Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΜΩΝ
Ζωγραφου 157 73, Αθηνα

Χρονομεταβαλλόμενη Βελτιστοποίηση σε Μητροειδή

Διπλωματικη Εργασια

του

Ορέστη Πλευράκη

Επιβλέπων: Δημήτρης Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΜΩΝ
Αθήνα, Ιούνιος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΜΩΝ
Ζωγράφου 157 73, Αθήνα

Χρονομεταβαλλόμενη Βελτιστοποίηση σε Μητροειδή

Διπλωματικη Εργασια

του

Ορέστη Πλευράκη

Επιβλέπων: Δημήτρης Φωτάκης
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την —.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
....	....	....
Δημήτρης Φωτάκης	Αριστείδης Παγουρτζής	Σταύρος Κολλιόπουλος
Επίκουρος Καθηγητής	Αναπληρωτής Καθηγητής	Καθηγητής
Ε.Μ.Π.	Ε.Μ.Π.	Ε.Κ.Π.Α.

Αθήνα, Ιούνιος 2017

(Υπογραφή)

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

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, τον κύριο Φωτάκη. Σαν καθηγητής, στα Διακριτά Μαθηματικά και στους Αλγορίθμους, μου έδειξε πλως μπορώ να κάνω αυτό που μου αρέσει μέσα από τη Θεωρητική Πληροφορική. Σαν επιβλέποντας της διπλωματικής μου εργασίας μου προσέφερε καθοδήγηση, ανεκτίμητες συμβουλές, στήριξη σε προσωπικό επίπεδο και με εισήγαγε στον κόσμο της έρευνας.

Επίσης, θα ήθελα να ευχαριστήσω τον κύριο Κολλιόπουλο, που μέσα απο τα μεταπτυχιακά του μαθήματα απέκτησα γνώσεις καθοριστικής σημασίας για την εκπόνηση αυτής της διπλωματικής εργασίας.

Οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου και στους φίλους μου που ήταν κοντά μου αυτά τα 6 υπέροχα προπτυχιακά χρονια. Η στήριξή τους είναι πραγματικά ανεκτίμητη.

Τέλος, θέλω να ευχαριστήσω την Ευαγγελία Καββαδά, στην οποία χρωστάω ένα πολύ μεγάλο μέρος της διαδρομής μου.

Περίληψη

Το πρόβλημα της Χρονομεταβαλλόμενης Βελτιστοποίησης σε Μητροειδή είναι μια γενίκευση του κλασικού προβλήματος εύρεσης βάσης ελαχίστου κόστους. Προτάθηκε από τους Gupta, Talwar και Wieder προκειμένου να μοντελοποιηθούν συστήματα τα οποία πρέπει να διατηρούνται συνεχώς, ενώ τα κόστη μεταβάλλονται με την πάροδο του χρόνου. Καθώς ο χρόνος περνάει, το κόστος των στοιχείων μπορεί να αλλάζει και αυτό οδηγεί σε ένα συμβιβασμό μεταξύ της διατήρησης μιας βάσης χαμηλού κόστους και της "σταθερότητας" της λύσης. Πιο συγκεκριμένα, η είσοδος είναι μια ακολουθία συναρτήσεων κόστους (μία για κάθε χρονική στιγμή). Ενώ αλλάζουμε τη βάση από βήμα σε βήμα, επιβάλλουμε ένα επιπλέον ενιαίο κόστος απόκτησης για κάθε τέτοια αλλαγή.

Στην παρούσα διπλωματική εργασία παρουσιάζουμε τον πρώτο ντετερμινιστικό rounding αλγόριθμο, που πετυχαίνει $O(\log r)$-προσέγγιση για το πρόβλημα, όπου r είναι ο βαθμός του Μητροειδούς, και που εγγυάται σταθερή προσέγγιση στο κόστος κράτησης. Οι Gupta et al. είχαν παρουσιάσει έναν τυχαιοκρατικό rounding αλγόριθμο που επίσης πετυχαίνει προσέγγιση $O(\log r)$, αλλά αυτό ισχύει τόσο για το κόστος κράτησης όσο και για το κόστος απόκτησης. Ο αλγόριθμος μας βασίζεται στην καλά κατανοητή δομή του πολυτόπου των ανεξαρτήτων συνόλων και εισάγει μια νέα rounding τεχνική που ενδέχεται να είναι ανεξαρτήτου ενδιαφέροντος. Επιπλέον, παρέχουμε τις πρώτες αποδείξεις για integrality γραμμικών προγραμμάτων για προβλήματα Χρονομεταβαλλόμενης Συνδυαστικής Βελτιστοποίησης. Συγκεκριμένα, δείχνουμε ότι το ΓΠ για τη Χρονομεταβαλλόμενη Βελτιστοποίηση σε Μητροειδή Διαμέρισης είναι ακέραιο. Δείχνουμε, επίσης, οτι το ΓΠ για τη Χρονομεταβαλλόμενη Βελτιστοποίηση σε Μητροειδή, για δύο χρονικές στιγμές, ακόμα και αν τα δύο Μητροειδή είναι διαφορετικά, είναι ακέραιο.

Λέξεις Κλειδιά

Χρονομεταβαλλόμενη Συνδυαστική Βελτιστοποίηση, Μητροειδή, Συνδετικά Δέντρα, Προσεγγιστικοί Αλγόριθμοι, Γραμμικός Προγραμματισμός

Abstract

Multistage Matroid Optimization is a time-evolving generalization of the classical minimum-weight base problem. It was proposed by Gupta, Talwar and Wieder to model systems that need to be maintained continually while the underlying costs change over time. In this time-evolving setting, the costs of the elements may change in each time step and this leads to a trade-off between maintaining a low-cost base and the "stability" of the solution. More specifically, the input is a sequence of cost functions (one for each time step); while we change the base from step to step, we incur an additional uniform acquisition cost for every such change.

In this thesis we present the first deterministic LP rounding $O(\log r)$-approximation algorithm for the problem, where r is the rank of the matroid, that achieves constant approximation at the holding cost. Gupta et al. had presented a randomized rounding scheme that also achieves $O(\log r)$ approximation, but this holds for both the holding and the acquisition cost. Our algorithm relies on the well understood structure of the independent set polytope and introduces a novel rounding technique that might be of independent interest. In addition, we provide the first proofs of integrality for linear programming relaxations for multistage combinatorial optimization problems. More specifically, we show that the natural LP for the multistage partition matroid optimization problem is integral. We also show integrality for the LP for the multistage matroid optimization, for two time steps, even if the two matroids are different.

Keywords

Multistage Combinatorial Optimization, Matroids, Spanning Trees, Approximation Algorithms, Linear Programming

Contents

Introduction

In combinatorial optimization we have problems with constraints that model an application frozen in one time step. However, in practice, one needs to solve instances of the combinatorial optimization problem that changes over time. Of course, one can trivially solve the problem independently in each time step. Nevertheless, changing the solution at consecutive time steps, often costs a **transition cost**. Consider, for example, the problem faced by a vendor who needs to get supply of an item from k different producers to meet her demand. On any given day, she could get prices from each of the producers and pick the k cheapest ones to buy from. As prices change, this set of the k cheapest producers may change. However, there is a fixed cost to starting and/or ending a relationship with any new producer. The goal of the vendor is to minimize the sum total of these two costs: an "acquisition cost" a(e) to be incurred each time she starts a new business relationship with a producer, and a per period cost $c_t(e)$ of buying in period t from the each of the k producers that she picks in this period, summed over T time periods. Observe that this problem is an example of maintaining a base of a k-uniform matroid. Finding the optimal solution is also trivial for matroids, since the greedy algorithm is optimal in this case. So, it is natural to ask, whether it is also easy to solve the multistage problem for general matroids. For example, one may want to maintain a spanning tree of a given graph at each step, where the edge costs $c_t(e)$ change over time, and an acquisition cost of $a(e)$ has to be paid every time a new edge enters the spanning tree.

The multistage matroid optimization problem was introduced by Gupta, Talwar and Wieder in [22]. The authors proved that when the acquisition costs are non-uniform, which means that if we change different elements, we may pay different acquisition costs, then the logarithmic approximation is optimal, unless $P = NP$. However, their reduction relies heavily on the non-uniformity of the acquisition costs. For this reason, we work on the case of uniform acquisition costs and we make a first step towards a constant approximation algorithm, by presenting an algorithm that has constant approximation at the holding cost and logarithmic approximation at the acquisition cost, a guarantee that the previous algorithms did not have. Furthermore, in [22], Gupta et al. prove that the problem restricted to partition matroids lies in P, even if the acquisition costs are non-uniform and time-dependent. They also show the same thing for T=2, even if the two matroids are different. We prove that the natural LPs for these problems are integral, presenting the first integrality proofs for LP relaxations for multistage optimization problems.

The time-evolving setting has also been applied to facility location problem. Eisenstat et al. in [18] introduced two variants of the problem. In both variants, the acquisition (in their paper they call it switching) cost is uniform and is payed when a client changes the facility that she is connected to. The difference lies in the fact that in the first variant the facilities do not change while in the second they can change. For both cases they present an O(lognT) approximation algorithm, where n is the number of clients and T the number of time steps. For the first case they show a matching lower bound, while for the second case An et al. in [2] presented a 14-approximation algorithm.

All these raise a natural question as far as multistage combinatorial optimization is concerned: if we have two combinatorial optimization problems A,B such that A is more difficult than B, in the static case, and we know that in the multistage setting, A has a constant approximation algorithm, is it the case that, in the multistage setting, B has also a constant approximation algorithm? The answer is **NO**, since in [22] the authors prove that $\forall \epsilon > 0$ there is no $O(n^{1-\epsilon})$-approximation algorithm for the multistage perfect matching, which for $T = 1$ lies in P. So, if the problem A is the facility location which is NP-hard and B is the min-cost perfect matching which belongs to P, then clearly the above proposition is false.

Matroids

Matroid theory was introduced by Hassler Whitney (1935) [43] and it was also independently discovered by Takeo Nakasawa. In his seminal paper, Whitney provided two axioms for independence, and defined any structure adhering to these axioms to be "matroids". His key observation was that these axioms provide an abstraction of "independence" that is common to both graphs and matrices. Because of this, many of the terms used in matroid theory resemble the terms for their analogous concepts in linear algebra or graph theory. In the 1950s W. T. Tutte became the foremost figure in matroid theory, a position he retained for many years. His contributions were plentiful, including the characterization of binary, regular, and graphic matroids by excluded minors; the regular-matroid representability theorem; the theory of chain groups and their matroids; and the tools he used to prove many of his results, the "Path theorem" and "Homotopy theorem" [41]. Henry Crapo and Thomas Brylawski [8] generalized to matroids Tutte's "dichromate", a graphic polynomial now known as the Tutte polynomial. Their work has been followed by a flood of papers.

Published in 1980, Paul Seymour's decomposition theorem for regular matroids [38] was the most significant and influential work of the late 1970s and the 1980s. Another fundamental contribution, by Kahn and Kung [26], showed why projective geometries and Dowling geometries play such an important role in matroid theory. By this time there were many other important contributors, but one should not omit to mention Geoff Whittle's extension to ternary matroids of Tutte's characterization of binary matroids that are representable over the rationals [44], perhaps the biggest single contribution of the 1990s.

Around 2000, the Matroid Minors Project of Jim Geelen, Gerards, Whittle, and others, which attempts to duplicate for matroids, that are representable over a finite field, the success of the Robertson–Seymour Graph Minors Project, has produced substantial advances in the structure theory of matroids. Many others have also contributed to that part of matroid theory, which is currently flourishing. For a detailed and compact presentation of matroid theory, we refer the reader to [36].

Edmonds in [17],[16] was the first who showed a polyhedral characterization of the independent set polytope. Optimizing a linear function subject to matroid constraint can be solved greedily, but what if we have a submodular function? Submodular maximization generalizes many fundamental problems in discrete optimization, including Max-Cut in directed/undirected graphs, maximum coverage, maximum facility location and marketing over social networks. Thus, because of its importance, this question has been extensively studied recently and there have been wonderful results in the area. We refer the interested reader to [11],[40],[10].

Related Work

Along with the work of Gupta et al. [22] and [2] et al., our work is related to several lines of research. In the online case, The MMM problem is also a special case of classical Metrical Task Systems [7]; see [1], [5] for more recent work. The best approximations for metrical task systems are poly-logarithmic in the size of the metric space. In our case the metric space is specified by the total number of bases of the matroid which is often exponential, so these algorithms only give a trivial approximation. In trying to unify online learning and competitive analysis, Buchbinder et al. [9] consider a problem on matroids very similar to ours. In their model all acquisition costs are the same and they work with fractional bases instead of integral ones. They give an O(logn)-competitive algorithm to solve the fractional online LP with uniform acquisition costs (among other unrelated results). In dynamic Steiner tree maintenance [21],[32],[24], the goal is to maintain an approximately optimal Steiner tree for a varying instance (where terminals are added) while changing few edges at each time step. In dynamic load balancing [19],[3] one has to maintain a good scheduling solution while moving a small number of jobs around.

In the offline case, Shachnai et al. [39] consider "reoptimization" problems: given a starting solution and a new instance, they want to balance the transition cost and the cost on the new instance. This is a two-timestep version of our problem, and the short time horizon raises a very different set of issues (since the output solution does not need to itself hedge against possible subsequent futures). They consider a number of optimization/scheduling problems in their framework. There is also work on "leasing" problems [4],[34],[33]: these are optimization problems where elements can be obtained for an interval of any length, where the cost is concave in the lengths; the instance changes at each timestep. The main differences are that the solution only needs to be feasible at each timestep (i.e., the holding costs are $\{0, \infty\}$), and that any element can be leased for any length ℓ of time starting at any timestep for a cost that depends only on ℓ, which gives these problems a lot of uniformity. In turn, these leasing problems are related to "buy-at-bulk" problems.

Chapters Overview

The problem that we deal with in this thesis is NP-hard and the standard approach in these cases is the design of approximation algorithms. At the same time, the approximation algorithms that we present are mostly based on linear programming. That's why, in Chapter 1, we introduce the reader to the basic LP-based techniques for approximating hard problems, via the example of Set Cover. We present a deterministic and a randomized rounding algorithm and we analyze the natural greedy algorithm using dual-fitting. After, we present some basic facts about linear programming that we will continuously invoke throughout this thesis. We outline the important Rank Lemma and other properties about extreme point solutions. We also discuss the polynomial time solvability of linear programs using the separation oracle.

In Chapter 2 we introduce the reader to the concept of matroids. We discuss why they constitute an important combinatorial abstraction and we present the basic properties and definitions. We show why the greedy algorithm works in the min-weight base problem and then we highlight the structure of the independent set polytope. Using this structure and an iterative rounding algorithm, we show that the linear program for matroid intersection is integral. Finally, we show how an extension of these techniques yields a $(k-1)-$approximation algorithm for the k-matroid intersection problem.

In Chapter 3 we present some positive and negative results on the Multistage Matroid Maintainance problem, shown in [22]. First, we show how the extension of Kruskal's algorithm in this time-evolving setting, analyzed through dual-fitting, gives a $logT$ approximation algorithm (T: number of timesteps). Then, we present a randomized rounding algorithm, which uses independent sampling in each timestep, but the randomness is shared between the timesteps. This is an $O(\log rT)$-approximation algorithm and can be modified to give an $O(\log r\frac{a_{max}}{a_{min}})$ approximation guarantee (r: rank of the matroid, a_{max}, a_{min}: the maximum and minimum acquisition costs respectively). After, we present an exact reduction from Set Cover, which shows that the logarithmic approximation is optimal. Continuing with the negative results, we show that MMM with different matroids is NP-hard to approximate better than a factor of $\Omega(T)$ as long as $T \geq 3$. Finally, we discuss the perfect matching maintainance problem and we show that, surprisingly, the hardness drastically increases: for any constant $\epsilon > 0$, there is no $O(n^{1-\epsilon})$-approximation.

Chapter 4 contains our research work. First, we show the the linear program for the MMM for partition matroids is integral, even in the case of time-dependent switching costs. Second, we prove integrality for the MMM for T=2, even when the matroids are different. Third, having observed that the reduction from set cover relies on the non-uniformity of the acquisition cost, Gupta et al. [22] asked whether there is a sublogarithmic approximation algorithm for the restriction to instances with uniform switching costs. We make a first step towards a positive answer, by presenting a deterministic algorithm with constant approximation at the holding cost and $O(\log r)$ at the acquisition cost.

Chapter 1

Introduction to Approximation Algorithms

The complexity class **P** contains the set of problems that can be solved in polynomial time. From a theoretical viewpoint, this describes the class of tractable problems, that is, problems that can be solved efficiently. The class **NP** is the set of problems that can be solved in non-deterministic polynomial time, or equivalently, problems for which a solution can be verified in polynomial time. NP contains many interesting problems that often arise in practice, but there is good reason to believe **P** $\neq$ **NP**. That is, it is unlikely that there exist algorithms to solve NP optimization problems efficiently, and so we often resort to heuristic methods to solve these problems. Heuristic approaches include backtrack search and its variants, mathematical programming methods, local seach, genetic algorithms, tabu search, simulated annealing etc. Some methods are guaranteed to find an optimal solution, though they may take exponential time; others are guaranteed to run in polynomial time, though they may not return an optimal solution. Approximation algorithms fall in the latter category; however, though they do not find an optimal solution, we can give guarantees on the quality of the solution found.

Definition 1. *An α-approximation algorithm for an optimization problem is a polynomial time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution.*

In this chapter we give a very brief overview of the basic techniques for designing approximation algorithms, based on linear programming. Our presentation is based on [45] and [42], where one can find a thorough presentation of the subject.

1.1 A linear programming formulation for the Set Cover problem

Definition 2. *In the Set Cover problem, we are given a ground set of elements $\mathcal{E} = \{e_1, ..., e_n\}$, some subsets of those elements $S_1, S_2, ..., S_m$ where each $S_j \subseteq \mathcal{E}$, and a non-*

negative weight $w_j \geq 0$ for each subset S_j. The goal is to find a minimum-weight collection of subsets that covers all of $\mathcal{E}$; that is, we wish to find an $I \subseteq [m]$ that minimizes $\sum_{j \in I} w_j$ subject to $\cup_{j \in I} S_j = \mathcal{E}$. If $w_j = 1$ for each subset j, the problem is called the unweighted set cover problem.

Observe that the Set Cover generalizes vertex cover. To see that the vertex cover problem is a special case of the set cover problem, for any instance of the vertex cover problem, create an instance of the set cover problem in which the ground set is the set of edges, and a subset S_i of weight w_i is created for each vertex $i \in V$ containing the edges incident to i. It is not difficult to see that for any vertex cover C, there is a set cover I = C of the same weight, and vice versa.

In this thesis, linear programming plays a central role in the design and analysis of approximation algorithms. Many of the techniques that we will use, are based on the theory of integer and linear programming in one way or another. Here we will give a very brief introduction to the area in the context of the set cover problem. For a much more in depth analysis of linear programming we refer the reader to [28]. Each linear program or integer program is formulated in terms of some number of decision variables that represent some sort of decision that needs to be made. The variables are constrained by a number of linear inequalities and equalities called constraints. Any assignment of real numbers to the variables such that all of the constraints are satisfied is called a *feasible solution*. In the case of the set cover problem, we need to decide which subsets S_j to use in the solution. We create a decision variable x_j to represent this choice. In this case we would like x_j to be 1 if the set S_j is included in the solution, and 0 otherwise. Thus, we introduce constraints $x_j \leq 1$ for all subsets S_j , and $x_j \geq 0$ for all subsets S_j . This is not sufficient to guarantee that $x_j \in \{0, 1\}$, so we will formulate the problem as an integer program to exclude fractional solutions (that is, nonintegral solutions); in this case, we are also allowed to constrain the decision variables to be integers. Requiring x_j to be integer along with the constraints $x_j \geq 0$ and $x_j \leq 1$ is sufficient to guarantee that $x_j \in \{0, 1\}$. We also want to make sure that any feasible solution corresponds to a set cover, so we introduce additional constraints. In order to ensure that every element e_i is covered, it must be the case that at least one of the subsets S_j containing e_i is selected. This will be the case if

$$\sum_{j : e_i \in S_j} x_j \geq 1$$

for each e_i, $i = 1, ..., n$.

In addition to the constraints, linear and integer programs are defined by a linear function of the decision variables called the *objective function*. The linear or integer program seeks to find a feasible solution that either maximizes or minimizes this objective function. Such a solution is called an optimal solution. The value of the objective function for a particular feasible solution is called the *value* of that solution. The value of the objective function for an optimal solution is called the value of the linear (or integer) program. We say we solve the linear program if we find an optimal solution. In the case of the set cover problem, we want to find a set cover of minimum weight. Given the decision variables x_j and constraints described above, the weight of a set cover given the x_j variables is $\sum_{j=1}^{m} w_j x_j$. Thus, the objective function of the integer program is $\sum_{j=1}^{m} w_j x_j$, and we wish to minimize this function. Integer and linear programs are usually written in a compact form stating first the objective function and then the constraints. Given the discussion above, the problem of finding a

minimum-weight set cover is equivalent to the following integer program:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j:e_i \in S_j} x_j \geq 1, \quad i = 1, ..., n \\ & x_j \in \{0,1\}, \qquad j = 1, ..., m \end{aligned} \tag{1.1}$$

Let Z_{IP}^* denote the optimum value of this integer program for a given instance of the set cover problem. Since the integer program exactly models the problem, we have that $Z_{IP}^* = OPT$ where OPT is the value of an optimum solution to the set cover problem. In general, integer programs cannot be solved in polynomial time. This is clear because the set cover problem is NP-hard, so solving the integer program above for any set cover input in polynomial time would imply that $P = NP$. However, linear programs are polynomial-time solvable. In linear programs we are not allowed to require that decision variables are integers. Nevertheless, linear programs are still extremely useful: even in cases such as the set cover problem, we are still able to derive useful information from linear programs. For instance, if we replace the constraints $x_j \in \{0,1\}$ with the constraints $x_j \geq 0$, we obtain the following linear program, which can be solved in polynomial time:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j:e_i \in S_j} x_j \geq 1, \quad i = 1, ..., n \\ & x_j \geq 0, \qquad j = 1, ..., m \end{aligned} \tag{1.2}$$

It is easy to see that the linear program (1.2) is a *relaxation* of the original integer program. By this we mean two things: first, every feasible solution for the original integer program (1.1) is feasible for this linear program; and second, the value of any feasible solution for the integer program has the same value in the linear program. m. Let Z_{LP}^* denote the optimum value of this linear program. Any optimal solution to the integer program is feasible for the linear program and has value Z_{IP}^*. Thus, any optimal solution to the linear program will have value $Z_{LP}^* \leq Z_{IP}^* = OPT$, since this minimization linear program finds a feasible solution of lowest possible value.

