta}$. This gives us

$$\frac{a_i}{\delta} - 1 \leq a_i' \leq \frac{a_i}{\delta} \implies a_i - \delta \leq \delta \cdot a_i \leq a_i$$

In addition, for any $S \in \{S_A, S_B, S_1^*, S_2^*\}$ we have $|S| \leq n$, which means that

$$\sum_{i \in S} a_i - n \cdot \delta \leq \sum_{i \in S} \delta \cdot a_i' \leq \sum_{i \in S} a_i$$

For 4.2 observe that $maxS \geq p$ for any $S \in \{SA, SB, S_1^*, S_2^*\}$. By this observation, we can show the second inequality

$$n \cdot \delta \leq n \cdot \frac{\varepsilon \cdot a_p}{3 \cdot n} \leq \frac{\varepsilon}{3} \cdot \sum_{i \in S} a_i$$

□

Lemma 4.2. $\mathcal{MR}(S_A, S_B, A) \leq \mathcal{MR}(S_A, S_B, A') + \frac{\varepsilon}{3}$

Proof.

$$\begin{aligned}
\mathcal{R}(S_A, S_B, A) &\leq \frac{\sum_{i \in S_A} a_i}{\sum_{j \in S_B} a_j} \leq \frac{\sum_{i \in S_A} \delta \cdot a_i' + \delta \cdot n}{\sum_{i \in S_B} a_j} \\
&\leq \frac{\sum_{i \in S_A} a_i'}{\sum_{i \in S_B} a_j'} + \frac{\delta \cdot n}{\sum_{i \in S_B} a_j} \\
&\leq \mathcal{MR}(S_A, S_B, A') + \frac{\varepsilon}{3}
\end{aligned}$$

The same way we have

$$\mathcal{R}(S_A, S_B, A) \leq \mathcal{MR}(S_A, S_B, A') + \frac{\varepsilon}{3}$$

Thus we completed the proof. □

Lemma 4.3. *For any* $\varepsilon \in (0, 1), \mathcal{MR}(S_1^*, S_2^*, A') \leq (1 + \frac{\varepsilon}{2}) \cdot \mathcal{MR}(S_1^*, S_2^*, A)$.

Proof. If $\mathcal{R}(S_1^*, S_2^*, A') \geq 1$, let $(S_1, S_2) = (S_1^*, S_2^*)$, otherwise $(S_1, S_2) = (S_2^*, S_1^*)$. Then it holds that

$$\begin{aligned}
\mathcal{MR}(S_1^*, S_2^*, A) = \mathcal{R}(S_1, S_2, A) &= \frac{\sum_{i \in S_1} a_i'}{\sum_{j \in S_2} a_j'} \\
&\leq \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j - n \cdot \delta} \\
&= \frac{\sum_{i \in S_2} a_i}{\sum_{j \in S_2} a_j - n \cdot \delta} \cdot \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}
\end{aligned}$$

$$= (1 + \frac{n \cdot \delta}{\sum_{j \in S_2} a_j - n \cdot \delta} \cdot \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$$

Because $S_2 \in \{S_1^*, S_2^*\}$, by 4.2 it follows that

$$\mathcal{MR}(S_1^*, S_2^*, A') \leq (1 + \frac{1}{\frac{3}{\varepsilon} - 1}) \cdot \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$$

$$= (1 + \frac{\varepsilon}{3 - \varepsilon}) \cdot \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$$

$$\leq (1 + \frac{\varepsilon}{2}) \cdot \frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$$

$$\leq (1 + \frac{\varepsilon}{2}) \cdot \mathcal{MR}(S_1^*, S_2^*, A)$$

This concludes the proof. □

In the following, the researchers state the next theorem, which proves that Algorithm 3 is an $(1 + \varepsilon)-$approximation algorithm.

Theorem 4.4. *Let S_A, S_B be the pair of sets returned by Algorithm 3 on input $(A = \{a_1, ..., a_n\}, p, \varepsilon)$ and S_1^*, S_2^* be an optimal solution, then:*

$$\mathcal{MR}(S_A, S_B, A) \leq (1 + \varepsilon) \cdot \mathcal{MR}(S_1^*, S_2^*, A).$$

Proof.

$$\mathcal{MR}(S_B, S_A, A) \leq \mathcal{MR}(S_A, S_B, A') + \frac{\varepsilon}{3}$$

$$\leq \mathcal{MR}(S_1^*, S_2^*, A') + \frac{\varepsilon}{3}$$

$$\leq (1 + \frac{\varepsilon}{2}) \cdot \mathcal{MR}(S_1^*, S_2^*, A) + \frac{\varepsilon}{3}$$

$$\leq (1 + \varepsilon) \cdot \mathcal{MR}(S_1^*, S_2^*, A)$$

□

Finally, in order to show that the constructed algorithm is an FPTAS it must hold that the complexity of Algorithm 3 is $O(poly(n, 1/\varepsilon))$. As we described above the complexity of this algorithm is $O(n \cdot Q)$ where $Q = \sum_{i=1}^{p} a_i'$. Thus, by the definition of the scaled items a_i' it follows,

$$Q = \sum_{i=1}^{p} a_i' \leq n \cdot a_p' \leq \frac{n \cdot a_p}{\delta} = \frac{3 \cdot n^2}{\varepsilon}$$

which means that the complexity of Algorithm 3 is $O(n^3/\varepsilon)$.

In conclusion, it is sufficient to perform $n - 1$ executions of the Algorithm i.e. solving the Semi-Restricted SSR problem and return the the best of the solutions. Therefore, the result is summarized in the next theorem.

Theorem 4.5. *The above described algorithm is an FPTAS for SSR that runs in $O(n^4/\varepsilon)$time.*

4.0.2 Establishing a connection between SSR and Subset Sum and formulate an FPTAS of time $\tilde{O}(min\{\frac{n^{2.3}}{\varepsilon^{2.6}}, \frac{n^2}{\varepsilon^3}, \frac{n^2}{\varepsilon^2}\})$

In this section, We present a new FPTAS for the Subset Sum Ratio problem, which, given a set of integers, asks for two disjoint subsets such that the ratio of their sums is as close to 1 as possible. Our scheme makes use of exact and approximate algorithms for the closely related Subset Sum problem, hence any progress over those—such as the recent improvement due to Bringmann and Nakos [SODA 2021]—carries over to our FPTAS. Depending on the relationship between the size of the input set n and the error margin ε, we improve upon the best currently known algorithm of Melissinos and Pagourtzis [COCOON 2018] of complexity $O(n^4/\varepsilon)$. In particular, the exponent of n in our proposed scheme may decrease down to 2, depending on the Subset Sum algorithm used. Furthermore, while the aforementioned state of the art complexity, expressed in the form $O((n + 1/\varepsilon)^c)$, has constant $c = 5$, our results establish that $c < 5$.
In particular, a restricted version of Subset Sum Ratio is being solved. We will present a fully polynomial time approximation scheme, which successfully approximates those solutions that involve large subset sums. Specifically, consider $A = \{a_1, \ldots, a_n\}$ the *sorted* input set and a given error margin $\varepsilon \in (0, 1)$. Then, the FPTAS presented in this section will consider the problem of finding two disjoint subsets of minimum ratio of subset sums.

Outline of the algorithm

We now present a rough outline of the presented algorithm, along with its respective pseudocode:

- At first, we search for approximate solutions involving exclusively large elements from $L(A, \varepsilon)$.

- To this end, we produce all the subset sums formed by these large elements. If their number exceeds n/ε^2, then we can easily find an approximate solution.

- Otherwise, for each of the produced subsets, we find its corresponding ε'-close set, for some appropriate ε' defined later.

- We prove that it suffices to consider only these pairs of subsets when searching for an approximate solution, irrespective of whether the solution contains small elements. In the case that it indeed does, we can efficiently add small elements to the ε'-close pairs in a greedy way.

- This procedure successfully approximates the optimal solution for Subset Sum Ratio, with the additional constraint that at least one of the solution subsets has sum of its large elements at least a_n.

Algorithm 1 ConstrainedSSR(A, ε, T)

Input : Sorted set $A = \{a_1, \ldots, a_n\}$, error margin ε and table of partial sums T.
Output : $(1+\varepsilon)$-approximation of the optimal solution respecting the constraint.

```
Partition A to M = {a_i ∈ A | a_i < ε · a_n} and L = {a_i ∈ A | a_i ≥ ε · a_n}.
Split interval [0, n · a_n] to n/ε^2 bins of size ε^2 · a_n.
while filling the bins with the subset sums of L do
    if two subset sums correspond to the same bin then
        return an approximation solution based on these.   ▷ O(n/ε^2) complexity.
    end if
end while
2^|L| ≤ n/ε^2  ⟺  |L| ≤ log(n/ε^2).
for each subset in a bin do                                  ▷ O(n/ε^2) subsets.
    Find its ε'-close set.                                   ▷ Complexity analysis in Section 5.
    Add small elements accordingly.                          ▷ O(log n) complexity, see Subsection 3.3.
end for
```

4.0.3 Regarding only large elements

We firstly search for an $(1+\varepsilon)$-approximate solution with $\varepsilon \in (0,1)$, without involving any of the elements that are smaller than $\varepsilon \cdot a_n$. Let $M = \{a_i \in A | a_i < \varepsilon \cdot a_n\}$ be the set of small elements and $L = A \setminus M = \{a_i \in A | a_i \geq \varepsilon \cdot a_n\}$ be the set of large elements.

After partitioning the input set, we split the interval $[0, n \cdot a_n]$ into smaller intervals, called bins, of size $l = \varepsilon^2 \cdot a_n$ each, as depicted in figure 4.1.

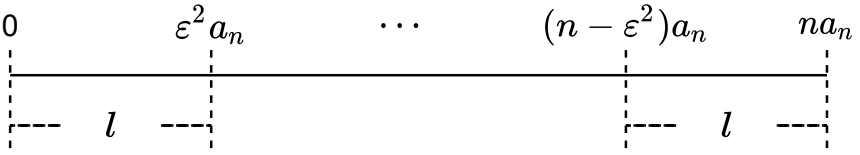

Figure 4.1: Split of the interval $0, n \cdot a_n$ to bins of size l.