In the following sections, we will give some examples of how the linear programming relaxation can be used to derive approximation algorithms for the set cover problem. Because we will frequently be referring to linear programs and linear programming, we will often abbreviate these terms by the acronym LP. Similarly, IP stands for either integer program or integer programming.

1.2 Deterministic Rounding

Suppose that we solve the linear programming relaxation of the set cover problem. Let x^* denote an optimal solution to the LP. How then can we recover a solution to the set cover problem? Here is a very easy way to obtain a solution: given the LP solution x^*, we include subset S_j in our solution if and only if $x_j^* \geq 1/f$, where f is the maximum number of sets in which any element appears. More formally, let $f_i = |\{j : e_i \in S_j\}|$ be the number of sets in which element e_i appears, $i = 1, ..., n$; then $f = max_{i \in [n]} f_i$. Let I denote the indices j of the subsets in this solution. In effect, we round the fractional solution x^* to an integer solution $\hat{x}$ by setting $\hat{x}_j = 1$, if $x_j^* \geq 1/f$, and $\hat{x}_j = 0$ otherwise. We shall see that it is straightforward to prove that $\hat{x}$ is a feasible solution to the integer program, and I indeed indexes a set cover.

Lemma 1.2.1. *The collection of subsets S_j, $j \in I$, is a set cover.*

Proof. Consider the solution specified by the lemma, and call an element e_i covered if this solution contains some subset containing e_i. We show that each element e_i is covered. Because the optimal solution x^* is a feasible solution to the linear program, we know that $\sum_{j:e_i \in S_j} x_j^* \geq 1$ for element e_i. By the definition of f_i and of f, there are $f_i \leq f$ terms in the sum, so at least one term must be at least 1/f. Thus, for some j such that $e_i \in S_j$, $x_j^* \geq 1/f$. Therefore, $j \in I$, and element e_i is covered. □

Lemma 1.2.2. *The rounding algorithm is an f-approximation algorithm for the set cover problem.*

Proof. It is clear that the algorithm runs in polynomial time. By our construction, $1 \leq f x_j^*$ for each $j \in I$. From this, and the fact that each term $f w_j x_j^*$ is nonnegative for $j = 1, ..., m$, we see that

$$\sum_{j \in I} w_j \leq \sum_{j=1}^{m} w_j (f x_j^*) = f Z_{LP}^* \leq f OPT$$

□

In the special case of the vertex cover problem, $f_i = 2$ for each vertex $i \in V$, since each edge is incident to exactly two vertices. Thus, the rounding algorithm gives a $2 - approximation$ algorithm for the vertex cover problem.

Observe that that if $\frac{Z_{IP}^*}{Z_{LP}^*}$ is very large, then we cannot hope for a good rounding algorithm. That's why we introduce the notion of *integrality gap*.

Definition 3. *The integrality gap of an integer program is the worst-case ratio over all instances of the problem of value of an optimal solution to the integer programming formulation to value of an optimal solution to its linear programming relaxation.*

We present an example of bounding the integrality gap of an integer program. Consider the case of the unweighted vertex cover IP, as a special case of the set cover IP. Now consider the instance of the complete graph with n vertices, K_n. Clearly, the optimal solution of the integer program in n-1. However, at the LP, by setting all variables to $\frac{1}{2}$, we satisfy all the constraints and the value of the objective function is $\frac{n}{2}$. Since $\frac{n-1}{n/2} \to 2$, the integrality gap is at least two. Since now, as we proved, $\frac{Z^*_{IP}}{Z^*_{LP}} \leq 2$, we conclude that the integrality gap is exactly 2 for this IP for the vertex cover and that there is no hope to design a better rounding algorithm than the $2-approximation$ algorithm that we already presented.

1.3 Duality

Often it will be useful to consider the dual of the linear programming relaxation of a given problem. Again, we will give a very brief introduction to the concept of the dual of a linear program. To begin, we suppose that each element e_i is charged some nonnegative price $y_i \geq 0$ for its coverage by a set cover. Intuitively, it might be the case that some elements can be covered with low-weight subsets, while other elements might require high-weight subsets to cover them; we would like to be able to capture this distinction by charging low prices to the former and high prices to the latter. In order for the prices to be reasonable, it cannot be the case that the sum of the prices of elements in a subset S_j is more than the weight of the set, since we are able to cover all of those elements by paying weight w_j. Thus, for each subset S_j we have the following limit on the prices:

$$\sum_{i:e_i \in S_j} y_i \leq w_j$$

We can find the highest total price that the elements can be charged by the following linear program:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} y_i \\ \text{subject to} \quad & \sum_{i:e_i \in S_j} y_i \leq w_j, \quad j = 1, ..., m \\ & y_i \geq 0, \qquad i = 1, ..., n \end{aligned} \tag{1.3}$$

This linear program is the dual linear program of the set cover linear programming relaxation (1.2). We can in general derive a dual linear program for any given linear program, but we will not go into the details of how to do so; see [28]. If we derive a dual for a given linear program, the given program is sometimes called the primal linear program. For instance, the original linear programming relaxation (1.2) of the set cover problem is the primal linear program of the dual (1.3). Notice that this dual has a variable y_i for each constraint of the primal linear program (that is, for the constraint $\sum_{j:e_i \in S_j} x_j \geq 1$) **(1)**, and has a constraint for each variable x_j of the primal. This is true of dual linear programs in general.

Dual linear programs have a number of very interesting and useful properties. For example, let x be any feasible solution to the set cover linear programming relaxation, and let y be any feasible set of prices (that is, any feasible solution to the dual linear program). Then consider the value of the dual solution y:

$$\sum_{i=1}^{n} y_i \leq \sum_{i=1}^{n} y_i \sum_{j:e_i \in S_j} x_j = \sum_{j=1}^{m} x_j \sum_{i:e_i \in S_j} y_i \leq \sum_{j=1}^{m} w_j x_j$$

where in the first step we use the constraints of the primal program, in the second step we reverse the sums and in the third step we use the constraints of the dual program. So, any feasible solution to the dual linear program has a value no greater than any feasible solution to the primal linear program. In particular, any feasible solution to the dual linear program has a value no greater than the optimal solution to the primal linear program, so for any feasible y, $\sum_{i=1}^{n} y_i \leq Z^*_{LP}$. This is called the *weak duality* property of linear programs. Since we previously argued that $Z^*_{LP} \leq OPT$, we have that for any feasible y, $\sum_{i=1}^{n} y_i \leq OPT$. This is a very useful property that will help us in designing approximation algorithms.

Additionally, there is a quite amazing strong duality property of linear programs. Strong duality states that as long as there exist feasible solutions to both the primal and dual linear programs, their optimal values are equal. Thus, if x^* is an optimal solution to the set cover linear programming relaxation, and y^* is an optimal solution to the dual linear program, then

$$\sum_{i=1}^{n} y^*_i = \sum_{j=1}^{m} w_j x^*_j$$

1.4 The Greedy Algorithm and the Dual-Fitting technique

The greedy strategy applies naturally to the set cover problem: iteratively pick the most cost-effective set and remove the covered elements, until all elements are covered. Let C be the set of elements already covered at the beginning of an iteration. During this iteration, define the cost-effectiveness of a set S to be the average weight at which it covers new elements, i.e., $w(S)/|S \setminus C|$. Define the price of an element to be the average cost at which it is covered. Equivalently, when a set S is picked, we can think of its cost being distributed equally among the new elements covered, to set their prices.

Algorithm 1 Greedy set cover algorithm

```
$C \leftarrow \emptyset$.
**while** $C \neq \mathcal{E}$ **do**
    Find the most cost-effective set in the current iteration, say $S$.
    Let $\alpha = \frac{w(S)}{|S \setminus C|}$, i.e., the cost-effectiveness of S.
    Pick $S$, and for each $e \in S \setminus C$, set $price(e) = \alpha$.
    $C \leftarrow C \cup S$
output the picked sets
```

We analyze the greedy algorithm via a method called *dual-fitting* The method of dual fitting can be described as follows, assuming a minimization problem: The basic algorithm is combinatorial – in the case of set cover it is in fact the simple greedy algorithm. Using the linear programming relaxation of the problem and its dual, one shows that the primal integral solution found by the algorithm is fully paid for by the dual computed; however, the dual is infeasible. By fully paid for we mean that the objective function value of the primal solution found is at most the objective function value of the dual computed. The main step in the analysis consists of dividing the dual by a suitable factor and showing that the shrunk dual is feasible, i.e., it fits into the given instance. The shrunk dual is then a lower bound on OPT, and the factor is the approximation guarantee of the algorithm. Now, we apply this method to set cover. The greedy algorithm defines dual variables price(e), for each element, e. Observe that the cover picked by the algorithm is fully payed for by this dual solution. However, in general, this dual solution is not feasible. We will show below that if this dual is shrunk by a factor of $H_n = \sum_{i=1}^{n} \frac{1}{i}$, it fits into the given set cover instance, i.e., no set is overpacked. For each element e define, $y_e = \frac{price(e)}{H_n}$.

Lemma 1.4.1. *The vector y defined above is a feasible solution for the dual program.*

Proof. We need to show that no set is overpacked by the solution y . Consider a set $\mathcal{S}_j$ consisting of k elements. Number the elements in the order in which they are covered by the algorithm, breaking ties arbitrarily, say $e_1, ..., e_k$.

Consider the iteration in which the algorithm covers element e_i. At this point, S_j contains at least $k - i + 1$ uncovered elements. Thus, in this iteration, $\mathcal{S}_j$ itself can cover e_i at an average cost of at most $\frac{w_j}{k-i+1}$. Since the algorithm chose the most cost-effective set in this iteration, $price(e_i) \leq \frac{w_j}{k-i+1}$. Thus,

$$y_{e_i} \leq \frac{1}{H_n} \frac{w_j}{k - i + 1}$$

Summing over all the elements of S_j:

$$\sum_{i=1}^{k} y_{e_i} \leq \frac{1}{H_n} w_j \sum_{i=1}^{k} \frac{1}{k - i + 1} = \frac{H_k}{H_n} w_j \leq w_j$$

Therefore, S_j is not overpacked. □

Lemma 1.4.2. *The approximation guarantee of the g reedy set c over algo- rithm is H_n.*

Proof. The cost of the set cover picked is

$$\sum_{e \in E} price(e) = H_n \sum_{i=1}^{n} y_{e_i} \leq H_n OPT$$

The last inequality follows from the fact that y is dual feasible. □

1.5 A Randomized Rounding algorithm

In this section, we consider one final technique for devising an approximation algorithm for the set cover problem. Although the algorithm is slower and has no better guarantee than the greedy algorithm of the previous section, we include it here because it introduces the notion of using randomization in approximation algorithms.

The algorithm will solve a linear programming relaxation for the set cover problem, and then round the fractional solution to an integral solution. Rather than doing so deterministically, however, the algorithm will do so randomly using a technique called randomized rounding . Let x^* be an optimal solution to the LP relaxation. We would like to round fractional values of x^* to either 0 or 1 in such a way that we obtain a solution $\hat{x}$ to the integer programming formulation of the set cover problem without increasing the cost too much. The central idea of randomized rounding is that we interpret the fractional value x_j^* as the probability that $\hat{x}_j$ should be set to 1. Thus, each subset S_j is included in our solution with probability x_j^* , where these m events (that S_j is included in our solution) are independent random events.

Let X_j be a random variable that is 1 if subset S_j is included in the solution, and 0 otherwise. Then the expected value of the solution is

$$\mathbb{E}[\sum_{j=1}^{m} w_j X_j] = \sum_{j=1}^{m} w_j Pr(X_j = 1) = \sum_{j=1}^{m} w_j x_j^* = Z_{LP}^*$$

or just the value of the linear programming relaxation, which is no more than OPT! As we will see, however, it is quite likely that the solution is not a set cover. Nevertheless, this illustrates why randomized rounding can provide such good approximation algorithms in some cases.

Let us now calculate the probability that a given element e_i is not covered by this procedure. This is the probability that none of the subsets containing e_i are included in the solution, or

$$\prod_{j:e_i \in S_j} (1 - x_j^*) \leq \prod_{j:e_i \in S_j} e^{-x_j^*} = e^{-\sum j:e_i \in S_j x_j^*} \leq e^{-1}$$

where the first step follows the inequality $e^x \geq 1 + x, \forall x \in \mathbb{R}$ and the last step follows from the fact that $\sum j : e_i \in S_j x_j^* \geq 1$. However, we would like this probability to be much smaller, in order to be very very likely to end up with a set cover. In fact, we can achieve such a bound in the following way. Fix a constant $c \geq 2$. For each subset S_j , we imagine a coin that comes up heads with probability x_j^* , and we flip the coin $c \ln n$ times. If it comes up heads in any of the $clnn$ trials, we include S_j in our solution, otherwise not. Thus, the probability that S_j is not included is $(1 - x_j^*)^{clnn}$. Furthermore

$$Pr[e_i \text{ } is \text{ } not \text{ } covered] = \prod (1 - x_j^*)^{clnn} = e^{-clnn \sum j:e_i \in S_j x_j^*} \leq e^{-clnn} = \frac{1}{n^c}$$

Thus, from the union bound we have that

$$Pr[some \text{ } e_i \text{ } is \text{ } not \text{ } covered \text{ } i = 1, ..., n] \leq \frac{1}{n^{c-1}}$$

We now need to prove only that the algorithm has a good expected value given that it produces a set cover.

Theorem 1.5.1. *The algorithm is a randomized $O(lnn)$ -approximation algorithm that produces a set cover with high probability.*

Proof. If X_j is a random variable that is 1 if the subset S_j is included in the solution, and 0 otherwise, from the union bound, $Pr(X_j = 1) \leq clnnx_j^*$. Thus, the expected value of the random procedure is $\mathbb{E}[\sum_{j=1}^m w_j X_j] \leq clnn \sum_{j=1}^m w_j x_j^* = clnnZ_{LP}^*$ However, we would like to bound the expected value of the solution given that a set cover is produced. Let F be the event that the solution obtained by the procedure is a feasible set cover, and let $\bar{F}$ be the complement of this event. We know from the previous discussion that $Pr(F) \geq 1 - \frac{1}{n^{c-1}}$. Hence, since $\sum_{j=1}^m w_j X_j$ is a non-negative random variable,

$$\mathbb{E}[\sum_{j=1}^m w_j X_j | \; F] \leq \frac{1}{Pr(F)} \mathbb{E}[\sum_{j=1}^m w_j X_j] \leq \frac{clnnZ_{LP}^*}{1 - \frac{1}{n^{c-1}}} \leq 2clnnZ_{LP}^*$$

for $n \geq 2$ and $c \geq 2$. □

1.6 Basic facts about Linear Programming

In this section we discuss linear programming and we present, without proof, basic facts about extreme point solutions to linear programs that are necessary in order to keep up with the topics that we analyze in this thesis. We then briefly discuss solution methods for linear programs, particularly stating the sufficiency of finding a separation oracle for the program to be able to solve it. Excellent introductory textbooks in this area are [28], [6].

Linear Programming

Using matrix notation, a linear program is expressed as follows:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \geq b \\ & x \geq 0 \end{array}$$

If x satises $Ax \geq b, x \geq 0$, then x is a feasible solution. If there exists a feasible solution to the linear program, it is feasible; otherwise it is infeasible. An optimal solution x^* is a feasible solution such that $c^T x^* = min\{c^T x \ s.t. \ Ax \geq b, x \geq 0\}$. The linear program is unbounded (from below) if $\forall \lambda \in \mathbb{R}$, there exists feasible x such that $c^T x < \lambda$. There are different forms in which a linear program can be represented. However, all these forms are equivalent to the form we consider above and can be converted into one another by simple linear transformations, see [28].

Extreme Point Solutions to Linear Programs

Definition 4. *Let $P = \{x : Ax = b, x \geq 0\} \subseteq \mathbb{R}^n$. Then $x \in \mathbb{R}^n$ is an **extreme point** solution of P if there does not exist a non-zero vector $y \in \mathbb{R}^n$ such that $x + y, x - y \in P$.*

Pictorially extreme point solutions are the corner points of the set of feasible solutions. The following basic result shows that there is always an optimal extreme point solution to bounded linear programs.

Lemma 1.6.1. *Let $P = \{x : Ax = b, x \geq 0\} \subseteq \mathbb{R}^n$ and assume that the optimum value min$\{c^T x$ s.t. $x \in P\}$ is finite. Then for any feasible solution $x \in P$, there exists an extreme point solution $x' \in P$ with $c^T x' \leq c^T x$.*

We now proceed with another characterization of the extreme point solutions. A subset of columns B of the constraint matrix A is called a basis if the matrix of columns corresponding to B, i.e. A_B, is invertible. A solution x is called basic if and only if there is a basis B such that $x_j = 0$ if $j \notin B$ and $x_B = A_B^{-1} b$. If in addition to being basic, it is also feasible, i.e., $A_B^{-1} b \geq 0$, it is called a **basic feasible solution** for short. The correspondence between bases and basic feasible solutions is not one to one. Indeed there can be many bases which correspond to the same basic feasible solution. The next theorem shows the equivalence of extreme point solutions and basic feasible solutions.

Theorem 1.6.2. *Let A be a $m \times n$ matrix with full row rank. Then every feasible x to $P = \{x : Ax = b, x \geq 0\}$ is a basic feasible solution if and only if x is an extreme point solution.*

The following Rank Lemma is an important ingredient in the correctness proofs of almost all iterative algorithms in this thesis.

Lemma 1.6.3. ***(Rank Lemma)*** *Let $P = \{x : Ax = b, x \geq 0\}$ and let x be an extreme point solution of P such that $x_i > 0$ for each i. Then the number of variables is equal to the number of linearly independent constraints of A, i.e. the rank of A.*

Finally, we present the definition of the integral polytope:

Definition 5. *Let P be a polytope and let x be an extreme point solution of P then x is integral if each coordinate of x is an integer. The polytope P is called integral if every extreme point of P is integral.*

Algorithms for Linear Programming

The simplex algorithm solves linear programs to get a basic feasible optimal solution. It works by starting at any basic feasible solution and moving to a neighboring basic feasible solution which improves the objective function. The convexity of the linear program ensures that once the simplex algorithm ends at a local optimum basic feasible point, it has achieved the global optimum as well. Many variants of the simplex algorithm have been considered, each defined by which neighboring basic feasible solution to move in case there are more than one improving basic feasible points in the neighborhood. Although the simplex algorithm works efficiently in practice, there are examples where each variant of the simplex algorithm runs in exponential time. Again, for more details, see e.g. [36]. Polynomial-time algorithms for solving linear programs fall in two categories: ellipsoid algorithms [30] and interior point algorithms [29].

Theorem 1.6.4. *There is an algorithm which returns an optimal extreme point solution to a linear program. Moreover, the running time of the algorithm is polynomial in the size of the linear program.*

In this thesis, we will also encounter linear programs where the number of constraints is exponential in the size of the problem (e.g., in the maximum-weight independent set problem in chapter 2) and it is not obvious that one can enumerate them, let alone solve them in polynomial time. We use the notion of separation to show that many exponentially sized linear programs can be solved in polynomial time.

Definition 6. *Given $x^* \in \mathbb{R}^n$ and a polytope $P = \{x : Ax = b, x \geq 0\}$, the separation problem is the decision problem whether $x^* \in P$. The solution of the separation problem is the answer to the membership problem and in case $x^* \notin P$, it should return a valid constraint $A_i x \geq b_i$ for P which is violated by x^*, i.e., $A_i x^* < b_i$.*

The following theorem of Grotschel, Lóvasz and Schrijver [20] shows that polynomial time separability is equivalent to polynomial time solvability of a linear program; we state it in a form that is convenient for combinatorial optimization problems. The basis of this equivalence is the ellipsoid algorithm.

Theorem 1.6.5. *Given a full-dimensional polytope P and a polynomial-time separation oracle for P, one can find an optimal extreme point solution to a linear objective function over P (assuming it is bounded) via the Ellipsoid algorithm that uses a polynomial number of operations and calls to the separation oracle.*

Clearly, one can solve the separation problem by checking each constraint but for problems where the number of constraints is exponential in size such a method is too slow. In this thesis, as we consider LP formulations with an exponential number of constraints, we will often provide efficient separation oracles showing that the linear program for the problem is solvable in polynomial time.

Chapter 2

Matroids and the Iterative Method

Matroids were introduced by Whitney in 1935 to try to capture abstractly the essence of dependence. Whitney's definition embraces a surprising diversity of combinatorial structures, like spanning trees. After introducing matroids and stating some basic properties, we address the two most important polynomial-time solvable problems in this formalism: that of finding a maximum weight basis and of finding a maximum weight common independent set of two matroids (the so-called two-matroid intersection problem). We show integral characterizations for both problems by exploiting the structure of the extreme points of the corresponding LP's. Finally, by using a method called "iterative rounding", we present a $(k-1)$-approximation algorithm for the unweighted k matroid intersection problem: finding a maximum cardinality common independent set in k matroids defined on the same ground set. The work of Jack Edmonds [17],[15] first showed the polyhedral characterization results presented in this chapter.

2.1 Preliminaries

Definition 7. *A pair $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ is a matroid if $\mathcal{I}$ is a nonempty collection of subsets of $\mathcal{S}$ with the following properties:*

1. $\emptyset \in \mathcal{I}$

2. $A \in \mathcal{I}$ *and* $B \subseteq A \Rightarrow B \in \mathcal{I}$

3. $A, B \in \mathcal{I}$ *and* $|A| > |B| \Rightarrow \exists x \in B \setminus A$ *such that* $A \cup \{x\} \in \mathcal{I}$

$\mathcal{S}$ is called the ground set of the matroid $\mathcal{M}$. A set $A \subseteq S$ is called independent if $A \in \mathcal{I}$ else it is called dependent. A maximal set $A \in \mathcal{I}$ is called a basis of M. Observe that Property 3 implies that all bases have the the same cardinality.

Examples of Matroids

1. **Graphic Matroid**: Given an undirected graph $G = (V, E)$, the graphic matroid of G is defined as $\mathcal{M}_G = (E, \mathcal{I}_G)$ where $\mathcal{I}_G = \{F \subseteq E \mid F \textit{ contains no cycles}\}$.

2. **Uniform Matroid**: Given a set $\mathcal{S}$ and an integer $k \geq 0$, the uniform matroid of rank k is defined as $\mathcal{M}_{\mathcal{S}}^k = (\mathcal{S}, \mathcal{I}_k)$ where $\mathcal{I}_k = \{T \subseteq S : \; |T| \leq k\}$.

3. **Partition Matroid**: Let $S_1, S_2, ..., S_n$ be a partition of $\mathcal{S}$ and $k_1, k_2, ..., k_n$ be nonnegative integers. Let $\mathcal{I} = \{T \subseteq \mathcal{S} : \; |T \cap S_i| \leq k_i \textit{ for all } 1 \leq i \leq n\}$.