Thus, there are a total of $B = n/\varepsilon^2$ bins. Notice that each possible subset of the input set will belong to a respective bin constructed this way, depending on its sum. Additionally, if two sets correspond to the same bin, then the difference of their subset sums will be at most l.

The next step of our algorithm is to generate all the possible subset sums, occurring from the set of large elements L. The complexity of this procedure is $O(2^{|L|})$, where $|L|$ is the cardinality of set L. Notice however, that it is possible to bound the number of the produced subset sums by the number of bins B, since if two sums belong to the same bin they constitute a solution, as shown in Lemma 4.6, in which case the algorithm terminates in time $O(n/\varepsilon^2)$.

Lemma 4.6. *If two subsets correspond to the same bin, we can find an* $(1+\varepsilon)$*-approximation solution.*

Proof. Suppose there exist two sets $L_1, L_2 \subseteq L$ whose sums correspond to the same bin, with $\Sigma(L_1) \leq \Sigma(L_2)$. Notice that there is no guarantee regarding the disjointness of said subsets, thus consider $L_1' = L_1 \setminus L_2$ and $L_2' = L_2 \setminus L_1$, for which it is obvious that $\Sigma(L_1') \leq \Sigma(L_2')$.

Additionally, assume that $L_1' \neq \emptyset$. Then it holds that

$$\Sigma(L_2') - \Sigma(L_1') = \Sigma(L_2) - \Sigma(L_1) \leq l$$

Therefore, the sets L'_1 and L'_2 constitute an $(1+\varepsilon)$-approximation solution, since

$$\frac{\Sigma(L'_2)}{\Sigma(L'_1)} \leq \frac{\Sigma(L'_1)+l}{\Sigma(L'_1)} = 1 + \frac{l}{\Sigma(L'_1)}$$
$$\leq 1 + \frac{\varepsilon^2 \cdot a_n}{\varepsilon \cdot a_n} = 1+\varepsilon$$

where the last inequality is due to the fact that $L'_1 \subseteq L$ is composed of elements $\geq \varepsilon \cdot a_n$, thus $\Sigma(L'_1) \geq \varepsilon \cdot a_n$.

It remains to show that $L'_1 \neq \emptyset$. Assume that $L'_1 = \emptyset$. This implies that $L_1 \subseteq L_2$ and since we consider each subset of L only once and the input is a set and not a multiset, it holds that $L_1 \subset L_2 \implies L'_2 \neq \emptyset$. Since L_1 and L_2 correspond to the same bin, it holds that

$$\Sigma(L_2) - \Sigma(L_1) \leq l \implies \Sigma(L'_2) - \Sigma(L'_1) \leq l \implies \Sigma(L'_2) \leq l$$

which is a contradiction, since L'_2 is a non empty subset of L, which is comprised of elements greater than or equal to $\varepsilon \cdot a_n$, hence $\Sigma(L'_2) \geq \varepsilon \cdot a_n > \varepsilon^2 \cdot a_n = l$, since $\varepsilon < 1$. □

Consider an ε' such that $1/(1-\varepsilon') \leq 1+\varepsilon$ for all $\varepsilon \in (0,1)$, for instance $\varepsilon' = \varepsilon/2$ (the exact value of ε' will be computed in Section 4.2). If every produced subset sum of the previous step belongs to a distinct bin, then, we compute their respective ε'-close sets, as described in Section **??**. We can approximate an optimal solution that involves exclusively large elements using these pairs.

Before we prove the previous statement, observe that, if the optimal solution involves sets $L_1, L_2 \subseteq L$ composed only of large elements, where $\Sigma(L_1) \leq \Sigma(L_2)$, then $\Sigma(L_1) = \Sigma(L_{2,})$, where $L_{2,}$ is a closest set of L_2, with respect to the set $L \setminus L_2$.

Lemma 4.7. *If the optimal ratio r involves sets consisting of only large elements, then there exists an ε'-close pair with ratio $r \leq (1+\varepsilon)\cdot r$.*

Proof. Assume that the sets $S_1^*, S_2^* \subseteq L$ form the optimal solution (S_2^*, S_1^*) and $\frac{\Sigma(S_2^*)}{\Sigma(S_1^*)} = r \geq 1$ is the optimal ratio. Then, as mentioned, it holds that $\Sigma(S_1^*) = \Sigma(S^*_{2,opt})$. For each set of large elements, there exists an ε'-close set and a corresponding ε'-close pair; let $(S_2^*, S^*_{2,\varepsilon'})$ be this pair for set S_2^*. Then,

$$\Sigma(S_2^*) \geq \Sigma(S_1^*) = \Sigma(S^*_{2,}) \geq \Sigma(S^*_{2,\varepsilon'}) \geq (1-\varepsilon')\cdot\Sigma(S_1^*)$$

Thus, it holds that

$$1 \leq \frac{\Sigma(S_2^*)}{\Sigma(S^*_{2,\varepsilon'})} \leq \frac{1}{(1-\varepsilon')} \cdot \frac{\Sigma(S_2^*)}{\Sigma(S_1^*)} \leq (1+\varepsilon)\cdot r$$

□

Therefore, we have proved that in the case where the optimal solution consists of sets comprised of only large elements, it is possible to find an $(1+\varepsilon)$-approximation solution. This is achieved by computing an ε'-close set for each subset $L_i \subseteq L$ belonging in some bin, using the algorithms described in the preliminaries, with respect to set $L \setminus L_i$ and target $\Sigma(L_i)$. The total cost of these algorithms will be thoroughly analyzed in Section 4.2 and depends on the algorithm used.

It is important to note that by utilizing an (exact or approximation) algorithm for Subset Sum, we establish a connection between the complexities of Subset Sum and approximating Subset Sum Ratio in a way that any future improvement in the first carries over to the second.

4.0.4 General $(1+\varepsilon)$-approximation solutions

Whereas we previously considered optimal solutions involving exclusively large elements, here we will search for approximations for those optimal solutions that use all the elements of the input set, hence include small elements, and satisfy our constraint. We will prove that in order to approximate those optimal solutions, it suffices to consider only the ε'-close pairs corresponding to each distinct bin and add small elements to them. In other words, instead of considering any two random disjoint subsets consisting of large elements[1] and subsequently adding to these the small elements, we can instead consider only the pairs computed in the previous step, the number of which is bounded by the number of bins $B = n/\varepsilon^2$. Moreover, we will prove that it suffices to add the small elements to our solution in a greedy way.

Since the algorithm has not detected a solution so far, due to Lemma 4.6 every computed subset sum of set L belongs to a different bin. Thus, their total number is bounded by the number of bins B, i.e.

$$2^{|L|} \leq (\frac{n}{\varepsilon^2}) \iff |L| \leq \log(\frac{n}{\varepsilon^2}).$$

We proceed by involving small elements in order to reduce the difference between the sums of ε'-close pairs, thus reducing their ratio.

Lemma 4.8. *Given the ε'-close pairs, one can find an $(1+\varepsilon)$-approximation solution for the constrained version of* Subset Sum Ratio*, in the case that the optimal solution involves small elements.*

sketch. Due to page limitations, we only give a short sketch of the proof here; the full proof is deferred to the appendix.

Let $S_1^* = L_1^* \cup M_1^*$ and $S_2^* = L_2^* \cup M_2^*$ be disjoint subsets that form an optimal solution, where $\Sigma(S_1^*) \leq \Sigma(S_2^*)$, $L_1^*, L_2^* \subseteq L$ and $M_1^*, M_2^* \subseteq M$.

For $\Sigma(L_1^*) < \Sigma(L_2^*)$ (respectively $\Sigma(L_2^*) < \Sigma(L_1^*)$), we show that is suffices to add an appropriate subset $M_k \subseteq M$ to $L_{2,\varepsilon'}^*$ (respectively $L_{1,\varepsilon'}^*$) in order to approximate the optimal solution $r = \frac{\Sigma(S_2^*)}{\Sigma(S_1^*)}$, where $M_k = \{a_i \in M | i \in [k]\}$ and $k \leq |M|$.

Therefore, by adding in a greedy way small elements to an ε'-close set of the set with the largest sum among L_1^* and L_2^*, we can successfully approximate the optimal solution. □

Adding small elements efficiently.

Here, we will describe a method to efficiently add small elements to our sets. As a reminder, up to this point the algorithm has detected an ε'-close pair (L_2, L_1), such that $L_1, L_2 \subseteq L$ with $\Sigma(L_1) < \Sigma(L_2)$. Thus, we search for some k such that $\Sigma(L_1 \cup M_k) \leq \Sigma(L_2) + \varepsilon \cdot a_n$, where $M_k = a_i \in M i \in [k]$. Notice that if $\Sigma(M) \geq \Sigma(L_2) - \Sigma(L_1)$, there always exists such a set M_k, since by definition, each element of set M is smaller than $\varepsilon \cdot a_n$. In order to determine M_k, we make use of an array of partial sums $T[k] = \Sigma(M_k)$, where $k \leq |M|$. Notice that T is sorted; therefore, since T is already available (see Final Algorithm), each time we need to compute a subset with the desired property this can be done in $O(\log k) = O(\log n)$ time.

4.1 Final Algorithm

The algorithm presented in the previous section constitutes an approximation scheme for Subset Sum Ratio, in the case where at least one of the solution subsets has sum of its large elements greater than, or equal to the max element of the input set. Thus, in order

[1] Note that the number of these random pairs is $3^{|L|}$.

to solve the Subset Sum Ratio problem, it suffices to run the previous algorithm n times, where n depicts the cardinality of the input set A, while each time removing the max element of A.

In particular, suppose that the optimal solution involves disjoint sets S_1^* and S_2^*, where $a_k = \max\{S_1^* \cup S_2^*\}$. There exists an iteration for which the algorithm considers as input the set $A_k = \{a_1, \ldots, a_k\}$. In this iteration, the element a_k is the largest element and the algorithm searches for a solution where the sum of the large elements of one of the two subsets is at least a_k. The optimal solution has this property so the ratio of the approximate solution that the algorithm of the previous section returns is at most $(1+\varepsilon)$ times the optimal.