4. **Linear Matroid**: Let A be an $m \times n$ matrix and $\mathcal{S} = \{1, ..., n\}$. For any $1 \leq i \leq n$, let A^i denote the i^{th}-column of A. The linear matroid over matrix A is defined as $\mathcal{M}_A = (\mathcal{S}, \mathcal{I}_A)$ where $\mathcal{I}_A = \{T \subseteq \mathcal{S} : \; A^i \textit{ for } i \in T \textit{ are linearly independent}\}$.

5. **Matroid Restriction**: Let $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ be a matroid and $T \subseteq \mathcal{S}$. Then the matroid restriction of $\mathcal{M}$ to the set T is the matroid $\mathcal{M}_T = (T, \mathcal{I}_T)$ where $\mathcal{I}_T = \{R : R \in \mathcal{I}, \; R \subseteq T\}$.

Let's check that the Graphic Matroid is indeed a matroid. First of all, the independent sets are all the forests. The empty set is a forest and if we remove edges from a forest it remains a forest. It remains to check the third property. So, let $F_1, F_2 \subseteq E$ with $|F_1| > |F_2|$ and let's suppose that $\forall e \in F_1 : \; F_2 \cup \{e\}$ is not a forest. For this to happen, it must be the case that all the edges of F_1 lie inside the connected components of F_2 and thus F_2 has less connected components than F_1, contradiction. It is similarly straightforward to show that the other examples are also matroids.

Base, Circuit, Rank, Span

Let $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ be a matroid. In the following lines, for a subset A of $\mathcal{S}$ and a $x \in \mathcal{S}$ we write $A + x$ for $A \cup \{x\}$ and $A - x$ for $A \setminus \{x\}$.

Definition 8. *A set $X \in \mathcal{S}$ such that $X \notin \mathcal{I}$ is called a dependent set of $\mathcal{M}$.*

Definition 9. *A loop is an element $x \in \mathcal{S}$ such that $\{x\}$ is dependent. Notice that a loop cannot appear in any sets in $\mathcal{I}$.*

Definition 10. *A base is an inclusion wise maximal set in $\mathcal{I}$.*

Proposition 2.1.0.1. *If B and $\hat{B}$ are bases of $\mathcal{M}$ then $|B| = |\hat{B}|$.*

The proof is straightforward from the third matroid property. Notice that the notion of base here is similar to that of a basis in linear algebra.

Lemma 2.1.1. *Let B and $\hat{B}$ be two different bases of $\mathcal{M}$. Let $x \in \hat{B} \setminus B$, then $\exists y \in B \setminus \hat{B}$ such that $\hat{B} - x + y$ is a base of $\mathcal{M}$.*

Proof. Since $\hat{B} - x \in \mathcal{I}$, $|\hat{B} - x| < |B|$, then $\exists y \in B \setminus \hat{B}$ such that $\hat{B} - x + y \in \mathcal{I}$. Since $|\hat{B} - x + y| = |B|$, $\hat{B} - x + y$ is a base. □

Definition 11. *Given $\mathcal{M} = (\mathcal{S},\mathcal{I})$ and $\hat{S} \subseteq S$, $\hat{B}$ is a base for $\hat{S}$ if $\hat{B}$ is a base of $\hat{\mathcal{M}}$, where $\hat{\mathcal{M}}$, is a restriction of $\mathcal{M}$ to $\hat{S}$.*

Proposition 2.1.1.1. *Given $\mathcal{M} = (\mathcal{S},\mathcal{I})$, let B be a base for X. Then for any $Y \supseteq X$, there exists a base $\hat{B}$ for Y that contains B.*

Proof. Notice that B is independent in the restriction of M to Y (henceforth independent in Y). Let $\hat{B}$ be the maximal independent set in Y that contains B. Since all maximal independent sets have same size, $\hat{B}$ is a base of Y . □

Definition 12. *Given $\mathcal{M} = (\mathcal{S},\mathcal{I})$, a circuit is a minimal dependent set (i.e., an inclusion wise minimal set in $2^{\mathcal{S}} \setminus \mathcal{I}$). Thus, if C is a circuit then $\forall x \in C : \ C - x \in \mathcal{I}$.*

The definition of a circuit is related to graph theory in the following sense: if $\mathcal{M}$ is the graphic matroid of a graph G, then the circuits of $\mathcal{M}$ are the cycles of G. Single element circuits of a matroid are loops. If M is a graphic matroid of a graph G, then the set of loops of M is precisely the set of loops of G.

Definition 13. *Given a matroid $\mathcal{M} = (\mathcal{S},\mathcal{I})$, the rank function $r_{\mathcal{M}} : 2^{\mathcal{S}} \to \mathbb{N}$ of the matroid $\mathcal{M}$ is defined as $r_{\mathcal{M}}(T) = max\{|U| : \ U \subseteq T \text{ and } U \in \mathcal{I}\}$.*

So, r(A) is the cardinality of the bases of A. We will drop the subscript $\mathcal{M}$ from the rank function $r_{\mathcal{M}}$ when the matroid $\mathcal{M}$ is clear from the context. Observe that $A \in \mathcal{I}$ if and only if $r(A) = |A|$. Also, a property of the rank function of matroids is that it belongs to a very important family of set functions, the submodular functions. In mathematics, a **submodular set function** (also known as a submodular function) is a set function whose value, informally, has the property that the difference in the incremental value of the function that a single element makes when added to an input set decreases as the size of the input set increases. We proceed with the formal definition.

Definition 14. *If $\mathcal{S}$ is a finite set, a submodular function is a set function $f : 2^{\mathcal{S}} \to \mathbb{R}$, which satisfies one of the following equivalent definitions*

1. *For every $X, Y \subseteq \mathcal{S}$ with $X \subseteq Y$ and every $x \in \mathcal{S} \setminus Y, x \in \mathcal{S} \setminus Y$ we have that $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$.*
2. *For every $S, T \subseteq \mathcal{S}$ we have that $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$.*

For the proof of the equivalence, we refer the reader to [37]. A special class of the submodular functions are the **modular functions**, which satisfy property 2 in the above definition with equality, i.e, For every $S, T \subseteq \mathcal{S}$ we have that $f(S) + f(T) = f(S \cup T) + f(S \cap T)$. It is easy to see that the indicator function is modular.

Lemma 2.1.2. *Let r be the rank function of matroid $\mathcal{M} = (\mathcal{S},\mathcal{I})$. Then r is submodular.*

Proof. We will use the first definition of submodularity. Observe that the differences that we have to compare are 0 or 1. Now, let's suppose that $r(Y \cup \{x\}) - r(Y) = 1$. Every base B of Y+x contains x. Let $\hat{B}$ be a base of X. Since $X \subseteq Y + x$, from Proposition 2.1.1.1, there exists a base B of Y+x such that $B \supseteq \hat{B}$. Then $\hat{B} + x$ is independent, implying $r_{\mathcal{M}}(X + x) - r_{\mathcal{M}}(X) = 1$ as $\hat{B} + x$ is a base in X+x. □

Exactly as in the case of linear matroids, we define the span of a subset of the ground set:

Definition 15. *Let $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ be a matroid. For any $X \subseteq \mathcal{S}$, the span of X, denoted by $span_{\mathcal{M}}(X)$, is defined as $span_{\mathcal{M}}(X) = \{y : \ y \in \mathcal{S}$ and $r_{\mathcal{M}}(X + y) = r_{\mathcal{M}}(X)\}$. A set $X \subseteq \mathcal{S}$ is spanning if $span_{\mathcal{M}}(X) = \mathcal{S}$.*

For more properties of bases, rank, span and circuits, we refer the reader to [37].

We now define two important operations on matroids.

Definition 16. (Deletion) *Given a matroid $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ and $x \in \mathcal{S}$ we define $\mathcal{M} \setminus x = (\mathcal{S} - x, \mathcal{I}_1)$, where $\mathcal{I}_1 = \{T - x \ : \ T \in \mathcal{I}\}$ to be the matroid obtained by deleting x from $\mathcal{M}$. The rank function of $\mathcal{M} \setminus x$, denoted by r_1, is related to the rank function r of $\mathcal{M}$ by the formula $r_1(T) = r(T)$ for $T \subseteq S - x$.*

Definition 17. (Contraction) *Given a matroid $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ and $x \in \mathcal{S}$ we define $\mathcal{M}/x = (\mathcal{S}-x, \mathcal{I}_2)$ as the matroid obtained by contracting x in $\mathcal{M}$, where $\mathcal{I}_2 = \{T \subseteq \mathcal{S}-x \ : \ T+x \in \mathcal{I}\}$, if $\{x\}$ is independent, and $\mathcal{I}_2 = \mathcal{I}$ if $\{x\}$ is dependent. The rank function of M/x, denoted by r_2, is related to the rank function of $\mathcal{M}$ by the formula $r_2(T) = r(T+x) - r(\{x\})$ for $T \subseteq \mathcal{S} - x$. Note that if $\{x\}$ is dependent, then $\mathcal{M}/x = \mathcal{M} \setminus x$.*

2.2 Maximum Weight Independent Set

Matroids have some important algorithmic properties, the simplest one being that the problem of determining the maximum weight independent set in a matroid can be solved using a greedy algorithm. The maximum weight independent set problem is stated as follows: Given $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ and $w : \ \mathcal{S} \rightarrow \mathbb{R}$, output

$$\max_{X \in \mathcal{I}} w(X)$$

However, before presenting the algorithm, we have to clarify the computational model on which we are working. More specifically, if the matroid was given to us through a list containing all its independent sets, then our input could be exponential in $|\mathcal{S}|$. Instead we resort to one of the following two oracles in order to efficiently solve optimization problems:

- An independence oracle that given $A \subseteq \mathcal{S}$, returns whether $A \in \mathcal{I}$ or not.
- A rank oracle that given $A \subseteq S$, returns $r_M(A)$.

Algorithm 2 The Greedy Algorithm

```
Remove from $\mathcal{M}$ all the elements that have negative weight.
Let $\mathcal{S} = \{e_1, e_2, ..., e_n\}$ such that $w(e_1) \geq w(e_2) \geq,,, \geq w(e_n) \geq 0$
$X \leftarrow \emptyset$
for i=1,...,n do
    if $(X + e_i) \in \mathcal{I}$ then $X \leftarrow X + e_i$
output X
```

These two oracles are equivalent in the sense that one can be recovered from the other in polynomial time. Given one of these two oracles, we have the greedy algorithm **Algorithm 1** that computes the max-weight independent set:

This algorithm is often called Kruskal's algorithm because it is exactly the Kruskal's algorithm in the case of Graphical Matroids for $w \geq 0$ (max-weight spanning tree).

Theorem 2.2.1. *The Greedy Algorithm correctly solves the Maximum Weight Independent Set problem.*

Proof. Without loss of generality, we consider the case of nonnegative weights, as it holds for the matroid after removing the elements with negative weight (the optimal solution does not contain them due to the second matroid property). Let $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ be the matroid that we have at the second step of the algorithm and the weight function on it $w \geq 0$. Clearly the optimal solution is a base. Call an independent set Y greedy if it is contained in a maximum-weight basis. It suffices to show that if Y is greedy, and x is an element in $\mathcal{S} \setminus Y$ such that $Y + x \in \mathcal{I}$ and such that w(x) is as large as possible, then Y+x is greedy. As Y is greedy, there exists a maximum-weight basis $B \supseteq Y$. If $x \in B$ then Y+x is greedy again. If $x \notin B$, then there exists a basis $\hat{B}$ containing Y+x and contained in B+x (this basis is produced by repeatedly applying the third matroid property). $\hat{B} = B - \hat{x} + x$ for some $\hat{x} \in B \setminus Y$. As w(x) is chosen maximum, $w(x) \geq w(\hat{x})$. Hence $w(\hat{B}) \geq w(B)$, and therefore $\hat{B}$ is a maximum-weight basis. So Y+x is greedy. $\square$

We should also note that we can adapt the greedy algorithm to solve the maximum weight base problem by making all weights non-negative by adding a large constant to each of the weights. Thus max-weight base problem, and equivalently min-weight base problem can be solved (by taking the weights to be the negative of the costs).

2.3 Matroid Polytope

We begin by giving a linear programming formulation for the problem maximum-weight independent set problem. Let x_e denote the indicator variable for element e, with the intent that $x_e = 1$ if e is in the solution and 0 otherwise. We obtain the following linear programming relaxation $LP_{mat}(\mathcal{M})$ after relaxing the integrality constraints on the variables x. In the following we use the shorthand $x(T)$ for $\sum_{e \in T} x_e$ for any $T \subseteq \mathcal{S}$.

$$\begin{aligned} \text{maximize} \quad & \sum_e w_e x_e \\ \text{subject to} \quad & x(T) \le r(T), \quad \forall T \subseteq \mathcal{S} \\ & x_e \ge 0, \quad \forall e \in \mathcal{S} \end{aligned}$$

The constraints of the LP_{mat} are exponential in the size of the ground set. Thus, in order to solve it via the ellipsoid method, we want a separation oracle that takes a vector $y \in \mathbb{R}^{|\mathcal{S}|}$ and outputs either that y is a feasible solution of the LP or a constraint that is violated by y. First of all, we can easily test for non-negativity. To test the second group of constraints, let's define $f: 2^{\mathcal{S}} \to \mathbb{R}$ such that:

$$f(T) = r(T) - x(T)$$

So, for the separation oracle, it suffices to find a subset of $\mathcal{S}$ that minimizes f and check whether the minimum is non-negative. If it is not, it returns the constraint corresponding to the minimizer of f. Observe, that since r(T) is a submodular function and x(T) is a modular function, f is submodular. So, we can use an algorithm for minimizing a submodular function, see [37]. However, there is a more efficient algorithm for separating over this LP given by Cunningham, see [37] for details.

For a set $T \subseteq \mathcal{S}$, let $\chi(T)$ denote the characteristic vector in $\mathbb{R}^{|\mathcal{S}|}$ that has a 1 corresponding to each element $e \in T$ and 0 otherwise. Now, the constraints of the above LP form a polytope. What we are going to show in this section is that this polytope is integral, i.e, all its extreme points have integer coordinates. What is the implication of such a theorem? Let's take a feasible integral solution x of this LP. From the nonnegativity constraints and the constraints on the singletons we have that $x \in \{0,1\}^{|\mathcal{S}|}$. So, $x = \chi(A)$ for some $A \subseteq \mathcal{S}$. Since x is feasible, $|A| = x(A) \le r(A) \Rightarrow x(\mathcal{S}) = x(A) = r(A)$, so x is the characteristic vector of the independent set $A \in \mathcal{I}$. Notice that for every $A \in \mathcal{I}$ we can set the weights such that A is the **unique** max-weight independent set (set weight 1 for all its elements and -1 all other weights). Summarizing, we have that each extreme point is the characteristic vector of an independent set and each independent set can be the unique optimal integral solution for some weight function. Thus, the extreme points are exactly all the characteristic vectors of the independent sets of $\mathcal{S}$. In other words, the above inequalities form the convex hull of the indicator vectors of the independent sets of $\mathcal{M}$!

The Uncrossing Technique and Characterization of Extreme Points

Now, we analyze the extreme point solutions of the LP_{mat}. Recall that an extreme point solution is the unique solution defined by n linearly independent tight inequalities, where $n = |\mathcal{S}|$ is the number of variables in the linear program. There are exponentially many inequalities in the LP_{mat} and an extreme point solution may satisfy many inequalities as equalities. To analyze an extreme point solution, an important step is to find a "good" set of tight inequalities defining it. If there is an element e with $x_e = 0$, this element can be removed from the matroid without affecting the feasibility and the objective value. So henceforth assume every element has $x_e > 0$.

Given an extreme point solution x to LP_{mat} let $\mathcal{F} = \{T \subseteq \mathcal{S} \ : \ x(T) = r(T)\}$ be the set of tight constraints. We first show that $\mathcal{F}$ is closed under intersection and union.

Lemma 2.3.1. *If $U, V \in \mathcal{F}$, then both $U \cup V$ and $U \cap V$ are in $\mathcal{F}$. Furthermore,*

$$\chi(U) + \chi(V) = \chi(U \cup V) + \chi(U \cap V)$$

.

Proof.

$$r(U)+r(V) = x(U)+x(V) = x(U \cup V)+x(U \cap V) \leq r(U \cup V)+r(U \cap V) \leq r(U)+r(V)$$

The first equality is by the fact that $U, V \in F$. The second equality follows from the fact that x(T) is modular. The third inequality follows from the constraints of the $LP_{mat}(\mathcal{M})$. The last equality is because of the submodularity of the rank function r. □

Now, let's proceed with a set family with a very "friendly" structure:

Definition 18. (Chain) *A subset $\mathcal{L} \subseteq 2^{\mathcal{S}}$ is a chain if*

$$A \in \mathcal{L}, B \in \mathcal{L} \Rightarrow A \subseteq B \text{ or } B \subseteq A$$

We now present an uncrossing argument that, using lemma 2.3.1, shows that the linearly independent set of tight constraints can be chosen to form a chain (remember that all $x_e > 0$ without loss of generality). If A is a family of subsets of $\mathcal{S}$, we denote by $span(A)$ the vector space generated by the set of vectors $\{\chi(T) \mid T \in A\}$.

Lemma 2.3.2. *If $\mathcal{L}$ is a maximal chain subfamily of $\mathcal{F}$, then $span(\mathcal{L}) = span(\mathcal{F})$.*

Proof. Suppose, by way of contradiction, that $\mathcal{L}$ is a maximal chain of $\mathcal{F}$ but $span(\mathcal{L}) \subset span(\mathcal{F})$. For any $A \subseteq \mathcal{S}$ such that $A \notin \mathcal{L}$, define $intersect(A, \mathcal{L})$ to be the number of sets in $\mathcal{L}$ which intersect A, i.e. $intersect(A, \mathcal{L}) = |\{T \in \mathcal{L} \mid T \setminus A \neq \emptyset, A \setminus T \neq \emptyset\}|$. Since $span(\mathcal{L}) \subset span(\mathcal{F})$, there exists a set A with $\chi(A) \notin span(\mathcal{L})$. Choose the one with the minimum $intersect(A, \mathcal{L})$. Since $\mathcal{L}$ is a maximal chain, we distinguish two cases for A. First, let's suppose that $A \cap C = \emptyset, \forall C \in \mathcal{L}$. Let C be the inclusion-wise maximum set of $\mathcal{L}$. Observe that using lemma 2.3.1 we have that $C \cup A \in \mathcal{F}$ and since $A \neq \emptyset$ ($\chi(A) \notin span(\mathcal{L})$) we have that $C \cup A \supset A$. Thus, the chain is not maximal, contradiction. The other case is that $intersect(A, \mathcal{L}) \geq 1$. Let T be a set in $\mathcal{L}$ which intersects A: $T \backslash A \neq \emptyset$, $A \backslash T \neq \emptyset$. Since $A, T \in \mathcal{F}$, by Lemma 2.3.1, both $A \cap T$ and $A \cup T$ are in $\mathcal{F}$. Also, both $intersect(A \cap T, \mathcal{L})$ and $intersect(A \cup T, \mathcal{L})$ are smaller than $intersect(A, \mathcal{L})$, which will be proved next in Proposition 2.3.2.1. Hence, by the minimality of $intersect(A, \mathcal{L})$, both $A \cap T$ and $A \cup T$ are in $span(\mathcal{L})$. By Lemma 2.3.1, $\chi(A) + \chi(T) = \chi(A \cup T) + \chi(A \cap T)$. Since $\chi(A \cup T) + \chi(A \cap T)$ are in $span(\mathcal{L})$ and $T \in \mathcal{L}$, the above equation implies that $\chi(A) \in span(\mathcal{L})$, a contradiction. It remains to prove Proposition 2.3.2.1 □

Proposition 2.3.2.1. *Let A be a set that intersects $T \in \mathcal{L}$. Then $intersect(A \cap T, \mathcal{L})$ and $intersect(A \cup T, \mathcal{L})$ are smaller than $intersect(A, \mathcal{L})$.*

Proof. Since $\mathcal{L}$ is a chain, for a set $R \in \mathcal{L}$ with $R \neq T$, R does not intersect T. So, whenever R intersects $A \cap T$ or $A \cup T$, R also intersects A. Also, T intersects A but not $A \cap T$ or $A \cup T$. Therefore, $intersect(A \cup T, \mathcal{L})$ and $intersect(A \cap T, \mathcal{L})$ are smaller than $intersect(A, \mathcal{L})$. □

This completes the proof of Lemma 2.3.2. Based on this and the Rank Lemma we can now proceed to the characterization of the extreme points of LP_{mat}.

Lemma 2.3.3. *Let x be any extreme point solution to $LP_{mat}(\mathcal{M})$ with $x_e > 0$ for each element $e \in \mathcal{S}$. Then there exists a chain $\mathcal{L}$ such that*

1. *$x(T) = r(T)$ for each $T \subseteq \mathcal{L}$*
2. *The vectors in $\{\chi(T) \ : \ T \in \mathcal{L}\}$ are linearly independent.*
3. *$|\mathcal{L}| = |\mathcal{S}|$*

Now that we have this structural lemma, we can prove the main theorem of this section:

Theorem 2.3.4. *The optimal solution of the $LP_{mat}(\mathcal{M})$ is integral.*

Proof. Let x be the optimal solution. As we have already mentioned, if we remove from $\mathcal{M}$ all the elements with $x_e = 0$ and take the LP_{mat} for the new matroid, the optimal solution at the remaining variables will stay invariant and it is easy to see that it will be an extreme point. So if we prove the integrality for the new LP, we proved it for the initial. Thus, without loss of generality, we can assume that $x_e > 0$, $\forall e \in \mathcal{S}$.

Let's suppose that there exists an element $e^* \in \mathcal{S}$ such that $x_{e^*} \in (0,1)$ (observe that the constraints on singletons impose $x_e \leq 1$ for all the elements e). Now, we proceed with a "token argument". We assign one token for each element $e \in \mathcal{S}$, for a total of $|\mathcal{S}|$ tokens. We will redistribute the tokens so that each set in $\mathcal{L}$ will receive one token and there are some extra token left. This implies that $|\mathcal{L}| < |\mathcal{S}|$, contradicting lemma 2.3.3.

To redistribute the tokens, each element gives its token to the smallest set of the chain $\mathcal{L}$ that contains it. Let's take two consecutive sets of the chain: $A \subseteq B$. Their constraints are tight, so:

$$x(A) = r(A) \ and \ x(B) = r(B) \ \Rightarrow x(B) - x(A) = r(B) - r(A)$$

But, $r(B) - r(A)$ is an integer and $x(B) - x(A) \geq 0$. Also, if $x(B) - x(A) = 0$, since $A \subseteq B$, we have that A=B ($x_e > 0 \ for \ all \ e$), which is impossible because of the linear independence. Hence $x(A) - x(B) \geq 1$. Also, if C is the inclusion-wise minimum set of the chain, $C \neq \emptyset$ because of the linear independence and thus $x(C) = r(C) \geq 1$. So, each set of the chain $\mathcal{L}$ gets at least one token. Let's focus on the set that takes the token of e*. If there is no such set we are done. Otherwise, since $0 < x_{e*} < 1$, this set receives at least two tokens. □

This shows that the $LP_{mat}(\mathcal{M})$ is an exact formulation of the max-weight independent set problem.