Consequently, n repetitions of the algorithm suffice to construct an FPTAS for Subset Sum Ratio.

Notice that if at some repetition, the sets returned due to the algorithm of Section **??** have ratio at most $1+\varepsilon$, then this ratio successfully approximates the optimal ratio $r \geq 1$, since $1+\varepsilon \leq (1+\varepsilon)\cdot r$, therefore they constitute an approximation solution.

Algorithm 2 SSR(A, ε)

Input : Sorted set $A = \{a_1, \ldots, a_n\}$ and error margin ε.
Output : $(1+\varepsilon)$-approximation of the optimal solution for Subset Sum Ratio.
1: Consider table T such that $T[k] = \sum_{i=1}^{k} a_i$. ▷ $\Theta(n)$ time.
2: **for** $i = n, \ldots, 1$ **do**
3: ConstrainedSSR$(\{a_1, \ldots, a_i\}, \varepsilon, T)$
4: **end for**

4.2 Complexity

The total complexity of the final algorithm is determined by three distinct operations, over the n iterations of the algorithm:

1. The cost to compute all the possible subset sums occurring from large elements. It suffices to consider the case where this is bounded by the number of bins $B = n/\varepsilon^2$, due to Lemma 4.6.

2. The cost to find the ε'-close pair for each subset in a distinct bin. The cost of this operation will be analyzed in the following subsection.

3. The cost to include small elements to the ε'-close pairs. There are B ε'-close pairs, and each requires $O(\log n)$ time, thus the total time required is $O(\frac{n}{\varepsilon^2}\log n)$.

4.2.1 Complexity to find the ε'-close pairs

Using exact Subset Sum computations.

The first algorithm we mentioned is a standard meet in the middle algorithm. Here we will analyze its complexity.

Let subset $L' \subseteq L$ such that $|L'| = k$. The meet in the middle algorithm on the set $L \setminus L'$ costs time

$$O\left(\frac{|L \setminus L'|}{2} \cdot 2^{\frac{|L \setminus L'|}{2}}\right).$$

Notice that the number of subsets of L of cardinality k is $\binom{|L|}{k}$ and that $|L| \leq \log(n/\varepsilon^2)$.

Additionally,

$$\sum_{k=0}^{|L|}\binom{|L|}{k}\cdot 2^{\frac{|L|-k}{2}}\cdot\frac{|L|-k}{2}=2^{|L|/2}\cdot\sum_{k=0}^{|L|}\binom{|L|}{k}\cdot 2^{-k/2}\cdot\frac{|L|-k}{2}$$
$$\leq 2^{|L|/2}\cdot\frac{|L|}{2}\cdot\sum_{k=0}^{|L|}\binom{|L|}{k}\cdot 2^{-k/2}$$

Furthermore, let $c=(1+2^{-1/2})$, where $\log c=0.7715...<0.8$. Due to Binomial Theorem, it holds that

$$\sum_{k=0}^{|L|}\binom{|L|}{k}\cdot 2^{-k/2}=(1+2^{-1/2})^{|L|}=c^{|L|}\leq c^{\log(n/\varepsilon^2)}=(n/\varepsilon^2)^{\log c}$$

Consequently, the complexity to find a closest set for every subset in a bin is

$$O(2^{|L|/2}\cdot\frac{|L|}{2}\cdot(n/\varepsilon^2)^{\log c})=O((n/\varepsilon^2)^{1/2}\cdot\log(n/\varepsilon^2)\cdot(n/\varepsilon^2)^{\log c})$$
$$=O(\frac{n^{1.3}}{\varepsilon^{2.6}}\cdot\log(n/\varepsilon^2))$$

Using approximate Subset Sum computations.

Here we will analyze the complexity in the case we run an approximate Subset Sum algorithm.

For subset $L_i\subseteq L$ of sum $\Sigma(L_i)$, we run an approximate Subset Sum algorithm ([Kell03, Brin21]), with error margin ε' such that

$$\frac{1}{1-\varepsilon'}\leq 1+\varepsilon\iff\varepsilon'\leq\frac{\varepsilon}{1+\varepsilon}$$

By choosing the maximum such ε', we have that

$$\varepsilon'=\frac{\varepsilon}{1+\varepsilon}\implies\frac{1}{\varepsilon'}=O(\frac{1}{\varepsilon})$$

Thus, if we use for instance the approximation algorithm[2] presented at [Kell03], the complexity to find an ε'-close set for every subset in a bin is

$$O(\frac{n}{\varepsilon^2}\cdot\min\{\frac{|L|}{\varepsilon'},|L|+\frac{1}{(\varepsilon')^2}\cdot\log(1/\varepsilon')\})=$$
$$O(\frac{n}{\varepsilon^2}\cdot\min\{\frac{|L|}{\varepsilon},|L|+\frac{1}{\varepsilon^2}\cdot\log(1/\varepsilon)\})=$$
$$O(\frac{n}{\varepsilon^2}\cdot\min\{\frac{\log(n/\varepsilon^2)}{\varepsilon},\log(n/\varepsilon^2)+\frac{1}{\varepsilon^2}\cdot\log(1/\varepsilon)\})$$