Theorem 2.3.5. *The extreme point solutions of $LP_{mat}(\mathcal{M})$ are the independent sets of matroid $\mathcal{M}$.*

2.4 Matroid Intersection

Given matroids $\mathcal{M}_1 = (\mathcal{S}, \mathcal{I}_1)$ and $\mathcal{M}_2 = (\mathcal{S}, \mathcal{I}_2)$ and a weight function $w : S \rightarrow R$, the maximum weight two matroid intersection problem is to find a set $T \subseteq \mathcal{S}$ of maximum weight which is independent in both $\mathcal{M}_1$ and $\mathcal{M}_2$, i.e, T is a maximizer of

$$max_{T \subseteq S,\ T \in \mathcal{I}_1 \cap \mathcal{I}_2} w(T)$$

where $w(T) = \sum_{e \in T} w_e$. We refer to the two matroid intersection problem as matroid intersection. This problem generalizes many important problems, including the maximum weight matching in bipartite graphs and maximum weight arborescence problem.

Examples of Matroid Intersection

1. **Matchings in Bipartite graph**: Given a bipartite graph $G = (A \cup B, E)$, let $\mathcal{M}_A = (E, \mathcal{I}_1)$ be a partition matroid on E where $\mathcal{I}_1 = \{F \subseteq E \mid d_F(v) \leq 1,\ \forall v \in A\}$. Similarly, let $\mathcal{M}_B = (E, \mathcal{I}_2)$ be a partition matroid on E where $\mathcal{I}_2 = \{F \subseteq E \mid d_F(v) \leq 1,\ \forall v \in B\}$. Observe that $T \in \mathcal{I}_1 \cap \mathcal{I}_2$ if and only if T is a matching in G. Hence, finding a maximum weight matching in G is equivalent to finding a maximum weight independent set in the intersection of matroids $\mathcal{M}_A$ and $\mathcal{M}_B$.

2. **Arborescence**: Given a directed graph $D = (V, A)$ and a root vertex $r \in V$, an r-arborescence is a subgraph of D so that there is a directed path from r to every vertex in $V \setminus \{r\}$. The minimum arborescence problem is to find an r-arborescence with minimum total cost. Let $\mathcal{M}_1 = (A, \mathcal{I}_1)$ be the graphic matroid on the underlying undirected graph of D (where we ignore arc directions). Let $\mathcal{M}_2 = (A, \mathcal{I}_2)$ be the partition matroid where $\mathcal{I}_2 = \{B \subseteq A :\ d^{in}_B(v) \leq 1,\ \forall v \in D \setminus \{r\}\ and\ d^{in}_B(r) = 0\}$. Observe that B is a common basis in $\mathcal{I}_1$ and $\mathcal{I}_2$ if and only if B is an arborescence rooted at r.

For the Matroid Intersection problem, there exists a polynomial time algorithm that computes the max-weight common independent set. However, for the purposes of this thesis, we will present an LP-based algorithm.

Linear Programming Relaxation

We now give a linear programming formulation for finding a maximum weight common independent set in the intersection of two matroids. Let x_e denote the indicator variable for element, with $x_e = 1$ if e is in the common independent set and 0 otherwise. We obtain the following linear programming relaxation $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ after relaxing the integrality constraints on the variables x. Here $r_i(T)$ denotes the rank of the set T in the matroid $\mathcal{M}_i$.

$$\begin{aligned}
\text{maximize} \quad & \sum_e w_e x_e \\
\text{subject to} \quad & x(T) \leq r_1(T), \quad \forall T \subseteq \mathcal{S} \\
& x(T) \leq r_2(T), \quad \forall T \subseteq \mathcal{S} \\
& x_e \geq 0, \quad \forall e \in \mathcal{S}
\end{aligned}$$

Solving the linear program. To get a separation oracle can be implemented if we are given as input independence oracles for each of the matroids $\mathcal{M}_1$ and $\mathcal{M}_2$, by using the work of Cunningham [37] as before, or any algorithm for minimizing submodular functions.

Characterization of Extreme Point Solutions

We now give a characterization of extreme points of the linear program $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ by showing that the independent set of tight constraints can be chosen to form a union of two chains. The proof is quite straightforward and uses the characterization of tight inequalities for the max-weight independent set problem.

Given an extreme point solution x to $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ let $\mathcal{F}_1 = \{T \subseteq \mathcal{S} \ : \ x(T) = r_1(T)\}$ and $\mathcal{F}_2 = \{T \subseteq \mathcal{S} \ : \ x(T) = r_2(T)\}$ be the set of tight constraints.

Lemma 2.4.1. *There exist two chains C_1 and C_2 such that $span(C_1 \cup C_2) = span(\mathcal{F}_1 \cup \mathcal{F}_2)$ and constraints in sets C_1 and C_2 are linearly independent.*

Proof. Applying Lemma 2.3.2 to families $\mathcal{F}_1$ and $\mathcal{F}_2$ separately, we obtain two chains $\hat{C}_1$ and $\hat{C}_2$ such that $span(\hat{C}_1) = span(\mathcal{F}_1)$ and $span(\hat{C}_2) = span(\mathcal{F}_2)$. Now, picking a maximal independent family from $\hat{C}_1 \cup \hat{C}_2$ gives us the desired chains. □

Thus, from the Rank Lemma:

Lemma 2.4.2. *Let x be any extreme point solution to $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ with $x_e > 0$ for each element $e \in \mathcal{S}$. Then there exist two chains C_1 and C_2 such that*

1. *$x(T) = r_i(T)$ for each $T \subseteq C_i$ for $i = \{1, 2\}$.*
2. *The vectors in $\{\chi(T) \ : \ T \in C_1\} \cup \{\chi(T) \ : \ T \in C_2\}$ are linearly independent.*
3. *$|C_1| + |C_2| = |\S_e|$.*

Iterative Algorithm

We now give an iterative algorithm which constructs an integral solution from the linear program and shows that the linear programming formulation is integral.

Algorithm 3 Iterative Matroid Intersection Algorithm

```
  $I \leftarrow \emptyset$.
  while $\mathcal{S} \neq \emptyset$ do
      Find an optimal extreme point solution x to $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$. Delete from both matroids every element $e \in \mathcal{S}$ with $x_e = 0$.
      If there is an element e with $x_e = 1$, then update $I \leftarrow I \cup \{e\}$, $\mathcal{M}_1 \leftarrow \mathcal{M}_1/e, \mathcal{M}_2 \leftarrow \mathcal{M}_2/e$.
  return I.
```

Correctness and Optimality

First of all, we show that the algorithm will terminate:

Lemma 2.4.3. *For any extreme point solution x to $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ with $x_e > 0$ for every element e, there exists an element e with $x_e = 1$.*

Proof. Suppose for a contradiction $0 < x_e < 1$ for each $e \in \mathcal{S}$. Then the number of variables is exactly $|\mathcal{S}|$. By Lemma 2.4.2, we obtain two chains C_1, C_2 defing x. We now show a contradiction to the fact that $|\mathcal{S}| = |C_1| + |C_2|$ by a counting argument. We give two tokens to each element in $\mathcal{S}$ for a total of $2|\mathcal{S}|$ tokens. Now, we collect two tokens for each member of C_1, C_2 and an extra token showing the contradiction. This is done as follows. Each element e assigns one token to the smallest set $T_i \in C_i$ such that $e \in Ti$ for $i = \{1, 2\}$. We now claim that each set in $C_1 \cup C_2$ obtains at least two tokens. The argument is identical for sets in C_1, C_2. Let $T \in C_1$ and R be the largest set in C_1 such that $R \subseteq T$. Now, we have $x(T) = r_1(T)$ and $x(R) = r_1(R)$. Subtracting, we obtain $x(T \setminus R) = r_1(T) - r_1(R)$. If $T \setminus R = \emptyset$ then T = R and we have a contradiction to the linear independence of the constraints. Also, since $x(T \setminus R)$ is an integer and $0 < x_e < 1$ for all e, we have that $|T \setminus R| \geq 2$. Thus, T receives one token for each element in $T \setminus R$ for a total of at least two tokens. Therefore, every set in $C_1 \cup C_2$ receives at least two tokens. Now, we show that there is at least one extra token. First of all, if an element does not belong to any set of a specific chain, then its corresponding token goes nowhere and we are done. So, it must be the case that the maximal element of each chain contains all the elements, which means that $\mathcal{S}$ belongs to both, contradiction because of the linear independence. □

Theorem 2.4.4. *The optimal solution of the $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ is integral.*

Proof. This is proved by induction on the number of iterations of the algorithm. The base case is trivial to verify. Let $\mathcal{M}_1 = (\mathcal{S}, \mathcal{I}_1)$, $\mathcal{M}_2 = (\mathcal{S}, \mathcal{I}_2)$ denote the matroids in the current iteration and x the optimal LP solution. If the algorithm finds an element e with $x_e = 0$ we remove e from both matroids. Observe that x restricted to $\mathcal{S} - e$, say x', is a feasible solution to $LP_{int}(\mathcal{M}_1 - e, \mathcal{M}_2 - e)$. This is easily checked using the rank function of $\mathcal{M}_i - e$ which is identical to the rank function $of M_i$ on the sets not containing e, for

$i = 1, 2$. By induction, we find a common independent set I of $\mathcal{M}_1 - e$, $\mathcal{M}_2 - e$ of weight at least $\sum_{e' \in \mathcal{S}-e} w(e')x'(e')$. Observe that I is also a common independent set of $\mathcal{M}_1$, $\mathcal{M}_2$ and costs at least $\sum_{e' \in \mathcal{S}-e} w(e')x'(e') = \sum_{e \in \mathcal{S}} w(e)x(e)$. Hence, the induction claim is true in this case.

Now, suppose the algorithm selects an element e with $x_e = 1$. Then the algorithm updates the matroids $\mathcal{M}_1, \mathcal{M}_2$ to $\mathcal{M}_1/e$, $\mathcal{M}_2/e$ and I to $I + e$. Let r_1 denote the rank function of $\mathcal{M}_1$ and r_1' denote the rank function of $\mathcal{M}_1/e$. We now claim that x restricted to $\mathcal{S} - e$, say x', is a feasible solution to $LP_{mat}(\mathcal{M}_1/e)$. For any set $T \subseteq \mathcal{S} - e$, we have $x'(T) = x(T + e) - x_e = x(T + e) - 1 \leq r_1(T + e) - 1 = r_1'(T)$. With exactly the same argument we have that x restricted to $\mathcal{S} - e$, say x', is a feasible solution to $LP_{mat}(\mathcal{M}_2/e)$. Thus, it is a feasible solution to $LP_{int}(\mathcal{M}_1/e, \mathcal{M}_2/e)$. By the induction hypothesis, we obtain an independent set I' of $\mathcal{M}/e$ of weight at least $w \cdot x'$. Then $I' + e$ is a common independent set of $\mathcal{M}_1$, $\mathcal{M}_2$ of weight at least $w \cdot x' + w_e = w \cdot x$ as required. This shows that the algorithm returns a maximum weight common independent set of $\mathcal{M}_1$, $\mathcal{M}_2$. □

It is easy to see that the above theorem shows that the $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ is an exact formulation of the maximum-weight matroid intersection problem.

Theorem 2.4.5. *The extreme point solutions of $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ correspond to independent sets in the intersection of $\mathcal{M}_1$ and $\mathcal{M}_2$.*

2.5 k Matroid Intersection via Iterative Rounding

Given k matroids $\mathcal{M}_1 = (\mathcal{S}, \mathcal{I}_1)$, $\mathcal{M}_2 = (\mathcal{S}, \mathcal{I}_2), ..., \mathcal{M}_k = (\mathcal{S}, \mathcal{I}_k)$ on the same ground set $\mathcal{S}$, the maximum k matroid intersection problem is to find a set $T \subseteq \mathcal{S}$ of maximum cardinality which is independent in all matroids $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k$. If someones tries to extend the integrality proof for three matroids will fail and there is a reason behind this:

Theorem 2.5.1. *Three Matroid Intersection is NP-hard.*

Proof. We use a reduction from the Hamiltonian path problem in directed graphs. Given a directed graph G with n vertices, and specified nodes s and t, the Hamiltonian path problem is the problem of determining whether there exists a simple path of length n-1 that starts at s and ends at t. It may be assumed without loss of generality that s has no incoming edges and t has no outgoing edges. Then, a Hamiltonian path exists if and only if there is a set of n-1 elements in the intersection of three matroids on the edge set of the graph: two partition matroids ensuring that the in-degree and out-degree of the selected edge set are both at most one, and the graphic matroid of the undirected graph formed by forgetting the edge orientations in G, ensuring that the selected edge set has no cycles. □

So, it is reasonable to seek for approximation algorithms for th k Matroid Intersection problem. We will present a 2-approximation algorithm for this problem when k=3, which is easily generalized to a $(k-1)-$approximation algorithm for the k Matroid Intersection problem. For now, we have k=3. The technique that we use is called "Iterative Rounding" and was introduced by Jain in [25] describing a 2-approximation algorithm for a large class of minimum-cost network design problems in undirected networks. For a very detailed and compact presentation of the power of this method see [31]. The general idea, applied to our framework is the following: using the structure of the optimal extreme point, prove that there is a variable with value greater or equal to 1/2 add that element to the solution, remove it from the matroid and recurse on the residual problem.

Linear Programming Relaxation

The linear programming relaxation, denoted by $LP_{3int}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$, for three-matroid intersection is a natural extension of $LP_{int}(\mathcal{M}_1, \mathcal{M}_2)$ for two-matroid intersection. Notice that we only consider the unweighted problem where $w_e = 1$ for all $e \in \mathcal{S}$.

$$\begin{array}{ll} \text{maximize} & \sum_e x_e \\ \text{subject to} & x(T) \le r_1(T), \quad \forall T \subseteq \mathcal{S} \\ & x(T) \le r_2(T), \quad \forall T \subseteq \mathcal{S} \\ & x(T) \le r_3(T), \quad \forall T \subseteq \mathcal{S} \\ & x_e \ge 0, \quad \forall e \in \mathcal{S} \end{array}$$

There is an efficient separation oracle for this exponential-size linear program, as in the case for two-matroid intersection. As for the structure of the extreme point solutions its proof follows the same lines as the proof of Lemma 2.4.2 for two-matroid intersection.

Characterization of Extreme Point Solutions

Lemma 2.5.2. *Let x be any extreme point solution to $LP_{3int}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$ with $x_e > 0$ for each element $e \in \mathcal{S}$. Then there exist three chains C_1, C_2, C_3 such that*

1. *$x(T) = r_i(T)$ for each $T \subseteq C_i$ for $i = \{1, 2, 3\}$.*
2. *The vectors in $\{\chi(T) : T \in C_1\} \cup \{\chi(T) : T \in C_2\} \cup \{\chi(T) : T \in C_3\}$ are linearly independent.*
3. *$|C_1| + |C_2| + |C_3| = |\mathcal{S}|$.*

Iterative Algorithm

Algorithm 4 Iterative Three Matroid Intersection Algorithm

```
$I \leftarrow \emptyset$.
while $\mathcal{S} \neq \emptyset$ do
    Find an optimal extreme point solution x to $LP_{3int}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$. Delete from all three matroids every element $e \in \mathcal{S}$ with $x_e = 0$.
    If there is an element e with $x_e \geq 1/2$, then update $I \leftarrow I \cup \{e\}$, $\mathcal{M}_1 \leftarrow \mathcal{M}_1/e, \mathcal{M}_2 \leftarrow \mathcal{M}_2/e, \mathcal{M}_3 \leftarrow \mathcal{M}_3/e$.
return I.
```

Correctness and Performance Guarantee

We first show that the iterative algorithm makes progress in each iteration. We then show that the algorithm returns a 2-approximate solution assuming it makes progress in each step.

Lemma 2.5.3. *For any extreme point solution x to $LP_{3int}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$ with $x_e > 0$ for every element e, there exists an element e with $x_e \geq 1/2$.*

Proof. Suppose for a contradiction $0 < x_e < 1/2$ for each $e \in \mathcal{S}$. Then the number of variables is exactly $|\mathcal{S}|$. By Lemma 2.5.2, we obtain three chains C_1, C_2, C_3 defing x. We now show a contradiction to the fact that $|\mathcal{S}| = |C_1| + |C_2| + |C_3|$ by a counting argument. We give three tokens to each element in $\mathcal{S}$ for a total of $3|\mathcal{S}|$ tokens. Now, we collect three tokens for each member of C_1, C_2, C_3 and an extra token showing the contradiction. This is done as follows. Each element e assigns one token to the smallest set $T_i \in C_i$ such that $e \in Ti$ for $i = \{1, 2, 3\}$. We now claim that each set in $C_1 \cup C_2 \cup C_3$ obtains at least three tokens. The argument is identical for sets in C_1, C_2, C_3. Let $T \in C_1$ and R be the largest set in C_1 such that $R \subseteq T$. Now, we have $x(T) = r_1(T)$ and $x(R) = r_1(R)$. Subtracting, we obtain $x(T \setminus R) = r_1(T) - r_1(R)$. If $T \setminus R = \emptyset$ then T = R and we have a contradiction to the linear independence of the constraints. Also, since $x(T \setminus R)$ is an integer and $0 < x_e < 1/2$ for all e, we have that $|T \setminus R| \geq 3$. Thus, T receives one token for each element in $T \setminus R$ for a total of at least three tokens. Therefore, every set in $C_1 \cup C_2 \cup C_3$ receives at least three tokens. Now, we show that there is at least one extra token. First of all, if an element does not belong to any set of a specific chain, then its corresponding token goes nowhere and we are done. So, it must be the case that the maximal element of each chain contains all the elements, which means that $\mathcal{S}$ belongs to all three chains, contradiction because of the linear independence. □

Theorem 2.5.4. *The iterative algorithm returns a 2-approximate solution to the maximum three-matroid intersection problem in polynomial time.*

Proof. This is proved by induction on the number of iterations of the algorithm. The base case is trivial to verify. Let $\mathcal{M}_1 = (\mathcal{S}, \mathcal{I}_1)$, $\mathcal{M}_2 = (\mathcal{S}, \mathcal{I}_2)$, $\mathcal{M}_3 = (\mathcal{S}, \mathcal{I}_3)$ denote the matroids in the current iteration and x the optimal LP solution. If the algorithm finds an element e with $x_e = 0$ we remove e from all the matroids. Observe that x restricted to $\mathcal{S} - e$, say x', is a feasible solution to $LP_{3int}(\mathcal{M}_1 - e, \mathcal{M}_2 - e, \mathcal{M}_3 - e)$. This is easily checked using the rank function of $\mathcal{M}_i - e$ which is identical to the rank function $of M_i$ on the sets not containing e, for all i. By induction, we find a common independent set I of $\mathcal{M}_1 - e$, $\mathcal{M}_2 - e$, $\mathcal{M}_3 - e$ of weight at least $\sum_{e' \in \mathcal{S}-e} x'(e')$. Observe that I is also a common independent set of $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ and costs at least $x'(\mathcal{S} - e) = x(\mathcal{S})$. Hence, the induction claim is true in this case.

We focus on the case when the algorithm selects an element e with $x_e \geq 1/2$. In this case the algorithm updates the matroid $\mathcal{M}_i$ to $\mathcal{M}_i/e$ and I to $I + e$. Let w(x) be the objective value of the solution x in the current iteration. To prove the performance guarantee, it suffices to prove that there is a feasible solution in the next iteration with objective value at least $w(x) - 2$. Since we add one element to I and the objective value decreases by at most two, by a standard inductive argument we can prove that the returned independent set has size at least half the objective value of $LP_{3int}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$, and thus the theorem follows.

To prove the claim, we need to demonstrate a feasible solution in the next iteration with objective value at least $w(x)-2$, after we select the element e and update the matroids $\mathcal{M}_i$ to $\mathcal{M}_i/e$. Consider the solution x restricted to $\mathcal{S}-e$, denoted by x'. Note that x' has objective value $w(x) - x_e$, but it may not be a feasible solution to $LP_{3int}(\mathcal{M}_1/e, \mathcal{M}_2/e, \mathcal{M}_3/e)$, the linear program in the next iteration. In the next paragraph we will show how to modify x' to satisfy all the constraints defined by matroid M_i/e, by decreasing the objective value by at most $1-x_e$. By performing this modification to each of the three matroids, we will have a feasible solution to $LP_{3int}(\mathcal{M}_1/e, \mathcal{M}_2/e, \mathcal{M}_3/e)$ with objective value at least $w(x) - x_e - 3(1 - x_e) = w(x) - 3 + 2x_e \geq w(x) - 2$ since $x_e \geq 1/2$, as desired.

It remains to show how to modify the solution x' to satisfy all the constraints defined by M_i/e, while decreasing the objective value by at most $1 - x_e$. Since x is a feasible solution to $LP_{3int}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$, it is obviously a feasible solution to $LP_{mat}(\mathcal{M}_i)$, the independent set polytope of matroid $\mathcal{M}_i$. Since the independent set polytope of a matroid is integral, the solution x can be written as a convex combination of independent sets in $\mathcal{M}_i$, i.e. $x = \sum_{j=1}^{N} \lambda_j \chi(I_j)$ for some N where $\lambda_j \geq 0$, $\forall j$, $\sum_{j=1}^{N} \lambda_j = 1$ and I_j is an independent set of $\mathcal{M}_i$ for each j. Assume that $e \notin I_j$ for $j = 1, ..., N'$ and $e \in I_j$ for $N' < j \leq N$. Then by definition $\sum_{j=1}^{N'} \lambda_j = 1 - x_e$. For each $1 \leq j \leq N'$, let $f_j \neq e$ be an element in the unique circuit (if exists) in $I_j + e$. Since $I_j - e + f_j$ is an independent set in $\mathcal{M}_i$, it follows by definition that $I_j - f_j$ is an independent set in $\mathcal{M}_i/e$. Similarly, $I_j - e$ is an independent set in M_i/e for $N' < j \leq N$. Thus

$$x^* = \lambda_1 \chi(I_1 - f_1) + ... + \lambda_{N'} \chi(I_{N'} - f_{N'}) + \lambda_{N'+1} \chi(I_{N'+1} - e) + ... + \lambda_N \chi(I_N - e)$$

is a feasible solution to $LPmat(\mathcal{M}_i/e)$, since it is a convex combination of independent sets in M_i/e. Furthermore $w(x^*) \geq w(x') - \sum_{j=1}^{N'} \lambda_j = w(x') - (1 - x_e)$, proving the theorem. $\square$

Chapter 3

Multistage Matroid Maintainance

3.1 Preliminaries

Having built the required background on matroids, linear programming and on basic rounding techniques, we can now formally define the *Multistage Matroid Maintainance* (MMM) problem. The results we discuss in this part were presented in the work of Gupta et al. "Changing bases: Multistage optimization for matroids and matchings" [22].