4.2.2 Total complexity

The total complexity of the algorithm occurs from the n distinct iterations required and depends on the algorithm chosen to find the ε'-close pairs, since both of the presented algorithms dominate the time of the rest of the operations. Thus, by choosing the fastest one (depending on the relationship between n and ε), the final complexity is

$$O(\min\{\frac{n^{2.3}}{\varepsilon^{2.6}}\cdot\log(n/\varepsilon^2),\frac{n^2}{\varepsilon^3}\cdot\log(n/\varepsilon^2),\frac{n^2}{\varepsilon^2}(\log(n/\varepsilon^2)+\frac{1}{\varepsilon^2}\cdot\log(1/\varepsilon))\})$$

[2] Of complexity $O(\min\{\frac{n}{\varepsilon},n+\frac{1}{\varepsilon^2}\cdot\log(1/\varepsilon)\})$ for n elements and error margin ε.

Chapter 5

Conclusions and future research directions

We conclude that there is an increasing interest in approximation algorithms for knapsack type problems (fair allocation of goods) and also we manage to produce an FPTAS that has two main benefits. The first is that it is faster than all of the previously presented approximation schemes for Subset Sum Ratio and also if we express the time complexity in the form $O((n+1/\varepsilon)^c)$ where $c > 0$, we show that Subset Sum Ratio can be approximated with $c < 5$ whereas the previous faster FPTAS achieved $c = 5$. Secondly, we establish a connection between Subset Sum Ratio and approximating Subset Sum which leads to the following: Any improvement in the complexity of approximating Subset Sum can be immediately transferred to our scheme, thus connecting the complexities of the two problems, even the fact that they are very close there was no similar result up until now.

Furthermore, we suggest that a possible future research direction could be to utilize faster exact algorithms such as the ones presented in the papers [Aust15] and [Aust16] where the researchers used a concentration parameter β and they solved the decision version of Subset Sum thus it is interesting to check whether analogous setting and arguments can solve the optimization version faster.

Additionally, another interesting future research direction is to study the Weak Approximation Version of Subset Sum as stated in [Brin21] which seems to provide amazing improvement for the closely related PARTITION problem. This implies, that it may provide further improvements to approximating Subset Sum and Subset Sum Ratio by relaxing the approximation constraints and utilize it in order to compute faster the $\varepsilon-$close sets.

Finally, there is also another application on fair allocation of goods in people's network where their utility function for the goods is additive and also they have the same value for each good. In this case, we improve the complexity for the minimization of envy-ratio allocation problem.

All of the above, show that the improvements suggested in this dissertation are important and also pinpoint the need for further research on the Subset Sum and Subset Sum Ratio problems.

Bibliography

[Alon95] Noga Alon, Raphael Yuster and Uri Zwick, "Color-Coding", *J. ACM*, vol. 42, no. 4, pp. 844–856, 1995.

[Auma11] Yonatan Aumann, Moshe Lewenstein, Noa Lewenstein and Dekel Tsur, "Finding witnesses by peeling", *ACM Trans. Algorithms*, vol. 7, no. 2, pp. 24:1–24:15, 2011.

[Aust15] Per Austrin, Petteri Kaski, Mikko Koivisto and Jesper Nederlof, "Subset Sum in the Absence of Concentration", in *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, vol. 30 of *LIPIcs*, pp. 48–61, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[Aust16] Per Austrin, Petteri Kaski, Mikko Koivisto and Jesper Nederlof, "Dense Subset Sum May Be the Hardest", in *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, vol. 47 of *LIPIcs*, pp. 13:1–13:14, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[Baka20] Eleni Bakali, Aggeliki Chalki and Aris Pagourtzis, "Characterizations and Approximability of Hard Counting Classes Below #P", in Jianer Chen, Qilong Feng and Jinhui Xu, editors, *Theory and Applications of Models of Computation, 16th International Conference, TAMC 2020, Changsha, China, October 18-20, 2020, Proceedings*, vol. 12337 of *Lecture Notes in Computer Science*, pp. 251–262, Springer, 2020.

[Bazg02] Cristina Bazgan, Miklos Santha and Zsolt Tuza, "Efficient Approximation Algorithms for the SUBSET-SUMS EQUALITY Problem", *J. Comput. Syst. Sci.*, vol. 64, no. 2, pp. 160–170, 2002.

[Bell57] Richard E. Bellman, *Dynamic programming*, Princeton University Press, 1957.

[Blah85] Richard E. Blahut, *Fast Algorithms for Digital Signal Processing*, Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1985.

[Bren76] Richard P. Brent, "Multiple-precision zero-finding methods and the complexity of elementary function evaluation", in J.F. Traub, editor, *Analytic Computational Complexity*, pp. 151–176, Academic Press, 1976.