Definition 19. *An instance of the Multistage Matroid Maintenance (MMM) problem consists of a matroid $\mathscr{M} = (E, \mathcal{I})$, with $r(E) = r$, an acquisition cost $a(e) \geq 0$ for each $e \in E$, and for every time step $t \in [T]$ and element $e \in E$, a holding cost $c_t(e) \geq 0$. The goal is to find bases $\{B_t \in \mathcal{I}\}_{t \in [T]}$ to minimize*

$$\sum_t (c_t(B_t) + a(B_t \setminus B_{t-1}))$$

where we define $B_0 := \emptyset$.

In the case of T=1, optimizing over the bases of the matroid is equivalent to optimizing over its spanning sets. Indeed, if the optimal solution is a spanning set, we can start removing appropriate elements, without reducing the rank, until we end up with a basis, which will not have greater cost than the initial. However, does this equivalence hold for general T? We will show that it does. But first, lets define the *Multistage Spanning set Maintenance* (MSM) problem.

Definition 20. *An instance of the Multistage Spanning set Maintenance (MSM) problem consists of a matroid $\mathscr{M} = (E, \mathcal{I})$, $r(E) = r$, an acquisition cost $a(e) \geq 0$ for each $e \in E$, and for every time step $t \in [T]$ and element $e \in E$, a holding cost $c_t(e) \geq 0$. The goal is to find spanning sets $\{S_t \subseteq E\}_{t \in [T]}$ to minimize*

$$\sum_t (c_t(S_t) + a(S_t \setminus S_{t-1})) \quad \textbf{\textit{(1)}}$$

where we define $S_0 := \emptyset$.

The following lemma shows the equivalence of maintaining bases and spanning sets. It is proven in [22] and here we provide an alternative proof.

Lemma 3.1.1. *Every feasible solution for MSM can be transformed, in polynomial time, into a feasible solution for MMM, without increasing the total cost.*

Proof. Let $S_1, ...S_T$ be a feasible solution for MSM. If all these sets are bases then we are done. Otherwise, let i be the minimum moment such that S_i is not a base. Let C be a circuit in S_i. Since $C \nsubseteq S_{i-1}$ (remember that $S_0 := \emptyset$ and the definition of i), there is an element $e \in C \setminus S_{i-1}$. We remove e from S_i and S_i remains a spanning set. All these procedures can be implemented in polynomial time (see chapter 2). The holding cost won't increase, since $c_t(e) \geq 0$. The acquisition cost won't increase since $e \in S_i \setminus S_{i-1}$ and $a(e) \geq 0$. Thus, the new solution has lower or equal cost and is feasible for MSM. We iterate on the above process until all S_i become bases. Since in each iteration, $\sum_t |S_t|$ decreases by 1, the algorithm will terminate in polynomial time. ☐

Corollary 3.1.1.1. *For matroids, the optimal solutions to MSM and MMM have the same costs.*

3.2 The Greedy Algorithm

Our first attempt to solve the problem is to extend Kruskal's algorithm, which provides the optimal solution for T=1. For this reason, we use a result of Wolsey.

Theorem. *(**Wolsey** [46]) We consider the problem $min\{\sum_{j \in S} w_j : f(S) = f(N), S \subseteq N\}$, where f is a nondecreasing submodular function on a finite set N. When f is integer valued and $f(\emptyset) = 0$, the greedy heuristic solution has approximation ratio $H(max_{j \in N} z(\{j\}))$, where $H(k) = \sum_{i=1}^{k} \frac{1}{i}$.*

However, in order to apply the greedy algorithm to our setting, we need to appropriately define the ground set, the weight function and the submodular function, in order to incorporate the acquisition costs to the weights. The idea is that we will no longer choose an element for a specific time step, but we will choose an edge for a specific interval that we want the element to be alive. More specifically, we make the following reduction: We will have T matroids $\mathscr{M}_1 = (E_{int}, \mathcal{I}_1), M_2 = (E_{int}, \mathcal{I}_2), ..., M_T = (E_{int}, \mathcal{I}_T)$. The common ground set will be

$$E_{int} = \{(e, [l, r]) \mid e \in E,\ 1 \leq l \leq r \leq T\}$$

the element x=(e, [l,r]) represents that the element e is alive for the interval $I_x = [l, r]$. For an element x, when we will want to refer to e we will write x.e and for the interval we will write I_x. The weight function will be

$$w((e, [l, r])) = a(e) + \sum_{t=l}^{r} c_t(e)$$

since we purchase the element e at time l and we keep it for the time interval [l,r]. As for the independent sets:

$$S \in \mathcal{I}_t \Leftrightarrow \forall x \in S: \ t \in I_x, \ \{x.e \mid x \in S\} \in \mathcal{I} \ and \ \nexists \ x_1, x_2 \in S: \ x_1 \neq x_2, \ x_1.e = x_2.e$$

The nondecreasing submodular function is

$$f(S) = \sum_t r_t(S)$$

since the sum of nondrecreasing submodular functions (the ranks) is a nondrecreasing submodular function.

It is easy to observe that MSM is equivalent to

$$min\{w(S) \mid f(S) = f(E_{int}), \ S \subseteq E_{int}\}$$

Thus, from [46], the greedy algorithm described in chapter 2 is an $H(max_{x \in E_{int}}(f(\{x\}) - f(\emptyset))) = H(T) = O(\log T)-$approximation algorithm. In [22] Gupta et al. provide an alternative dual fitting proof from scratch. We will present the analysis, but first lets see what the greedy algorithm does at our problem:

We consider the interval view of the problem. Given a current subset $A \subseteq E_{int}$, the benefit of adding an element x to A is

$$ben_A(x) = \sum_t (r_t(A \cup \{x\}) - r_t(A)) = \sum_{t \in I_x} (r_t(A \cup \{x\}) - r_t(A))$$

Initially $A = \emptyset$ and the greedy algorithm iteratively picks an element $x \in E_{int} \setminus A$ maximizing $ben_A(x)/w(x)$ and adds x to A. This is done until $f(A) = f(E_{int}) = rT$, where $r = r_{\mathcal{M}}(E)$. In other words, at the end, A induces a spanning set for each time step.

We analyze the algorithm through dual fitting. For this reason, we introduce the LP_1 with the following variables:

- y_x, $x \in E_{int}$ indicating whether we take element x or not
- z_{xt}, $x \in E_{int}$, $t \in I_x$

$$\begin{array}{ll} \text{minimize} & \sum_{x \in E_{int}} w(x) y_x \\ \text{subject to} & \vec{z_t} \ \in \ \mathcal{P}_B(\mathcal{M}_{ext}), \quad \forall t \in [T] \\ & z_{xt} \leq y_x, \quad \forall x \in E_{int}, \quad \forall t \in I_x \\ & y_x \geq 0, \quad \forall x \in E_{int} \end{array}$$

where $\mathcal{M}_{ext}$ is the natural extension of the initial matroid $\mathcal{M}$ on the new ground set E_{int}, where all the elements x=(e,[l,r]) associated with the same e are parallel to each other. $P_B(\mathcal{M}_{ext})$ is the base polytope of $\mathcal{M}_{ext}$, which lies in $\mathbb{R}^{|E_{int}|}$. As for the coordinates of the vector $\vec{z_t}$:

$$z_t(x) = \begin{cases} z_{xt} & \text{if } t \in I_x \\ 0 & otherwise \end{cases}$$

Using Lagrangian variables $b_{xt} \geq 0$ for each x and $t \in I_x$, we write the LP_2 which outputs a lower bound for the LP_1:

$$\begin{array}{ll} \text{minimize} & \sum_{x \in E_{int}} w(x) y_x + \sum_{x, t \in I_x} b_{xt}(z_{xt} - y_x) \\ \text{subject to} & \vec{z_t} \ \in \ \mathcal{P}_B(\mathcal{M}_{ext}), \quad \forall t \in [T] \\ & y_x \geq 0, \quad \forall x \in E_{int} \end{array}$$

The objective function of LP_2 is equal to

$$\sum_{x \in E_{int}} (w(x) - \sum_{t \in I_x} b_{xt}) y_x + \sum_{x, t \in I_x} b_{xt} z_{xt}$$

The optimal value of the LP_2 is a lower bound for the optimal value of LP_1 for every choice of nonnegative b_{xt}. In particular, if we choose these Lagrangian variables to satisfy the constraints:

$$w(x) - \sum_{t \in I_x} b_{xt} \geq 0, \ \forall x \in E_{int}$$

then clearly, the LP_2 will set all y_x to zero. In this case, LP_2 takes the form:

$$\begin{array}{ll} \text{minimize} & \sum_{x, t \in I_x} b_{xt} z_{xt} \\ \text{subject to} & \vec{z_t} \ \in \ \mathcal{P}_B(\mathcal{M}_{ext}), \quad \forall t \in [T] \end{array}$$

but in this case

$$\sum_{x:\ t \in I_x} b_{xt} z_{xt} \ \ subject\ to\ \vec{z_t} \ \in \ \mathcal{P}_B(\mathcal{M}_{ext})$$

can be minimized independently for each t. Due to the integrality of the matroid base polytope, the optimal value of LP_2, under the aforementioned choice of b_{et} will be

$$\sum_t mwb(\{b_{xt}\}_{x:\ t \in I_x})$$

where for a fixed t, we remove from $\mathcal{M}_{ext}$ all these x that $t \notin I_x$, we put weight b_{xt} to the remaining x and $mwb(\{b_{xt}\}_{x:\ t \in I_x})$ is the minimum weight base. We choose the best lower bound, LP_3, for LP_1, which is:

$$\begin{array}{ll} \text{minimize} & \sum_t mwb(\{b_{xt}\}_{x:\ t \in I_x}) \\ \text{subject to} & \sum_{t \in I_x} b_{xt} \leq w(x), \forall x \in E_{int} \\ & b_{xt} \geq 0, \quad \forall x \in E_{int}, \quad \forall t \in I_x \end{array}$$

The analysis follows the dual fitting proofs of [13, 35]

Theorem 3.2.1. *The greedy algorithm outputs an $O(log|I_{max}|)-$approximation to MSM, where $|I_{max}|$ is the length of the longest interval that an element is alive for. Hence, it gives an $O(logT)-$approximation.*

Proof. First of all, the matroid that we study in this proof is $\mathcal{M}_{ext}$. For a subset A of E_{int}, we write $A^t = \{x \in A | \; t \in I_x\}$. For the proof, consider some point in the run of the greedy algorithm where a set $A \subseteq E_{int}$ of elements has been picked. We show a nonnegative setting of duals b_{xt} such that

- The objective value of LP_3 (dual value) equals the current objective value of LP_1 (primal value) w(A) and

- $\sum_{t \in I_x} b_{xt} \leq O(log|I_x|)w(x)$, $\forall x \in E_{int}$

It is useful to maintain, for each time t, a minimum weight base B_t of the subset $span(A^t) \cap E^t_{int}$ according to weights $\{b_{xt}\}_{x \in E^t_{int}}$. We start with $b_{xt} = 0$, $A_t = B_t = \emptyset$, for all t, which satisfies the above properties.

Suppose that until some step these properties hold and we have collected the set A. we now pick x maximizing $ben_A(x)/w(x)$ and get new set $C := A \cup \{x\}$. Call a time step t "interesting" if $r(C^t) = r(X_t) + 1$. There are $ben_A(x)$ interesting time steps. Now, we need to update the duals. For each interesting t and for each $y \in (span(C^t) \cap E^t_{int}) \setminus (span(A^t) \cap E^t_{int})$ update $b_{yt} \leftarrow w(x)/ben_A(x)$. Note that the element x itself satisfies the condition of being in $(span(C^t) \cap E^t_{int}) \setminus (span(A^t) \cap E^t_{int})$ for precisely the interesting time steps, and hence $\sum_{t:interesting} b_{xt} = ben_A(x)w(x)/ben_A(x) = w(x)$. In all the noninteresting time steps the min-weight bases remains invariant. In an interesting time step t, consider the min-weight base B_t (of the previous step). The elements in $(span(C^t) \cap E^t_{int}) \setminus (span(A^t) \cap E^t_{int})$ increased the rank by one, in time t, and they all got the same weight, which due to the greedy criterion have bigger b_{yt} than all the previous elements. Thus, the Kruskal's algorithm will choose the base $B'_t \leftarrow B_t + x$ for every interesting time step t. Thus, the first property inductively holds. It remains to show the second one:

Let's focus on an element $y \in E_{int}$. Initially all b_{yt} are zero and the greedy decides whether it will add y to the solution based on $w(y)/ben_A(y)$ for a current solution A and if it updates a b_{yt} then its value will be lower or equal to $w(y)/ben_A(y)$ and $ben_A(y)$ will be decreased. Thus, at the end of the algorithm:

$$\sum_{t \in I_y} b_{yt} \leq w(y)(\frac{1}{I_y} + \frac{1}{I_y - 1} + ... + 1) = O(log|I_y|)w(y), \;\; \forall y \in E_{int}$$

and each element can only be alive for all T time steps. From the standard dual fitting argument, we have that the greedy algorithm is an $O(logT)-$approximation algorithm. □

3.3 The Randomized Rounding Algorithm

In [22], Gupta et al. also present an $O(log(rT))-$randomized rounding approximation algorithm for the MMM. First of all, we write the natural extension of the classical Matroid LP:

$$\begin{aligned}
\text{minimize} \quad & \sum_{t,e} c_t(e)z_t(e) + \sum_{t,e} a(e)y_t(e) \\
\text{subject to} \quad & \vec{z_t} \in \mathcal{P}_B(\mathcal{M}), \quad \forall t \in [T] \\
& y_t(e) \geq z_t(e) - z_{t-1}(e), \quad \forall e \in E, \quad \forall t \in [T] \\
& y_t(e), z_t(e) \geq 0, \ \forall e \in E, \quad \forall t \in [T]
\end{aligned}$$

Whenever we mention an LP in this section, we will mean the above LP. Let OPT$= (\vec{y_t}, \vec{z_t})_{t\in[T]}$ be the optimal solution for the above LP. Observe that in the above constraints, $\vec{z_0}$ is included, but it is not given to the LP as a variable but as an identically zero vector. Also observe that $y_t(e) = max(z_t(e) - z_{t-1}(e), 0)$, $\forall t, e$.

Let's fix some t. Then, $\vec{z_t}$ induces probabilities of selection for each element is E. Based on these probabilities, we act like we did for the set cover in chapter 1. Initially $S_t = \emptyset$. We add to S_t, independently, each element e, with probability $z_t(e)$. However, if we do this, there is a considerable probability that the $r(S_t) < r$. Thus, we iterate this experiment and after a logarithmic number of steps, S_t will be a spanning set with high probability. However, this sampling procedure it is done **dependently** between consecutive time steps, in order to achieve low acquisition cost. This is achieved through **shared randomness**. At the end, if $S_1, ..., S_T$ are not spanning sets, then produce a solution greedily, which is at most T times the optimal. Since this happens with low probability, the expected cost is not large. Finally, from lemma 3.1.1, we can transform these T spanning sets into T bases, without increasing the total cost. The algorithm goes as follows:

Algorithm 5 Randomized Rounding

```
1: Solve the LP and get the $(\vec{y_t}, \vec{z_t})_{t\in[T]}$
2: $L = 8(2+\sqrt{3})ln(rT)$
3: For each $e \in E$ choose independent $\tau_e \sim U[0, 1/L]$
4: For each t, define $\hat{S_t} = \{e \in E \mid z_t(e) \geq \tau_e\}$
5: If all $\hat{S_t}$ have full rank, convert them to a solution for MMM at no extra cost and
   return $(\hat{S_1}, ..., \hat{S_T})$.
6: For each i, produce $S_i$ through Kruskal's algorithm, with weights $(a(e)+c_i(e))_{e\in E}$ and
   return $(S_1, ..., S_T)$.
```

Let $w(OPT)$ be the cost of the optimal solution of the LP and let $w(\mathcal{S})$ be the cost of the output of the algorithm (where $\mathcal{S}$ can be either $(\hat{S_1}, ..., \hat{S_T})$ or $(S_1, ..., S_T)$).

First of all, we must settle that, with high probability, $\hat{S_1}, ..., \hat{S_T}$ will have full rank.

Lemma 3.3.1. *For a fractional base $z \in \mathcal{P}_B(\mathcal{M})$, let $R(z)$ be the set obtained by picking each element $e \in E$ independently with probability z_e. Then $\mathbb{E}[r(R(z))] \geq r(1 - 1/e)$.*

Proof. We use the results of Chekuri et al. [12] on so-called contention resolution schemes. In their paper, for a matroid $\mathcal{M}$, they give a randomized procedure π_z that takes the random set $R(z)$ and outputs an independent set $\pi_z(R(z))$ in $\mathcal{M}$, such that $\pi_z(R(z)) \subseteq R(z)$, and for each element e in the support of z, $Pr[e \in \pi_z(R(z)) \mid e \in R(z)] \geq 1 - 1/e$. Thus, we get:

$$\mathbb{E}[r(R(z))] \geq \mathbb{E}[r(\pi_z(R(z)))] = \sum_{supp(z)} Pr[e \in \pi_z(R(z))] \geq$$

$$\sum_{supp(z)} Pr[e \in \pi_z(R(z)) \mid e \in R(z)] Pr[e \in R(z)] \geq \sum_{supp(z)} (1 - 1/e) z_e = r(1 - 1/e)$$

The first inequality used the fact that $\pi_z(R(z)) \subseteq R(z)$, the following equality used that $\pi_z(R(z))$ is independent with probability 1, the second inequality used the property of the CR scheme, and the final equality used the fact that z was a fractional base.

□

Corollary 3.3.1.1. *R(z) has rank at least r/2 with probability at least* $1 - 2/e > 1/4$

Proof. Apply reverse Markov inequality for the random variable $r(R(z)) \leq r$ □

Lemma 3.3.2. *For any fixed* $t \in [T]$*, the set* $\hat{S}_t$ *has full rank with probability at least* $1 - 1/(rT)^2$

Proof. The algorithm produces the set $\hat{S}_t$ by threshold rounding of the fractional base $z_t \in \mathcal{P}_{\mathcal{B}}(\mathcal{M})$. Instead, consider taking L different sets $T_1, ..., T_L$, where each set is produced independently, by including each element e independently with probability $z_t(e)$. The final set will be $T = \cup_{i=1}^{L} T_i$. We proceed with the following stochastic domination claim:

Claim 3.3.3. $Pr[r(\hat{S}_t) = r] \geq Pr[r(T) = r]$

Proof. It suffices to prove that for each $e \in E$, $Pr[e \in T] \leq Pr[e \in \hat{S}_t]$ and the claim will follow from the standard stochastic domination argument. He have that $Pr[e \notin T] = (1 - z_t(e))^L$. We also have that $Pr[e \notin \hat{S}_t] = max(0, \frac{\frac{1}{L} - z_t(e)}{\frac{1}{L}}) = max(0, 1 - Lz_t(e))$ and since $z_t(e) \in [0, 1]$, the inequality follows from Bernoulli's inequality. □

So, it suffices to give a lower bound on the probability that T has full rank. For this, we use Corollary 3.3.1.1: the set T_1 has rank at least r/2 with probability at least 1/4. Now, focusing on the matroid $\mathcal{M}' = \mathcal{M}/span(T_1)$ which say has rank r', the same argument says thst the set T_2 has rank r'/2 with probability at least 1/4 etc. Proceeding in this way, the probability that the rank of T is less than r is at most the probability that we see fewer than $log_2 r$ heads in $L = 8(2 + \sqrt{3}) ln(rT)$ flips of a coin of bias 1/4. We bound this probability using the following Chernoff bound:

Proposition 3.3.3.1. *Let* $X1, ..., X_n$ *be independent Bernoulli variables such that for each i,* $Pr[x_i] \geq p$*. Let* $X = \sum_{i=1}^{n} X_i$ *and* $mp = \mu \leq \mathbb{E}[X]$*. Then, for* $0 < \delta < 1$*:*

$$Pr[X \leq (1 - \delta)\mu] \leq e^{-\mu\delta^2/2}$$

In our case, $n = L = 8(2+\sqrt{3})ln(rT)$, $X_1, ..., X_L$ are the coin flips, $p = 1/4$, $\mu = L/4$. We set $\delta = 1 - \frac{1}{2+\sqrt{3}}$. So, by applying the Chernoff bound, we get:

$$Pr[X \leq \frac{1}{2+\sqrt{3}} 8(2+\sqrt{3})ln(rT)/4] = Pr[X \leq 2ln(rT)] \leq e^{-(1-\frac{1}{2+\sqrt{3}})^2 8(2+\sqrt{3})ln(rT)/8} = \frac{1}{(rT)^2}$$

But,

$$Pr[X \leq log_2 r] \leq Pr[X \leq 2ln(rT)] \leq \frac{1}{(rT)^2}$$

Thus, $Pr[r(\hat{S}_t) < r] \leq Pr[r(T) < r] \leq \frac{1}{(rT)^2}$ □

Now it is time to prove the main theorem:

Theorem 3.3.4. $\mathbb{E}[w(\mathcal{S})] \leq O(log(rT))w(OPT)$

Proof. First of all, by the union bound, the probability that all $\hat{S}_i$ have full rank is at least $1 - \frac{1}{Tr^2}$. Let $w(\hat{S}_1, ..., \hat{S}_T)$ be the total cost of $\hat{S}_1, ..., \hat{S}_T$. Notice that $\{S_i\}_{i \in [T]}$ may not be a feasible solution. However, these sets can be plugged into the objective function of MSM and the result is this cost. Let F be the event that all $\hat{S}_i$ have full rank and F^c its complement..

$$\mathbb{E}[w(\mathcal{S})] = \mathbb{E}[w(\hat{S}_1, ..., \hat{S}_T) \mid F]Pr[F] + w(S_1, ..., S_T)Pr[F^c] \leq= \mathbb{E}[w(\hat{S}_1, ..., \hat{S}_T)] + w(S_1, ..., S_T)\frac{1}{Tr^2}$$

It remains to bound $\mathbb{E}[w(\hat{S}_1, ..., \hat{S}_T)]$ and $w(S_1, ..., S_T)$.

- For a fixed t, since $Pr[e \in \hat{S}_t] = min\{Lz_t(e), 1\}$,

$$E[c_t(\hat{S}_t)] = \sum_e c_t(e)Pr[e \in \hat{S}_t] \leq L \sum_e c_t(e)z_t(e) \tag{3.1}$$

  Moreover, $e \in \hat{S}_t \setminus \hat{S}_{t-1}$ exactly when τ_e satisfies $z_{t-1}(e) < \tau_e < z_t(e)$, which happens with probability at most

$$\frac{max\{z_t(e) - z_{t-1}(e), 0\}}{\frac{1}{L}} \leq Ly_t(e).$$

  Thus,

$$E[a(\hat{S}_t \setminus \hat{S}_{t-1})] = \sum_e a(e)Pr[e \in \hat{S}_t \setminus \hat{S}_{t-1}] =\leq L \sum_e a(e)y_t(e) \tag{3.2}$$

  From 3.1, 3.2 and summing over all t, we get:

$$\mathbb{E}[w(\hat{S}_1, ..., \hat{S}_T)] \leq Lw(OPT)$$

- It is easy to see that $w(S_1, ..., S_T) \leq Tw(OPT)$. Indeed,

$$w(S_1, ..., S_T) \leq \sum_t (c_t(S_t) + a(S_t)) = \sum_t (c_t + a)(S_t)$$

because, by rebuying an element that we already have, we increase the cost. Now, let's fix a $t \in [T]$ and let's go to the initial LP and remove all the constraints that impose that $\vec{z}_t$ is a fractional base, except the one corresponding at time t. Since $y_t(e) = max(z_t(e) - z_{t-1}(e), 0)$ and $\mathcal{P}_{\mathcal{B}}(\mathcal{M})$ is integral, the optimal value of the objective function of this reduced LP will be exactly the outcome of the Kruskal's algorithm for the matroid $\mathcal{M}$ with weights given by the function $a + c_t$, which is exactly $(c_t + a)(S_t)$. This reduced LP, was produced by removing some constraints of the initial, thus the optimal value of the objective did not increase. So, $(c_t + a)(S_t) \leq w(OPT)$. Summing over all t we get the desired result.

Summarizing,

$$\mathbb{E}[w(\mathcal{S})] \leq \mathbb{E}[w(\hat{S_1}, ..., \hat{S_T})] + w(S_1, ..., S_T)\frac{1}{Tr^2} \leq Lw(OPT) + Tw(OPT)\frac{1}{Tr^2} = O(log(rT))w(OPT)$$

□

However, the dependence of T at the approximation ratio can be avoided. More specifically, by slightly modifying the randomized rounding algorithm, Gupta et al. in [22] achieve approximation ratio $O(\frac{a_{max}}{a_{min}} logr)$. Observe that in case where the acquisition costs are uniform, this yields an $O(logr)-$approximation algorithm. But, when this is not the case, can we avoid both the dependence on T and on $\frac{a_{max}}{a_{min}}$? Can we hope for a $O(logr)-$approximation algorithm for the MMM? If the algorithm is based on the LP of this section, the answer is no. In [22], the authors show that the $O(min\{logT, log\frac{a_{max}}{a_{min}}\})$ term in our rounding algorithm is unavoidable. This is a graphical matroid instance, with n (even) vertices and m edges. $logT$ and $log\frac{a_{max}}{a_{min}}$ are θn, m = θn^2 and the linear program has the aforementioned gap.

Lemma 3.3.5. *The LP has an $\Omega(min\{logT, log\frac{a_{max}}{a_{min}}\})$ integrality gap.*

This means that if the aspect ratio of the acquisition costs is not bounded, the linear program has a $logT$ gap, even when T is exponentially larger than r.

Proof. This is a graphical matroid instance, with n (even) vertices and m edges. $logT$ and $log\frac{a_{max}}{a_{min}}$ are θn, m = θn^2 and the linear program has the aforementioned gap. The set of vertices is $\{v_0, v_1, ..., v_n\}$ and T=$\binom{n}{\frac{n}{2}}$. The edges (v_0, v_i) for $i \in [n]$ have acquisition cost $a(v_0, v_i) = 1$ and holding cost $c_t(v_0, v_i) = 0$ for all t. The edges (v_i, v_j) for $i, j \in [n]$ have acquisition cost $\frac{1}{nT}$ and have holding cost determined as follows: we find a bijection between the set [T] and the set of partitions (U_t, V_t) of $\{v_1, ..., v_n\}$ with each of U_t and V_t having size $\frac{n}{2}$. In time step t, all edges inside U_t and V_t have holding cost 0 and all the edges in the set $E(U_t, V_t)$ have holding cost ∞.

First of all, a feasible integral solution has cost at least n/2+1. Indeed, let's suppose that there is a feasible solution, that uses at most n/2 edges of the type (v_0, v_i) (otherwise the acquisition cost is at least n/2+1). At some time step t, all these edges belong to the set $E(\{v_0\}, U_t) \cup E(\{v_0\}, V_t)$. Thus, since the solution cannot include, at time t, edges with holding cost ∞, $\delta(V_t)$ or $\delta(U_t)$ is empty at time t, contradiction. Thus, every feasible integral solution has cost $\Omega(n)$.

Finally, we show that on this instance, the LP, has a feasible solution of cost $O(1)$. We set $z_t(v_0, v_i) = 2/n$ for all $i \in [n]$ and $t \in [T]$. For all the other edges e, if at time step t they have zero holding cost, we set $z_t(e) = 4/n$.

Proposition 3.3.5.1. *z_t is in the spanning tree polytope for all $t \in [T]$.*

Proof. We will perform a random experiment which outputs a spanning tree for the time step t. Take uniformly at random a spanning tree T_{U_t} from the clique U_t, a spanning tree T_{V_t} from the clique V_t, one edge from $E(v_0, \cup_{i=1}^{\frac{n}{2}} v_i)$ and one edge from $E(v_0, \cup_{i=\frac{n}{2}+1}^{n} v_i)$. Now, let's see with what probability we take each edge to the random tree. An edge (v_0, v_i) is taken with probability $1/(n/2) = 2/n$ for each $i \in [n]$. Each of the others, by symmetry, are taken with probability $\frac{n/2-1}{\frac{n/2(n/2-1)}{2}} = 4/n$. So, there is a probability measure over the characteristic vectors of spanning trees for time t such that the expected vector is z_t. But, the expected vector is given by a convex combination, corresponding to this probability measure. The proposition follows. $\square$

Finally, the total acquisition cost is at most $\frac{2}{n}n + Tn^2\frac{4}{n}\frac{1}{nT} = O(1)$. The holding costs payed are zero. Thus the LP has a feasible fractional solution with total cost $O(1)$. The claim follows (remember that $log\binom{n}{\frac{n}{2}}=\Theta(n)$). $\square$

3.4 Hardness results

In the previous sections, we showed how to approximate the optimal solution of MSM and MMM up to a logarithmic factor. The algorithms used were quite similar with the corresponding ones for the set cover problem (the greedy and the randomized rounding). For the set cover, the logarithmic approximation ratio is optimal, unless $P = NP$, see [14]. Is this a coincidence? Maybe there is another more convoluted way to approximate sublogarithmically these two problems. The following theorem, proved in [22] shows that this is not possible, unless $P = NP$.

Theorem 3.4.1. *The MSM and MMM problems are $NP-hard$ to approximate better than $\Omega(min\{logr, logT\})$ even for graphical matroids.*

Proof. We give a reduction from Set Cover to the MSM problem for graphical matroids. Given an instance $(\mathcal{U}, \mathcal{F})$ of set cover, with $m = |\mathcal{F}|$ sets and $n = |\mathcal{U}|$ elements and $m = poly(n)$ (the set cover restricted to these instances cannot be approximated better than $\Omega(logn)$). We construct a graph as follows. There is a special vertex r, and m set vertices (with vertices s_i for each set $S_i \in \mathcal{F}$). There are m edges $e_i := (r, s_i)$ which all have acquisition cost $a(e_i) = 1$ and holding cost $c_t(e_i) = 0$, for all t. Taking these edges at some time step encodes whether we take a set to our solution or not. What remains to do is to force the optimal solution to obtain edges that encode a set cover. We have to do with spanning trees, so our only "weapon" is the connectivity requirement. More specifically, all other edges will be short-term and will have zero acquisition cost. There are T time steps. In time step $j \in [n]$, define subset $F_j := \{s_i \mid u_j \in S_i\}$ to be vertices corresponding to sets containing element $u_j \in \mathcal{U}$. In this specific time step, we have the edges that make F_i a clique and all edges (r, y) for $y \in \bar{F_j} := \{s_i \mid u_j \notin S_i\}$. All these edges have zero acquisition cost a(e), and are only alive at time j (which can be done by using infinite weights during the other time steps).

Now, let O be the cardinality of the optimal set cover. The cost of any solution for MSM (or MMM) is greater or equal to O. Indeed, in time step j, there must be an edge connecting r with some vetrex in F_j. At the same time, there is a solution to MSM (and to MMM) that has cost exactly O. Let $S_{i_1}, ..., S_{i_O}$ be the optimal set cover. At first, we maintain all edges (r, s_{i_k}), k=1,...,O, for all time steps, paying only acquisition cost O. We can now add edges for free and ensure connectivity at all time steps. In particular, at time step j we buy all the edges that are alive at this time step. The graph is connected at time j, because at this time, there exists an edge between r and a point in F_j, say s_{i_k}, all the other vertices in F_j are connected through the clique with s_{i_k} and all the vertices in $\bar{F_j}$ are connected with r through the other free edges. Thus, our reduction is strict. Finally, the number of time periods is T = n, and the rank of the matroid is m = poly(n) for these hard instances. This gives us the claimed hardness.

□

Someone may wonder why we do not discuss the generalized version of MMM, where the matroid changes over time. The reason is that this problem is really hard, as the following theorem, proved in [22] indicates.

Theorem 3.4.2. *The MMM problem with different matroids is NP-hard to approximate better than a factor of $\Omega(T)$, even for partition matroids and zero holding costs, as long as $T \geq 3$.*

Proof. The reduction is from 3D-Matching (3DM). An instance of 3DM has three sets X, Y, Z of equal size $|X| = |Y| = |Z| = k$, and a set of hyperedges $E \subseteq X \times Y \times Z$. The goal is to choose a set of disjoint $hyperedges M \subseteq E$ such that $|M| = k$. 3DM is APX-hard, see [27]. This means that there is a constant $\epsilon > 0$ such that there is no polynomial time algorithm that can decide whether there is a matching of size k or all matchings have cardinality less than or equal to $(1-\epsilon)k$, unless P=NP.

First, consider the instance of MMM with three timesteps T = 3. The universe elements correspond to the hyperedges. For t = 1, create a partition with k parts, with hyperedges sharing a vertex in X falling in the same part.The matroid M_1 is now to choose a set of elements with at most one element in each part. For t = 2, the partition now corresponds to hyperedges that share a vertex in Y , and for t = 3, hyperedges that share a vertex in Z. Set the acquisition costs a(e) = 1 for all hyperedges.

Now, since 3DM is APX-hard, there is a constant $\epsilon > 0$ such that there is no $\epsilon-$approximation algorithm for 3DM, unless P=NP. Now, let's consider two cases:

- There is a matching of size k. Observe that this happens exactly when the optimal cost of MMM is k. Indeed, to take the first base, you need to pay acquisition cost k. The cost is exactly k when no new element is taken to build the other two bases. Thus, from the definition of the three matroids, the optimal solution has cost k if and only if there is a perfect 3DM.
- All matchings have cardinality less than or equal to $(1-\epsilon)k$. In this case, the optimal solution for MMM will have cost at least $(1+\epsilon)k$. Indeed, let's consider the optimal solution for this instance of MMM. From the definition of the 3 matroids, we have that the elements that belong the intersection of the 3 bases correspond to a matching and thus are less than or equal to $(1-\epsilon)k$. Now the elements that belong to the first base but not at all three bases are at least ϵk and since all bases have cardinality k, when an element is dropped, another is bought. Thus, there is an extra ϵk acquisition cost that has to be added at the initial cost k.

Now, by repeating the above matroid triple in time, we can extend the initial gap. More specifically, at time $t \in [T]$ we will have the matroid $\mathcal{M}_{tmod3+1}$. In the first case ("yes" case) the optimal solution for MMM pays cost exactly k. In the second case ("no" case) the optimal solution for MMM pays an additional ϵk acquisition cost every 3 time steps. Thus, the overall cost will be $(1+T\epsilon)k$. From the APX-hardness, there cannot be an $(1+T\epsilon) - approximationalgorithm$ for MMM, unless P=NP. The claim follows from the fact tha ϵ is a constant.

$\square$

Notice that the time-varying MSM problem does admit an $O(\log rT)$ approximation, as the randomized rounding (or the greedy algorithm) shows. However, the equivalence of MMM and MSM does not go through when the matroids change over time! The restriction that the matroids vary over time is essential for the NP-hardness, since if the partition matroid is the same for all times, the complexity of the problem drops radically, as we will show at the next chapter.

3.5 Perfect Matching Maintenance

In this section we study the *Perfect Matching Maintenance* (PMM) problem:

Definition 21. *An instance of the Perfect Matching Maintenance (PMM) consists of a graph $G(V,E)$, $n=|V|$, $m=|E|$, an acquisition cost $g \geq 0$ for each $e \in E$, and for every time step $t \in [T]$ and edge $e \in E$, a holding cost $c_t(e) \geq 0$. The goal is to find perfect matchings $\{P_t\}_{t\in[T]}$ to minimize*

$$\sum_t (c_t(P_t) + g|P_t \setminus P_{t-1}|)$$

where we define $P_1 \setminus P_0 := \emptyset$.

Observe that for the PMM we consider uniform acquisition costs, similarly to the case of the Dynamic Facility Location problem, see [18], [2]. For T=1 dynamic facility location is NP-hard and for general T it can be approximated with a constant factor, see [2]. For T=1, perfect matching maintenance is in P. Is this an indication that the problem can be approximated with a constant factor or even better for general T? In this section we will show that the answer is no, by presenting results proven in [22].

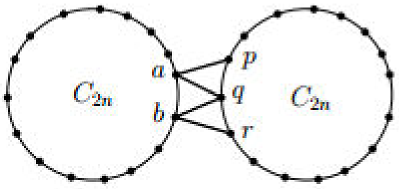
First of all, let's study the natural LP relaxation:

$$\begin{array}{ll} \text{minimize} & \sum_{t,e} c_t(e) z_t(e) + g \sum_{t,e} y_t(e) \\ \text{subject to} & \vec{z_t} \in PM(G), \quad \forall t \in [T] \\ & y_t(e) \geq z_t(e) - z_{t-1}(e), \quad \forall e \in E, \quad \forall t \in [T] \\ & y_t(e), z_t(e) \geq 0, \ \forall e \in E, \quad \forall t \in [T] \end{array}$$

The polytope PM(G) is the perfect matching polytope for G (its vertices are exactly the indicator vectors of the perfect matchings of G), see [37]. Observe that in the above constraints, $\vec{z_0}$ is included, but it is not given to the LP as a variable but as an identically zero vector.

Lemma 3.5.1. *The LP for the PMM has $\Omega(n)$ integrality gap.*

Proof. The integrality gap instance has four time steps, g=1 and the edge set is indicated in the figure below:



We say that an edge at time t is alive if $c_t(e) = 0$, otherwise we say that e is not alive at time t and then $c_t(e) = \infty$. For each time step we write which edges are **not** alive:
t=1: (a,p), (a,q), (b,q), (b,r)
t=2: (a,b), (p,q), (a,q), (b,r)
t=3: (a,p), (a,q), (b,q), (b,r)
t=4: (a,b), (q,r), (a,p), (b,q)

Observe that at each time step the alive edges for an even cycle or a union of even cycles, thus the fractional solution $\vec{z_t} = (1/2, ..., 1/2)$ is feasible for all t. The total cost is the total fractional acquisition cost , which is O(1).

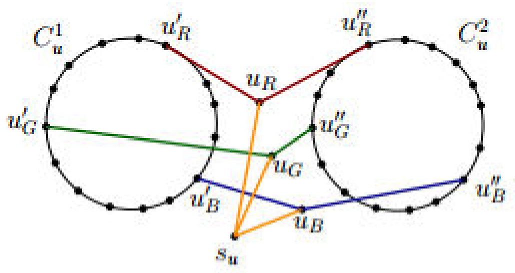
Now, let's take an integral feasible solution. Consider the perfect matching at time t = 1, which must consist of matchings on both the cycles. (Moreover, the matching in time 3 must be the same, else we would change $\Omega(n)$ edges). Suppose this matching uses exactly one edge from (a,b) and (p,q). Then when we drop the edges (a,b), (p,q) and add in (a,p), (b,q), we get a cycle on 4n vertices, but to get a perfect matching on this in time 2 we need to change $\Omega(n)$ edges. Else the matching uses exactly one edge from (a,b) and (q,r), in which case going from time 3 to time 4 requires $\Omega(n)$ changes. Thus, the total cost of every integral feasible solution is $\Omega(n)$. The lemma follows. We should note that we would also have integrality gap $\Omega(n)$ if we kept only the time steps t=2 and t=4 (in that case T=2).

□

Observe that a crucial propery that is used in order to prove the integrality gap is that the optimal solution of the PMM is highly unstable at small changes of the alive edges. Note that this does not hold at matroids, where if an element stops being alive, we can exchange it with another one. Another way to see this is that in the matroid base polytope, the neighboring vertices correspond to bases whose difference has cardinality 1. This definitely does not hold for the perfect matching polytope (take the one for the C_{2n}). This property will be exploited to show the following hardness result for PMM:

Theorem 3.5.2. *For any $\epsilon > 0$, there is no $O(N^{1-\epsilon}-$polynomial time approximation algorithm for PMM, unless P=NP, where N is the number of vertices in the graph. This holds even when the holding costs are in $\{0, \infty\}$, acquisition costs are 1 for all edges, and the number of time steps is a constant.*

Proof. The proof is via reduction from 3-coloring. In 3-coloring, we have three colors, say red, blue, green and the question is whether we can color the vertices of the graph with these colors such that no two vertices that are connected with an edge share the same color. We assume we are given an instance of 3-coloring $G = (V, E)$ where the maximum degree of G is constant. It is known that the 3-coloring problem is still NP-hard for graphs with bounded degree, see [23]. At the beginning we consider $T = 2|E|$. We construct a gadget X_u for each vertex $u \in V$, as in the figure below:

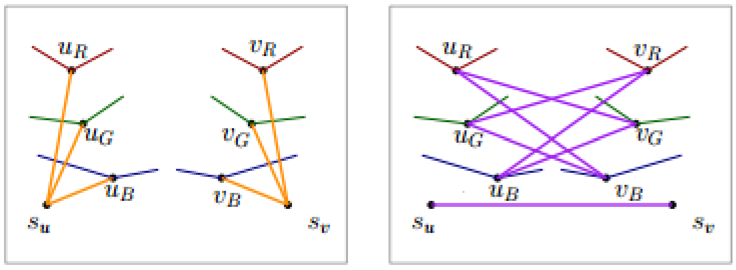


More specifically, in each gadget X_u:

- There are two cycles of length 3ℓ, where ℓ is odd. The first cycle (say C_u^1) has three distinguished vertices u_R^1, u_G^1, u_B^1 at distance ℓ from each other. The second cycle (say C_u^2) has similar distinguished vertices u_R^2, u_G^2, u_B^2 at distance ℓ from each other.
- There are three more "interface" vertices u_R, u_G, u_B. Vertex u_R is connected to u_R^1 and to u_R^2, similarly for u_G and u_B.

- There is a special "switch" vertex s_u, which is connected to all three of $\{u_R, u_G, u_B\}$. Call these edges the switch edges.

Due to the two odd cycles, every perfect matching in X_u has the structure that one of the interface vertices is matched to some vertex in C_u^1, another to a vertex in C_u^2 and the third to the switch s_u. The subscript of the vertex that is matched to s_u encodes the color assigned to vertex u.
At every odd time step $t \in [T]$, the only allowed edges are those within the gadgets $X_{u\{}u \in V\}$: i.e., all the holding costs for edges within the gadgets is zero, and all edges between gadgets have holding costs ∞. This is called the "steady state".
We make a bijection between the even time steps and the edge set E. At every even time step t, we move into a "test state", which intuitively tests whether the corresponding edge satisfies the color constraint. We do this as follows. Say that the edge corresponding at time t is the (u, v). At time t, the switch edges in X_u, X_v become unavailable (have infinite holding costs). Moreover, now we allow some edges that go between X_u and X_v, namely the edge (s_u, s_v), and the edges (u_i, v_j) for $i, j \in \{R, G, B\}$ and $i \neq j$. Note that any perfect matching on the vertices of $X_u \cup X_v$ which only uses the available edges would have to match (s_u, s_v) and one interface vertex of X_u must be matched to one interface vertex of X_v. Moreover, by the structure of the allowed edges, the colors of these vertices must differ. (The other two interface vertices in each gadget must still be matched to their odd cycles to get a perfect matching.) The transition of the alive edges from time t-1 to time t is indicated in the figure below:



Suppose the graph G was indeed 3-colorable, say $X : V \leftarrow \{R, G, B\}$ is the proper coloring. In the steady states, we choose a perfect matching within each gadget X_u so that $(s_u, u_{X(u)})$ is matched. In the test state 2t, corresponding to the edge (u, v), we match (s_u, s_v) and $(u_{X(u)}, v_{X(v)})$. Since the coloring X was a proper coloring, these edges are present and this is a valid perfect matching using only the edges allowed in this test state. Note that the only changes between time 2t-1 and 2t were to replace the matching edges $(s_u, u_{X(u)})$ and $(s_v, v_{X(v)})$ by (s_u, s_v) and $(u_{X(u)}, v_{X(v)})$ respectively. Hence the total acquisition cost incurred at time 2t is 2, and the same acquisition cost is incurred at time 2t + 1 to revert to the steady state. Hence the total acquisition cost, summed over all the time steps, is $4|E|$.

Suppose that G is not 3 colorable. Let's fix a feasible solution of PMM: $\{P_t\}_{t\in[T]}$. For a cycle $C_u^j, j \in \{1,2\}$ we define $r_t(C_u^j)$ to be the vertex that belongs to C_u^j and at time t it is matched outside of it (remember that for each time step this vertex is unique). If for any C_u^j, $r_t(C_u^j) \neq r_1(C_u^j)$, for some t, then it is easy to observe that the solution has to pay $\Omega(\ell)$ acquisition cost. Since G is not 3-colorable, looking the feasible solution at the first time step, we can find two gadgets X_u, X_v such that $(u,v) \in E$ and $(s_u, u_x), (s_v, v_x) \in P_1$, $x \in \{R, G, B\}$. Now, let's look at P_t, where t is the time where edge (u,v) is tested. Since (u_x, v_x) is not alive at time t (actually it is never alive), $r_t(C_u^j) \neq r_1(C_u^j)$. Thus, the total cost of an feasible solution of PMM has cost $\Omega(\ell)$. Set $\ell = n^{\frac{1}{\epsilon}-1}$.