[Brin17] Karl Bringmann, "A Near-Linear Pseudopolynomial Time Algorithm for Subset Sum", in Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pp. 1073–1084, SIAM, 2017.

[Brin20] Karl Bringmann and Vasileios Nakos, "Top-k-convolution and the quest for near-linear output-sensitive subset sum", in Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pp. 982–995, ACM, 2020.

[Brin21] Karl Bringmann and Vasileios Nakos, "A Fine-Grained Perspective on Approximating Subset Sum and Partition", in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pp. 1797–1815, SIAM, 2021.

[Capr00] Alberto Caprara, Hans Kellerer and Ulrich Pferschy, "A PTAS for the Multiple Subset Sum Problem with different knapsack capacities", *Inf. Process. Lett.*, vol. 73, no. 3-4, pp. 111–118, 2000.

[Capr03] Alberto Caprara, Hans Kellerer and Ulrich Pferschy, "A 3/4-Approximation Algorithm for Multiple Subset Sum", *J. Heuristics*, vol. 9, no. 2, pp. 99–111, 2003.

[Ciel03a] Mark Cieliebak, Stephan J. Eidenbenz and Aris Pagourtzis, "Composing Equipotent Teams", in Andrzej Lingas and Bengt J. Nilsson, editors, *Fundamentals of Computation Theory, 14th International Symposium, FCT 2003, Malmö, Sweden, August 12-15, 2003, Proceedings*, vol. 2751 of *Lecture Notes in Computer Science*, pp. 98–108, Springer, 2003.

[Ciel03b] Mark Cieliebak, Stephan J. Eidenbenz and Paolo Penna, "Noisy Data Make the Partial Digest Problem NP-hard", in Gary Benson and Roderic D. M. Page, editors, *Algorithms in Bioinformatics, Third International Workshop, WABI 2003, Budapest, Hungary, September 15-20, 2003, Proceedings*, vol. 2812 of *Lecture Notes in Computer Science*, pp. 111–123, Springer, 2003.

[Ciel04] Mark Cieliebak and Stephan J. Eidenbenz, "Measurement Errors Make the Partial Digest Problem NP-Hard", in Martin Farach-Colton, editor, *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, vol. 2976 of *Lecture Notes in Computer Science*, pp. 379–390, Springer, 2004.

[Ciel08] Mark Cieliebak, Stephan J. Eidenbenz, Aris Pagourtzis and Konrad Schlude, "On the Complexity of Variations of Equal Sum Subsets", *Nord. J. Comput.*, vol. 14, no. 3, pp. 151–172, 2008.

[Corm09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, *Introduction to Algorithms, 3rd Edition*, MIT Press, 2009.

[Dell19] Mauro Dell'Amico, Maxence Delorme, Manuel Iori and Silvano Martello, "Mathematical models and decomposition methods for the multiple knapsack problem", *Eur. J. Oper. Res.*, vol. 274, no. 3, pp. 886–899, 2019.

[Dutt21] Pranjal Dutta and Mahesh Sreekumar Rajasree, "Efficient reductions and algorithms for variants of Subset Sum", *CoRR*, vol. abs/2112.11020, 2021.

[Gawr18] Pawel Gawrychowski, Liran Markin and Oren Weimann, "A Faster FPTAS for #Knapsack", in Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, vol. 107 of *LIPIcs*, pp. 64:1–64:13, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Gens94] Levner Gens, "A Fast Approximation Algorithm For The Subset-Sum Problem", *Information Systmes and Operational Research*, vol. 32, pp. 143–148, 9 1994.

[Gopa11] Parikshit Gopalan, Adam R. Klivans, Raghu Meka, Daniel Stefankovic, Santosh S. Vempala and Eric Vigoda, "An FPTAS for #Knapsack and Related Counting

Problems", in Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pp. 817–826, IEEE Computer Society, 2011.

[Horo74] Ellis Horowitz and Sartaj Sahni, "Computing Partitions with Applications to the Knapsack Problem", *J. ACM*, vol. 21, no. 2, pp. 277–292, 1974.

[Ibar75] Kim Ibarra, "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems", *Journal of the ACM*, vol. 22, pp. 463–468, 10 1975.

[Jin19] Ce Jin and Hongxun Wu, "A Simple Near-Linear Pseudopolynomial Time Randomized Algorithm for Subset Sum", in Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, vol. 69 of *OASICS*, pp. 17:1–17:6, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Jin21] Ce Jin, Nikhil Vyas and Ryan Williams, "Fast Low-Space Algorithms for Subset Sum", in Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pp. 1757–1776, SIAM, 2021.

[Karp72] Richard M. Karp, *Reducibility among Combinatorial Problems*, pp. 85–103, Springer US, Boston, MA, 1972.

[Kell03] Hans Kellerer, Renata Mansini, Ulrich Pferschy and Maria Grazia Speranza, "An efficient fully polynomial approximation scheme for the Subset-Sum Problem", *J. Comput. Syst. Sci.*, vol. 66, no. 2, pp. 349–370, 2003.

[Khan17] Muhammad Ali Khan, "Some Problems on Graphs and Arrangements of Convex Bodies", 2017.

[Koil19] Konstantinos Koiliaris and Chao Xu, "Faster Pseudopolynomial Time Algorithms for Subset Sum", *ACM Trans. Algorithms*, vol. 15, no. 3, pp. 40:1–40:20, 2019.

[Lahy19] Rahma Lahyani, Khalil Chebil, Mahdi Khemakhem and Leandro C. Coelho, "Matheuristics for solving the Multiple Knapsack Problem with Setup", *Comput. Ind. Eng.*, vol. 129, pp. 76–89, 2019.

[Lawl79] Lawler, "Fast Approximation Algorithms for Knapsack Problems", *Mathematics of Operations Research*, vol. 4, pp. 303–477, 11 1979.

[Lipt04] Richard J. Lipton, Evangelos Markakis, Elchanan Mossel and Amin Saberi, "On approximately fair allocations of indivisible goods", in Jack S. Breese, Joan Feigenbaum and Margo I. Seltzer, editors, *Proceedings 5th ACM Conference on Electronic Commerce (EC-2004), New York, NY, USA, May 17-20, 2004*, pp. 125–131, ACM, 2004.

[Loks10] Daniel Lokshtanov and Jesper Nederlof, "Saving space by algebraization", in Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pp. 321–330, ACM, 2010.

[Meli18] Nikolaos Melissinos and Aris Pagourtzis, "A Faster FPTAS for the Subset-Sums Ratio Problem", in Lusheng Wang and Daming Zhu, editors, *Computing and Combinatorics - 24th International Conference, COCOON 2018, Qing Dao, China, July 2-4, 2018, Proceedings*, vol. 10976 of *Lecture Notes in Computer Science*, pp. 602–614, Springer, 2018.

[Meli20] Nikolaos Melissinos, Aris Pagourtzis and Theofilos Triommatis, "Approximation Schemes for Subset Sum Ratio Problems", in Minming Li, editor, *Frontiers in Algorithmics - 14th International Workshop, FAW 2020, Haikou, China, October 19-21, 2020, Proceedings*, vol. 12340 of *Lecture Notes in Computer Science*, pp. 96–107, Springer, 2020.

[Much19] Marcin Mucha, Jesper Nederlof, Jakub Pawlewicz and Karol Wegrzycki, "Equal-Subset-Sum Faster Than the Meet-in-the-Middle", in Michael A. Bender, Ola Svensson and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, vol. 144 of *LIPIcs*, pp. 73:1–73:16, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Nano13] Danupon Nanongkai, "Simple FPTAS for the subset-sums ratio problem", *Inf. Process. Lett.*, vol. 113, no. 19-21, pp. 750–753, 2013.

[Naor95] Moni Naor, Leonard J. Schulman and Aravind Srinivasan, "Splitters and Near-Optimal Derandomization", in *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pp. 182–191, IEEE Computer Society, 1995.

[Pago06] Aris Pagourtzis and Stathis Zachos, "The Complexity of Counting Functions with Easy Decision Version", in Rastislav Kralovic and Pawel Urzyczyn, editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, vol. 4162 of *Lecture Notes in Computer Science*, pp. 741–752, Springer, 2006.

[Papa94] Christos H. Papadimitriou, "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence", *J. Comput. Syst. Sci.*, vol. 48, no. 3, pp. 498–532, 1994.

[Pisi99] David Pisinger, "Linear Time Algorithms for Knapsack Problems with Bounded Weights", *J. Algorithms*, vol. 33, no. 1, pp. 1–14, 1999.

[Rizz19] Romeo Rizzi and Alexandru I. Tomescu, "Faster FPTASes for counting and random generation of Knapsack solutions", *Inf. Comput.*, vol. 267, pp. 135–144, 2019.

[Schm90] Jeanette P. Schmidt and Alan Siegel, "The Spatial Complexity of Oblivious k-Probe Hash Functions", *SIAM J. Comput.*, vol. 19, no. 5, pp. 775–786, 1990.

[Stef12] Daniel Stefankovic, Santosh S. Vempala and Eric Vigoda, "A Deterministic Polynomial-Time Approximation Scheme for Counting Knapsack Solutions", *SIAM J. Comput.*, vol. 41, no. 2, pp. 356–366, 2012.

[Trio20] Theofilos Triommatis and Aris Pagourtzis, "Approximate #Knapsack Computations to Count Semi-fair Allocations", in Jianer Chen, Qilong Feng and Jinhui Xu, editors, *Theory and Applications of Models of Computation, 16th International Conference, TAMC 2020, Changsha, China, October 18-20, 2020, Proceedings*, vol. 12337 of *Lecture Notes in Computer Science*, pp. 239–250, Springer, 2020.

[Volo17] Nadav Voloch, "MSSP for 2-D sets with unknown parameters and a cryptographic application", *Contemporary Engineering Sciences*, vol. 10, pp. 921–931, 10 2017.

[Woeg92] Gerhard J. Woeginger and Zhongliang Yu, "On the Equal-Subset-Sum Problem", *Inf. Process. Lett.*, vol. 42, no. 6, pp. 299–302, 1992.