The vertex set for the PMM has cardinality $N = \Theta(n\ell) = \Theta(n^{\frac{1}{\epsilon}})$. Thus, if G is 3-colorable, then the optimal solution of PMM has cost $O(4|E|) = O(n) = O(N^{\epsilon})$ (constant degree) and if G is not 3-colorable, then the optimal solution of PMM has cost $\Omega(ell) = \Omega(N^{1-\epsilon})$. This gap proves the theorem for general T. To show it for constant T, we can parallelize the above procedure. More specifically, since the graph has bounded degree, we can partition the edges of G into a constant number of matchings $M_1, M_2, ..., M_k$ for some $k = O(1)$ (using Vizing's theorem). Hence, at time step 2t, we test the edges of the matching M_t. The number of time steps now is $T = 2k$, which is a constant. □

Chapter 4

MMM in special cases

In this chapter, we present our contribution to the MMM problem. First of all, the reduction and the integrality gap instance presented in Chapter 3 are for the case of the graphic matroid. So, in this chapter, we study the LP for the case of the partition matroid in the more general case of time-changing acquisition costs and we prove that it is integral. Second, in Chapter 3 we saw that if the matroids change over time, the problem is hard, even for T=3. Here we show that the LP is integral for T=2, even for different matroids. Finally, the straightforward and direct reduction from set cover for MMM indicates that we cannot do anything better than a simple logarithmic approximation algorithm. So, as in [12], we consider the case where the acquisition costs are uniform. For this very interesting special case, we present an algorithm that has constant approximation at the holding cost and logarithmic approximation at the acquisition cost.

4.1 Integral LP formulations

In [22], the authors present exact polynomial time algorithms for the case of MMM for partition matroids and for T=2. Here we prove the integrality of the LP in these two cases.

4.1.1 Partition Matroids

The LP that we used for MMM, in the case of partition matroids takes the following form:

$$\begin{aligned}
\textit{minimize} \quad & \sum_{t=1}^{T}\sum_{e\in\mathcal{S}} c_t(e)z_t(e) + \sum_{t=2}^{T}\sum_{e\in\mathcal{S}} a_t(e)y_t(e) \\
\textit{subject to} \quad & z_t(S_i) = k_i, \quad \forall i \in [n],\ t \in [T] \\
& y_t(e) \geq z_t(e) - z_{t-1}(e), \quad \forall e \in \mathcal{S}, \quad t = 2, ..., T \\
& 0 \leq z_t(e) \leq 1,\ \forall e \in \mathcal{S}, \quad \forall t \in [T] \\
& y_t(e) \geq 0,\ \forall e \in \mathcal{S}, \quad t = 2, ..., T
\end{aligned}$$

Note that in this case we allow the acquisition cost to be time dependent. We should make a technical note that the acquisition cost $a_1(B_1 \setminus B_0) = a_1(B_1)$ is integrated in the holding cost c_1. Second, it is easy to observe that for each S_i the problem is solved independently. Thus, it suffices to show the integrality of the LP for the Uniform Matroid. So, for the rest of the section, we will refer to the following LP:

$$\begin{aligned}
\textit{minimize} \quad & \sum_{t=1}^{T}\sum_{e\in\mathcal{S}} c_t(e)y_t(e) + \sum_{t=2}^{T}\sum_{e\in\mathcal{S}} a_t(e)z_t(e)\\
\textit{subject to} \quad & z_t(\mathcal{S}) = k, \quad \forall t \in [T]\\
& y_t(e) \geq z_t(e) - z_{t-1}(e), \quad \forall e \in \mathcal{S}, \quad t = 2, ..., T\\
& 0 \leq z_t(e) \leq 1, \ \forall e \in \mathcal{S}, \ \ \forall t \in [T]\\
& y_t(e) \geq 0, \ \forall e \in \mathcal{S}, \quad t = 2, ..., T
\end{aligned}$$

*Let $\{z_t(e), y_t(e)\}_{t,e}$ be an extreme point solution of the LP. Note that $y_t(e) = max(z_t(e) - z_{t-1}(e), 0)$, $\forall\ e \in \mathcal{S},\ t \geq 2$ **(1)**. Indeed, if this is not the case, since the constraints force $y_t(e)$ to be greater or equal to $max(0, z_t(e) - z_{t-1}(e))$ and this is the only constraint to $y_t(e)$, for suciently small $\epsilon > 0$, without changing the other variables, if we subtract ϵ from $y_t(e)$ the solution is still feasible. Obviously, if we add ϵ to $y_t(e)$ the solution is still feasible. Hence, $\{z_t(e), y_t(e)\}_{t,e}$ can be written as a convex combination of two feasible solutions of the LP, contradiction.*

*We will prove that $\{z_t(e), y_t(e)\}_{t,e}$ has integral entries. Because of **(1)** it suffices to prove that $\{z_t(e)\}_{t,e}$ has integral entries. Let's suppose that this is not the case, i.e $\exists(e,t) \in \mathcal{S}\times[T]$ such that $0 < z_t(e) < 1$. Let $F_t = \{e \in \mathcal{S} :\ 0 < z_t(e) < 1\}$, $O_t = \{e \in \mathcal{S} :\ z_t(e) = 1\}$ and $Z_t = \{e \in \mathcal{S} :\ z_t(e) = 0\}$. Clearly, $\cup_t F_t \neq \emptyset$. We will show that $\{z_t(e), y_t(e)\}_{t,e}$ can be written as convex combination of two feasible solutions of the LP, which is a contradiction.*

We will say, that two sets $T_1, T_2 \subseteq \mathcal{S} \times [T]$, $T_1 \cap T_2 = \emptyset$, enable a feasible perturbation of $\{z_t(e), y_t(e)\}_{t,e}$ if $\exists\epsilon > 0$ such that

1. *$\{z'_t(e), y'_t(e)\}_{t,e}$ and $\{z''_t(e), y''_t(e)\}_{t,e}$ are feasible solutions for the LP, where*

$$z'_t(e) = \begin{cases} z_t(e) + \epsilon & \textit{if } (e,t) \in T_1\\ z_t(e) - \epsilon & \textit{if } (e,t) \in T_2\\ z_t(e) & \textit{otherwise}\end{cases}$$

$$z''_t(e) = \begin{cases} z_t(e) - \epsilon & \textit{if } (e,t) \in T_1\\ z_t(e) + \epsilon & \textit{if } (e,t) \in T_2\\ z_t(e) & \textit{otherwise}\end{cases}$$

and $y'_t(e) = max(z'_t(e) - z'_{t-1}(e), 0)$, $\forall\ e \in \mathcal{S},\ t \geq 2$, $y''_t(e) = max(z''_t(e) - z''_{t-1}(e), 0)$, $\forall\ e \in \mathcal{S},\ t \geq 2$

2. *For all $e \in \mathcal{S}$, $t \geq 2$: $z_t(e) - z_{t-1}(e)$, $z'_t(e) - z'_{t-1}(e)$, $z''_t(e) - z''_{t-1}(e)$ have all the same sign (+,0,-).*

*Observe that in this case, $z_t(e) = \frac{z'_t(e)+z''_t(e)}{2}$ $\forall e, t$ **(2)**. Now take one $e \in \mathcal{S}$ and one $T \in \{2, ..., T\}$. If $y_t(e) = 0$, then $z_t(e) - z_{t-1}(e) \leq 0$, thus,from the second property, $y'_t(e) = y''_t(e) = 0$. Else, $y_t(e) = z_t(e) - z_{t-1}(e) > 0 \Rightarrow z'_t(e) - z'_{t-1}(e) > 0$ and $z''_t(e) - z''_{t-1}(e) > 0$. Hence, $y'_t(e) = z'_t(e) - z'_{t-1}(e)$ and $y''_t(e) = z''_t(e) - z''_{t-1}(e)$. So, from **(2)** we have that $y_t(e) = \frac{y'_t(e)+y''_t(e)}{2}$. Thus, $\{z_t(e), y_t(e)\}_{t,e}$ can be written as a convex combination of $\{z'_t(e), y'_t(e)\}_{t,e}$ and $\{z''_t(e), y''_t(e)\}_{t,e}$.*

We will show that we can find two sets $T_1, T_2 \subseteq \mathcal{S} \times [T]$, $T_1 \cap T_2 = \emptyset$, that enable a feasible perturbation of $\{z_t(e), y_t(e)\}_{t,e}$.

Now, for each $(e,t) \in \mathcal{S} \times [T]$, let $I_{e,t} = [t_1, t_2]$ where

$$t_1 = min\{t' : \ 1 \leq t' \leq t, \ z_{t'}(e) = z_{t'+1}(e) = ... = z_t(e)\}$$
$$t_2 = max\{t' : \ t \leq t' \leq T, \ z_t(e) = z_{t+1}(e) = ... = z_{t'}(e)\}$$

We will refer to t_1 as $I_{e,t}.\ell$ and to t_2 as $I_{e,t}.r$.

Now, given $T_1, T_2 \subseteq \mathcal{S} \times [T]$, $T_1 \cap T_2 = \emptyset$ such that

$$(e,t) \in T_i \Rightarrow \cup_{t' \in I_{e,t}} (e, t') \subseteq T_i, \ \forall i \in \{1,2\} \ \textbf{(3)}$$

*we claim that $\exists \ \epsilon > 0$ such that the second property (the one with the signs) holds. Indeed, consider an $e \in \mathcal{S}$ and a $t \geq 2$ and let's focus on $z_t(e) - z_{t-1}(e)$. If $z_t(e) - z_{t-1}(e) = 0$, then from **(3)** we have that either both $(e,t), (e,t-1) \in T_i$ or none of them does, for $i = 1, 2$. Now, for*

$$\epsilon = \frac{1}{2} min_{e,t: \ z_t(e) - z_{t-1}(e) \neq 0} \{|z_t(e) - z_{t-1}(e)|\}$$

property 2 holds.

Let

$$t_{start} = min\{t \in [T] | \ F_t \neq \emptyset\}$$
$$t_{end} = max\{t \geq t_{start} | \ \exists e_1, e_2 \in F_t : \ e_1 \neq e_2 \text{ and } I_{e_1,t}.r = I_{e_2,t}.r = t\}$$

t_{start} is the first moment that we see fractional values and t_{end} is the first moment that two fractional equality intervals end simoultaneously.

Proposition 4.1.0.1. *t_{end} is well-defined.*

Proof. First, notice that if $F_t \neq \emptyset$, then since $z_t(\mathcal{S}) = k$, there exist at least two different elements $e_1, e_2 \in \mathcal{S}$ such that $z_t(e_1), z_t(e_2)$ are fractional. Let $A = \{t \geq t_{start} | \ F_t = \emptyset\}$. If $A = \emptyset$, then $\exists e_1, e_2 \in F_T : \ e_1 \neq e_2$ *and* $I_{e_1,T}.r = I_{e_2,T}.r = T$. Else, if t^* is the minimum element of A, then $\exists e_1, e_2 \in F_{t^*-1} : \ e_1 \neq e_2$ *and* $I_{e_1,t^*-1}.r = I_{e_2,t^*-1}.r = t^* - 1$. □

For $t \in [T-1]$, let $c(t) = \{e \in F_t |\ z_t(e) \neq z_{t+1}(e)\}$ and observe that

$$|c(t)| \leq 1,\ \forall t \in [t_{start}, t_{end})$$

For each $t \in [t_{start}, t_{end})$ such that $|c(t)| = 1$, let $c(t) = \{e(t)\}$. For such a time t, we have that:

$$z_t(\mathcal{S}) = z_{t+1}(\mathcal{S}) = k \Leftrightarrow$$
$$|O_t| + z_t(F_t \setminus \{e(t)\}) + z_t(e(t)) = |O_{t+1} \setminus \{e(t)\}| + z_{t+1}(F_{t+1} \setminus \{e(t)\}) + z_{t+1}(e(t)) = k \Rightarrow$$
$$z_{t+1}(F_{t+1} \setminus \{e(t)\}) + z_{t+1}(e(t)) - z_t(F_t \setminus \{e(t)\}) - z_t(e(t)) \in \mathbb{Z} \Rightarrow$$
$$z_{t+1}((F_{t+1} \setminus F_t) \setminus \{e(t)\}) + (z_{t+1}(e(t)) - z_t(e(t))) \in \mathbb{Z} \textbf{ (4)}$$

*Where the last step holds because $c(t) = \{e(t)\}$. But, $0 \leq z_{t+1}(e(t)) \leq 1$, $0 < z_t(e(t)) < 1$, $z_{t+1}(e(t)) \neq z_t(e(t))$. So, $0 < |z_{t+1}(e(t)) - z_t(e(t))| < 1$ and from **(4)**: $z_{t+1}((F_{t+1} \setminus F_t) \setminus \{e(t)\}) \neq 0$, i.e $(F_{t+1} \setminus F_t) \setminus \{e(t)\} \neq \emptyset$. Since $e(t) \in F_t$, $(\mathcal{S} \setminus F_t) \cap F_{t+1} \neq \emptyset$. Observe that $I_{e,t+1}.\ell = t+1$, $\forall e \in (\mathcal{S} \setminus F_t) \cap F_{t+1}$. Now, we define a function f, which maps element-time interval tuples to element-time interval tuples, its domain is $\{(e, I_{e,t})|\ t \in (t_{start}, t_{end}],\ e \in ((\mathcal{S} \setminus F_t) \cup \{e(t)\}) \cap F_{t+1}\}$ and*

$$f(e, I_{e,t+1}) = (e(t), I_{e(t),t})$$

which actually says that e(t) is "responsible" for the fractionality of e at time t+1.

Now, observe that $\forall t \in [t_{start}, t_{end})$ and $\forall e \in F_{t+1}$ there are two possibilities:

1. *$I_{e,t} = I_{e,t+1}$ or*

2. *$f(e, I_{e,t+1}) = (e(t), I_{e(t),t})$*

Corollary 4.1.0.1. *$\forall (e,t) \in \mathcal{S} \times [T]$, such that $e \in F_t$, $I_{e,t} \subseteq (t_{start}, t_{end}]$, if $I_{e,t}.\ell = t'$, then $f(e, I_{e,t}) = (e(t'-1), I_{e(t'-1),t'-1})$.*

We will now construct two lists of element-interval tuples, L_1, L_2, using the following algorithm:

where e_1 and e_2 are the ones from the definition of t_{end}.

Lemma 4.1.1. *The following statements are true:*

1. *The algorithm is well-defined.*

2. *$I_1, I_2 \subseteq [t_{start}, t_{end}]$ in the course of the algorithm.*

3. *For each iteration, $e \in F_{I_1.\ell}$ and $e' \in F_{I_1.\ell}$.*

Proof. The proof is by induction on the number of iterations of the algorithm. The base case follows from the definitions of t_{start}, t_{end}. Now, let suppose that I_1, I_2 enter the while loop at some iteration, i.e $I_1.\ell \neq I_2.\ell$. At this iteration, the corresponding elements are

```
1: $L_1, L_2 \leftarrow \emptyset$
2: $I_1 \leftarrow I_{e_1,t_{end}}$
3: $I_2 \leftarrow I_{e_2,t_{end}}$
4: $e \leftarrow e_1$
5: $e' \leftarrow e_2$
6: while $I_1.\ell \neq I_2.\ell$ do
7:     if $I_1.\ell > I_2.\ell$ then
8:         $L_1 \leftarrow L_1 \cup (e, I_1)$
9:         $(e, I_1) \leftarrow f(I_1)$
10:    else
11:        $L_2 \leftarrow L_2 \cup (e', I_2)$
12:        $(e', I_2) \leftarrow f(I_2)$
13: return $L_1, L_2$
```

e,e'. From the induction hypothesis, we have that $e \in F_{I_1.\ell}$ and $e' \in F_{I_1.\ell}$. We consider the case, where $I_2.\ell > I_1.\ell$, the other case is proved via the same argument. $I_2.\ell \geq t_{start}+1$, i.e $I_2 \subseteq (t_{start}, t_{end}]$ and thus, from corollary 4.1.0.1: $f(e', I_2)$ is well-defined and $f(e', I_2) = (e(t^*-1), I_{e(t^*-1),t^*-1})$, where $t^* = I_2.\ell$. We also have that $e(t^*-1) \in F_{t^*-1} \Rightarrow I_{e(t^*-1),t^*-1} \subseteq [t_{start}, t_{end}]$, from the definition of t_{start}. □

From the definition of f we have that at each iteration, we add at one of the lists an interval which ends where the last interval added to this list beginns. Since $I_1, I_2 \subseteq [t_{start}, t_{end}]$ in the course of the algorithm, we have that the algorithm will terminate. The algorithm ends with some I_1, I_2. Let $t_{alg} = I_1.\ell = I_2.\ell$

Lemma 4.1.2. *$\forall t \in [t_{alg}, t_{end}]$, there exists an iteration of the algorithm, where the intervals are I_1, I_2, the elements are e,e' and $t \in I_1 \cap I_2$ and $e \neq e'$.*

Proof. For $t = t_{end}$, the lemma holds from the definition of t_{end}. Let's suppose that the lemma is not true for some $t \in [t_{alg}, t_{end}]$ and let t^* be the maximum such t. Thus, at some time during the course of the algorithm, we have intervals I_1, I_2, elements e,e' and $t+1 \in I_1 \cap I_2$ and $e \neq e'$. If $I_1.\ell = I_2.\ell = t_{alg}$, then $[t_{alg}, t+1] \subseteq I_1 \cap I_2$ contradiction. So, let's suppose that $I_1.\ell > I_2.\ell$ (for the other case the contradiction is derived via the same argument). Obviously, $t \notin I_1 \cap I_2$. Let I'_1 be the update of I_1. So, $t = I_1.\ell - 1 = I'_1.r$. Since $I_2.r \geq I_1.\ell$ (they have nonempty intersection) and $I_2.\ell \leq I_1.\ell - 1$, we have that $t \in I'_1 \cap I_2$. The fact that I'_1 ends at time $I_1.\ell - 1$, I_2 ends after time $I_1.\ell - 1$ and $t \in I'_1 \cap I_2$ indicates that I'_1 and I_2 do not correspond to the same element, i.e. $e(I_1.\ell - 1) \neq e'$, contradiction. □

*Let $T_1 = \cup_{(e,I) \in L_1} \cup_{t \in I} (e, t)$ and $T_2 = \cup_{(e,I) \in L_2} \cup_{t \in I} (e, t)$ **(5)**. From lemma 4.1.2 and from the fact that each list is a list of consecutive intervals, we have that $T_1 \cap T_2 = \emptyset$.*

Theorem 4.1.3. *T_1,T_2 enable a feasible perturbation of the extreme point.*

Proof. First of all, from the property 3 of lemma 4.1.1, we have that each $z_t(e)$, $(e,t) \in T_1 \cup T_2$ is fractional. Thus, ϵ can be chosen to be sufficiently small, so that the constraints $0 \leq z'_t(e) \leq 1$, $0 \leq z''_t(e) \leq 1$ are satisfied $\forall e \in \mathcal{S}, t \in [T]$. Each list is a list of consecutive intervals, so lemma 4.1.2. indicates that for a specific time step, we either have no perturbation or two perturbations, which are eliminated in the sum. More specifically, $z'_t(\mathcal{S}) = z''_t(\mathcal{S}) = k$, $\forall e \in \mathcal{S}, t \in [T]$. Thus, the first property of the *feasible perturbation* is satisfied. Also, T_1, T_2 are a union of maximal equality intervals, see **(5)**. Hence, they satisfy **(3)** and as we have proved, this implies that the property 2 of the *feasible perturbation* is satisfied.

□

The fact that an extreme point of a polytope cannot be written as a convex combination of other points of the polytope gives as the main theorem:

Theorem 4.1.4. *The natural LP for MMM, for the case of Partition Matroids is integral.*

4.1.2 T=2

Here, we prove that the LP formulation of a generalization of MMM for T=2 is integral. We call this generalization Partial Zero Switch MMM and we define it below:

Definition 22. *A $T = 2$ instance of the Partial Zero Switch Multistage Matroid Maintenance problem consists of two matroids $\mathcal{M}_1 = (\mathcal{S}_1, \mathcal{I}_1)$, $\mathcal{M}_2 = (\mathcal{S}_2, \mathcal{I}_2)$, a set $A \subseteq \mathcal{S}_1 \cap \mathcal{S}_2$, an acquisition cost $a(e) \geq 0$ for each $e \in \mathcal{S}_1 \cup \mathcal{S}_2$, and two holding cost functions $c_1(e)$, $e \in \mathcal{S}_1$ and $c_2(e)$, $e \in \mathcal{S}_2$. The goal is to find bases $B_1 \in \mathcal{I}_1$ and $B_2 \in \mathcal{I}_2$ such that $((B_1 \setminus B_2) \cup (B_2 \setminus B_1)) \cap A = \emptyset$ and minimize*

$$c_1(B_1) + c_2(B_2) + a(B_2 \setminus B_1)$$

The set A is called zero switch set. and we say that the two bases respect A.

Why this is a generalization of MMM? Again here, the acquisition cost $a(B_1 \setminus B_0) = a(B_1)$ is integrated in the holding cost c_1. Also, in MMM we have a common matroid for all time steps. Plus, in MMM, $A = \emptyset$. Finally, for technical reasons we have removed the non-negativity constraint of the holding costs from the definition.

Now, let's see what is this set A. The set A, given in the input, imposes the constraint that each of its elements is either taken to both bases or to none of them. It's time to write down the LP formulation. The variables-vectors of the LP will be $z \in \mathbb{R}^{|A|}$, $z_1 \in \mathbb{R}^{|\mathcal{S}_1 \setminus A|}$, $z_2 \in \mathbb{R}^{|\mathcal{S}_2 \setminus A|}$, $y \in \mathbb{R}^{|(\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A|}$. We refer to this formulation as $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$.

$$
\begin{aligned}
\textit{minimize} \quad & \sum_{e\in A}(c_1(e)+c_2(e))z(e) + \sum_{e\in \mathcal{S}_1\setminus A} c_1(e)z_1(e) + \sum_{e\in \mathcal{S}_2\setminus A} c_2(e)z_2(e) + \\
& \sum_{e\in(\mathcal{S}_1\cap\mathcal{S}_2)\setminus A} a(e)y(e) + \sum_{e\in \mathcal{S}_2\setminus \mathcal{S}_1} a(e)z_2(e) \\
\textit{subject to} \quad & z_1(T\setminus A) + z(T\cap A) \le r_1(T), \quad \forall T\subseteq \mathcal{S}_1 \\
& z_1(\mathcal{S}_1\setminus A) + z(A) = r_1(\mathcal{S}_1) \\
& z_2(T\setminus A) + z(T\cap A) \le r_2(T), \quad \forall T\subseteq \mathcal{S}_2 \\
& z_2(\mathcal{S}_2\setminus A) + z(A) = r_2(\mathcal{S}_2) \\
& y(e) \ge z_2(e) - z_1(e), \quad \forall e\in(\mathcal{S}_1\cap\mathcal{S}_2)\setminus A \\
& z(e)\ge 0, \quad \forall e\in A \\
& z_1(e)\ge 0, \quad \forall e\in \mathcal{S}_1\setminus A \\
& z_2(e)\ge 0, \quad \forall e\in \mathcal{S}_2\setminus A \\
& y(e)\ge 0, \quad \forall e\in(\mathcal{S}_1\cup\mathcal{S}_2)\setminus A
\end{aligned}
\tag{4.1}
$$

To get a separation oracle, given as input independence oracles for each of the matroids, we can use the work of Cunningham [37] or any algorithm for minimizing submodular functions.

Characterization of Extreme Point Solutions

We now give a characterization of extreme points of the linear program by showing that the independent set of tight constraints that gives the coordinates of an extreme point, except of these that belong to y, can be chosen to form a union of two chains. The proof is quite straightforward and uses the characterization of tight inequalities for the max-weight independent set problem.

Given an extreme point solution (z, z_1, z_2, y) *to* $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$, *let* $\mathcal{F}_1 = \{T \subseteq \mathcal{S}_1 \;:\; z_1(T\setminus A) + z(T\cap A) = r_1(T)\}$ *and* $\mathcal{F}_2 = \{T \subseteq \mathcal{S}_2 \;:\; z_2(T\setminus A) + z(T\cap A) = r_2(T)\}$.

Lemma 4.1.5. *There exist two chains* $C_1 \subseteq 2^{\mathcal{S}_1}$ *and* $C_2 \subseteq 2^{\mathcal{S}_2}$ *such that* $span(C_1\cup C_2) = span(\mathcal{F}_1 \cup \mathcal{F}_2)$ *and constraints in sets* C_1 *and* C_2 *are linearly independent.*

Proof. Applying Lemma 2.3.2 to families $\mathcal{F}_1$ and $\mathcal{F}_2$ separately, we obtain two chains $\hat{C}_1$ and $\hat{C}_2$ such that $span(\hat{C}_1) = span(\mathcal{F}_1)$ and $span(\hat{C}_2) = span(\mathcal{F}_2)$. Now, picking a maximal independent family from $\hat{C}_1 \cup \hat{C}_2$ gives us the desired chains. □

Notice that the number of variables is $|\mathcal{S}_1|+|\mathcal{S}_2|+|\mathcal{S}_1\cap\mathcal{S}_2|-2|A|$. *So, the characteristic vectors of the constraints have this dimension. For* $T\subseteq \mathcal{S}_1$, *let* $\chi_1(T)$ *be the characteristic vector of the constraint* $z_1(T\setminus A)+z(T\cap A)\le r_1(T)$. *For* $T\subseteq\mathcal{S}_2$, *let* $\chi_2(T)$ *be the characteristic vector of the constraint* $z_2(T\setminus A)+z(T\cap A)\le r_2(T)$.

Lemma 4.1.6. *Let (z, z_1, z_2, y) be any extreme point solution to $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$ such that (z, z_1, z_2) has positive entries and $\nexists e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A : \ z_1(e) = z_2(e)$. Then there exist two chains C_1 and C_2 such that*

1. *$z_i(T \setminus A) + z(T \cap A) = r_i(T)$ for each $T \subseteq C_i$, for $i = \{1, 2\}$.*
2. *The vectors in $\{\chi_1(T) \ : \ T \in C_1\} \cup \{\chi_2(T) \ : \ T \in C_2\}$ are linearly independent.*
3. *$|C_1| + |C_2| = |\mathcal{S}_1| + |\mathcal{S}_2| - |A|$.*

Proof. The basic feasible solution is produced by a system of linear equations. The equations that come from the matroid constraints can be chosen to have the structure given at lemma 4.1.5. The remaining equations come from the inequalities $y(e) \geq z_2(e) - z_1(e)$ and $y(e) \geq 0$, $\forall e \in (\mathcal{S}_1 \cup \mathcal{S}_2) \setminus A$. But since $\nexists e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A : \ z_1(e) = z_2(e)$, at most one of these inequalities, for each $e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$ is equality. Thus, to compute z, z_1, z_2 from this system of linear equations, we only need the equations that come from the matroid constraints, i.e from the chains. The lemma follows. □

Iterative Algorithm

We now give an iterative algorithm which constructs an integral solution from the linear program and shows that the linear programming formulation is integral. Notice that this algorithm is different from these presented at chapter two, because, except of the matroids and the bases, it also updates the costs. This is reasonable, because if there is an element $e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$ with $z_2(e) = 1 \neq z_1(e)$ and we contract e at $\mathcal{M}_1$, we have to pass the information at the residual problem, that this elemt is already taken, see ***Algorithm 5****.*

Correctness and Optimality

Now, we show that in each iteration there will be at least one realizable condition, i.e the algorithm will terminate:

Lemma 4.1.7. *For any extreme point solution (z, z_1, z_2, y) to $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$ such that (z, z_1, z_2) has positive entries and $\nexists e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A : \ z_1(e) = z_2(e)$, either there exists an element $e \in A$ with $z_e = 1$ or an element $e \in \mathcal{S}_1 \setminus A$ with $z_1(e) = 1$ or an element $e \in \mathcal{S}_2 \setminus A$ with $z_2(e) = 1$.*

Proof. Suppose for a contradiction $0 < z(e) < 1$ for each $e \in A$ and $0 < z_1(e) < 1$ for each $e \in \mathcal{S}_1 \setminus A$ and $0 < z_2(e) < 1$ for each $e \in \mathcal{S}_2 \setminus A$ **(1)**. By Lemma 4.1.6, we obtain two chains C_1, C_2 defining z, z_1, z_2. We now show a contradiction to the fact that $|C_1| + |C_2| = |\mathcal{S}_1| + |\mathcal{S}_2| - |A|$ by a counting argument. We give two tokens to each element in $\mathcal{S}_1 \cap \mathcal{S}_2$ and one token to each element in $(\mathcal{S}_2 \cup \mathcal{S}_2) \setminus (\mathcal{S}_1 \cap \mathcal{S}_2)$ for a total of $|\mathcal{S}_1| + |\mathcal{S}_2|$ tokens. Now, we distribute these tokens to the sets of the two chains. This is done as follows. Each element assigns one token to the smallest set $T_i \in C_i$ such that $e \in Ti$ for $i = \{1, 2\}$. We now claim that each set in $C_1 \cup C_2$ obtains at least two tokens. The

Algorithm 6 Iterative Algorithm for Partial Zero Switch Multistage Matroid Maintenance for T=2

```
$B_1 \leftarrow \emptyset$, $B_2 \leftarrow \emptyset$.
**while** $\mathcal{S}_1 \cup \mathcal{S}_2 \neq \emptyset$ **do**
    Find an optimal extreme point solution $(z, z_1, z_2, y)$ to $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$.

    If there is an $e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$ with $z_1(e) = z_2(e)$, then update $A \leftarrow A \cup \{e\}$.
    Else if there is an $e \in A$ with $z(e) = 0$, delete it from both matroids and from A.
    Else if there is an $e \in \mathcal{S}_1 \setminus A$ with $z_1(e) = 0$, delete it from $\mathcal{M}_1$.
    Else if there is an $e \in \mathcal{S}_2 \setminus A$ with $z_2(e) = 0$, delete it from $\mathcal{M}_2$.
    Else if there is an $e \in A$ with $z(e) = 1$, then update $B_1 \leftarrow B_1 \cup \{e\}$, $B_2 \leftarrow B_2 \cup \{e\}$,
  $\mathcal{M}_1 \leftarrow \mathcal{M}_1/e, \mathcal{M}_2 \leftarrow \mathcal{M}_2/e$, $A \leftarrow A \setminus \{e\}$.
    Else if there is an $e \in \mathcal{S}_1 \setminus A$ with $z_1(e) = 1$, then update $B_1 \leftarrow B_1 \cup \{e\}$, $\mathcal{M}_1 \leftarrow \mathcal{M}_1/e$
  and if also $e \in \mathcal{S}_2$ then set $a(e) \leftarrow 0$.
    Else if there is an $e \in \mathcal{S}_2 \setminus A$ with $z_2(e) = 1$, then update $B_2 \leftarrow B_2 \cup \{e\}$, $\mathcal{M}_2 \leftarrow \mathcal{M}_2/e$
  and if also $e \in \mathcal{S}_1$ then set $c_1(e) \leftarrow c_1(e) - a(e)$.
return $B_1$, $B_2$.
```

argument is identical for sets in C_1, C_2. Let $T \in C_1$ and R be the largest set in C_1 such that $R \subseteq T$. Now, we have $z_1(T \setminus A) + z(T \cap A) = r_1(T)$ and $z_1(R \setminus A) + z(R \cap A) = r_1(R)$. Subtracting, we obtain $z_1((T \setminus R) \setminus A) + z((T \setminus R) \cap A) = r_1(T) - r_1(R)$. If $T \setminus R = \emptyset$ then T = R and we have a contradiction to the linear independence of the constraints. Also, $r_1(T) - r_1(R)$ is an integer, so $z_1((T \setminus R) \setminus A) + z((T \setminus R) \cap A)$ is an integer and from **(1)**, we have that $|T \setminus R| \geq 2$. Thus, T receives one token for each element in $T \setminus R$ for a total of at least two tokens. Therefore, every set in $C_1 \cup C_2$ receives at least two tokens. The distributed tokens were $|\mathcal{S}_1| + |\mathcal{S}_2|$ and from lemma 4.1.6 the received tokens are at least $2(|\mathcal{S}_1| + |\mathcal{S}_2| - |A|)$. So,

$$|\mathcal{S}_1| + |\mathcal{S}_2| \geq 2(|\mathcal{S}_1| + |\mathcal{S}_2| - |A|) \Rightarrow 2|A| \geq |\mathcal{S}_1| + |\mathcal{S}_2| \Rightarrow A = \mathcal{S}_1 = \mathcal{S}_2$$

Thus, each element gave two tokens and each set must have received exactly two tokens. Now, we show that there is at least one extra token which is a contradiction. First of all, if an element does not belong to any set of a specific chain, then its corresponding token goes nowhere and we are done. So, it must be the case that the maximal set of each chain contains all the elements, which means that $\mathcal{S}_1 = \mathcal{S}_2 = A$ belongs to both chains, contradiction because of the linear independence. □

*We should note that if (z, z_1, z_2, y) is an extreme point of $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$, then $y(e) = max(0, z_2(e) - z_1(e))$ for all $e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$ **(2)** and the argument is the same as the one we presented for the case of partition matroids. Now, the main theorem is proved via the standard induction argument. However, because of the many cases, the proof is lengthy and we present it in detail for completeness.*

Theorem 4.1.8. *The optimal solution of the $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$ is integral*

Proof. This is proved by induction on the number of iterations of the algorithm. The base case is trivial to verify. Let $\mathcal{M}_1 = (\mathcal{S}_1, \mathcal{I}_1)$, $\mathcal{M}_2 = (\mathcal{S}, \mathcal{I}_2)$, A denote the matroids and the *zero switch set* in the current iteration and (z, z_1, z_2, y) the optimal LP solution.

If there is an element $e^* \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$ with $z_1(e^*) = z_2(e^*)$ then from **(2)**, $y(e^*) = 0$. So, if we add to the LP the additional constraints $z_1(e^*) = z_2(e^*)$ and $y(e^*) = 0$, the optimal solution does not change. This new LP is equivalent to $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A + e^*)$. This is the LP that the algorithm solves at the next iteration. From the induction hypothesis, we will find bases B_1, B_2 of $\mathcal{M}_1, \mathcal{M}_2$ that respect $A + e^*$ and are optimal solution for $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A+e^*)$. Thus, because of the aforementioned equivalence and the fact that B_1, B_2 respect A, they are optimal solution for $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$

Else if there is an element $e^* \in A$ with $z(e^*) = 0$, then if we add to the LP the additional constraint $z(e^*) = 0$, the optimal solution does not change. This new LP is equivalent to $LP_{pzs}(\mathcal{M}_1 - e^*, \mathcal{M}_2 - e^*, A - e^*)$. This is the LP that the algorithm solves at the next iteration. From the induction hypothesis, we will find bases B_1, B_2 of $\mathcal{M}_1 - e^*, \mathcal{M}_2 - e^*$ that respect $A - e^*$ and are optimal solution for $LP_{pzs}(\mathcal{M}_1 - e^*, \mathcal{M}_2 - e^*, A - e^*)$. Observe that these are also bases of $\mathcal{M}_1, \mathcal{M}_2$ respectively and respect A. Hence, because of the eqivalence, they are optimal solution for $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$.

Else if there is an element $e^* \in \mathcal{S}_1 \setminus A$ with $z_1(e^*) = 0$, we distinguish two cases. First case: $e^* \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$. Then, from **(2)**, $y(e^*) = z_2(e^*)$. So, adding to the LP, the constraints $z_1(e^*) = 0$ and $y(e^*) = z_2(e^*)$, the optimal solution does not change. This new LP is equivalent to $LP_{pzs}(\mathcal{M}_1 - e^*, \mathcal{M}_2, A)$. This is the LP that the algorithm solves at the next iteration. From the induction hypothesis, we will find bases B_1, B_2 of $\mathcal{M}_1 - e^*, \mathcal{M}_2$ that respect A and are optimal solution for $LP_{pzs}(\mathcal{M}_1 - e^*, \mathcal{M}_2, A)$. Observe that B_1 is a base of $\mathcal{M}_1$. Hence, because of the eqivalence, they are optimal solution for $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$. Second case: $e^* \in \mathcal{S}_1 \setminus \mathcal{S}_2$. Adding to the LP, the constraint $z_1(e^*) = 0$ the optimal solution does not change. This new LP is equivalent to $LP_{pzs}(\mathcal{M}_1 - e^*, \mathcal{M}_2, A)$ and we proceed as before.

Else if there is an element $e^* \in \mathcal{S}_2 \setminus A$ with $z_2(e^*) = 0$, we again distinguish two cases. First case: $e^* \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$. Then, from **(2)**, $y(e^*) = 0$. So, adding to the LP, the constraints $z_1(e^*) = 0$ and $y(e^*) = 0$, the optimal solution does not change. This new LP is equivalent to $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2 - e^*, A)$ and we proceed as in the case $z_1(e^*) = 0$ (and for the second case too).

Else if there is an element $e^* \in A$ with $z(e^*) = 1$, then if we add to the LP the additional constraint $z(e^*) = 1$, the optimal solution does not change. This new LP is equivalent to the LP:

$$\begin{aligned} \text{minimize} \quad & (c_1(e^*) + c_2(e^*)) + \sum_{e \in A - e^*} (c_1(e) + c_2(e))z(e) + \sum_{e \in \mathcal{S}_1 \setminus A} c_1(e)z_1(e) + \sum_{e \in \mathcal{S}_2 \setminus A} c_2(e)z_2(e) + \\ & \sum_{e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A} a(e)y(e) + \sum_{e \in \mathcal{S}_2 \setminus \mathcal{S}_1} a(e)z_2(e) \\ \text{subject to} \quad & (z', z_1, z_2, y) \textit{ is a feasible solution of } LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2/e^*, A - e^*) \end{aligned} \tag{4.2}$$

where z' is the restriction of z to $A - e^*$. But the objective function of this LP is the objective funtion of $LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2/e^*, A - e^*)$ plus $(c_1(e^*) + c_2(e^*))$. But, $LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2/e^*, A - e^*)$ is the LP that the algorithm solves at the next iteration. From the induction hypothesis, we will find bases B_1, B_2 of $\mathcal{M}_1/e^*, \mathcal{M}_2/e^*$ that respect $A - e$ and are optimal solution for $LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2/e^*, A - e^*)$. $B_1 + e^*$, $B_2 + e^*$ are bases of $\mathcal{M}_1, \mathcal{M}_2$, respect A and have cost the optimal cost of $LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2/e^*, A - e^*)$ plus $(c_1(e^*) + c_2(e^*))$. Thus, they are optimal solution for $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$.

Else if there is an element $e^* \in \mathcal{S}_1 \setminus A$ with $z_1(e^*) = 1$, then we distinguish two cases. First case: $e^* \in \mathcal{S}_2$. Then, $e^* \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$ and from **(2)**, $y(e^*) = 0$. So, adding to the LP the constraints $z_1(e^*) = 1$, $y(e^*) = 0$, the optimal solution does not change. This new LP is equivalent to the LP:

$$\begin{aligned} \text{minimize} \quad & c_1(e^*) + \sum_{e \in A} (c_1(e) + c_2(e))z(e) + \sum_{e \in (\mathcal{S}_1 - e*) \setminus A} c_1(e)z_1(e) + \sum_{e \in \mathcal{S}_2 \setminus A} c_2(e)z_2(e) + \\ & \sum_{e \in ((\mathcal{S}_1 \cap \mathcal{S}_2) - e^*) \setminus A} a(e)y(e) + \sum_{e \in \mathcal{S}_2 \setminus \mathcal{S}_1} a(e)z_2(e) \\ \text{subject to} \quad & (z, z_1', z_2, y') \textit{ is a feasible solution of } LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2, A) \end{aligned} \tag{4.3}$$

where z_1' is the restriction of z_1 to $(\mathcal{S}_1 \setminus A) - e^*$ and y' is the restriction of y to $((\mathcal{S}_1 \cap \mathcal{S}_2) - e^*) \setminus A$. But the objective function of this LP is $c_1(e^*)$ plus the objective funtion of $LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2, A)$ where $a(e^*)$ is updated to zero, say LP_1. But, LP_1 is the LP that the algorithm solves at the next iteration. From the induction hypothesis, we will find bases B_1, B_2 of $\mathcal{M}_1/e^*, \mathcal{M}_2$ that respect A and are optimal solution for LP_1. $B_1 + e^*$ is a basis of $\mathcal{M}_1$, $B_1 + e^*$ and B_2 respect A and have cost the optimal cost of LP_1 plus $c_1(e^*)$. Thus, from the aforementioned equivalence, they are optimal solution for $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$. Second case: $e^* \notin \mathcal{S}_2$. Then adding to the LP the constraint $z_1(e^*) = 1$, the optimal solution does not change. This new LP is equivalent to the LP:

$$\begin{aligned} \text{minimize} \quad & c_1(e^*) + \sum_{e \in A} (c_1(e) + c_2(e))z(e) + \sum_{e \in (\mathcal{S}_1 - e*) \setminus A} c_1(e)z_1(e) + \sum_{e \in \mathcal{S}_2 \setminus A} c_2(e)z_2(e) + \\ & \sum_{e \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A} a(e)y(e) + \sum_{e \in \mathcal{S}_2 \setminus \mathcal{S}_1} a(e)z_2(e) \\ \text{subject to} \quad & (z, z_1', z_2, y) \textit{ is a feasible solution of } LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2, A). \end{aligned} \tag{4.4}$$

where z_1' is defined as before. But the objective function of this LP is $c_1(e^*)$ plus the objective funtion of $LP_{pzs}(\mathcal{M}_1/e^*, \mathcal{M}_2, A)$, which is the LP that the algorithm solves at the next iteration. Using the induction hypothesis, we proceed as before.

Else if there is an element $e^* \in \mathcal{S}_2 \setminus A$ with $z_2(e^*) = 1$, then we again distinguish two cases. First case: $e^* \in \mathcal{S}_1$. Then, $e^* \in (\mathcal{S}_1 \cap \mathcal{S}_2) \setminus A$ and from **(2)**, $y(e^*) = 1 - z_1(e^*)$. So, adding to the LP the constraints $z_2(e^*) = 1$, $y(e^*) = 1 - z_1(e^*)$, the optimal solution does not change. This new LP is equivalent to the LP:

$$\begin{aligned} \text{minimize} \quad & c_2(e^*) + \sum_{e\in A}(c_1(e)+c_2(e))z(e) + \sum_{e\in \mathcal{S}_1\setminus A} c_1(e)z_1(e) + \sum_{e\in(\mathcal{S}_2-e^*)\setminus A} c_2(e)z_2(e) + \\ & \sum_{e\in((\mathcal{S}_1\cap\mathcal{S}_2)-e^*)\setminus A} a(e)y(e) + a(e^*)(1-z_1(e^*)) + \sum_{e\in \mathcal{S}_2\setminus\mathcal{S}_1} a(e)z_2(e) \\ \text{subject to} \quad & (z, z_1, z_2', y') \ is\ a\ feasible\ solution\ of\ LP_{pzs}(\mathcal{M}_1, M_2/e^*, A) \end{aligned} \tag{4.5}$$

where z_2' is the restriction of z_2 to $(\mathcal{S}_2 \setminus A) - e^*$ and y' is the restriction of y to $((\mathcal{S}_1 \cap \mathcal{S}_2) - e^*) \setminus A$. But the objective function of this LP is equal to $c_2(e^*)$ plus the objective funtion of $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2/e^*, A)$ where $c_1(e^*)$ is updated to $c_1(e^*) - a(e^*)$, say LP_2. But, LP_2 is the LP that the algorithm solves at the next iteration. From the induction hypothesis, we will find bases B_1, B_2 of $\mathcal{M}_1, \mathcal{M}_2/e^*$ that respect A and are optimal solution for LP_2. $B_2 + e^*$ is a basis of $\mathcal{M}_2$, B_1 and B_2/e^* respect A and have cost at most the optimal cost of LP_2 plus $c_2(e^*) + a(e^*)$. Thus, from the aforementioned equivalence, they are optimal solution for $LP_{pzs}(\mathcal{M}_1, \mathcal{M}_2, A)$. Second case: $e^* \notin \mathcal{S}_1$. For this case the induction is proved exactly as in the case where $z_1(e^*) = 1$ and $e^* \notin \mathcal{S}_2$.

□

4.2 Uniform switching costs: Division into "epochs" and an Iterative Rounding algorithm

We have seen that the logarithmic approximation is optimal for MMM. However, the reduction from set cover and the integrality gap instance rely heavily on the fact that the elements may have different acquisition costs. Thus, it is reasonable to ask whether the case of uniform acquisition costs is hard and whether there are sub-logarithmic approximations for it, as in the case of Dynamic Facility Location, see [12]. The MMM with uniform acquisition costs can be proven to be $APX-hard$ for the case of graphical matroids [Fotakis, Lampis, Paschos, Plevrakis '17]. In this thesis, we make a first step towards a constant approximation algorithm, by presenting an algorithm that has constant approximation at the holding cost and logarithmic approximation at the acquisition cost.

The LP is the same as in the general case, presented at chapter 3, but here all the acquisition costs are equal to g. We call this LP: $LP_{MMM}(\mathcal{M}, [1, T])$.

$$\begin{array}{ll} \textit{minimize} & \sum_{t,e} c_t(e) z_t(e) + g \sum_{t,e} y_t(e) \\ \textit{subject to} & \vec{z_t} \in \mathcal{P}_B(\mathcal{M}), \quad \forall t \in [T] \\ & y_t(e) \geq z_t(e) - z_{t-1}(e), \quad \forall e \in E, \quad \forall t \in [T] \\ & y_t(e), z_t(e) \geq 0, \ \forall e \in E, \quad \forall t \in [T] \end{array}$$

Let $(z_t^*(e), y_t^*(e))_{e,t}$ be the optimal LP solution. First of all, as we have seen in the algorithms in chapter 3, if T is very big, then the approximation ratio is unsatisfactory. However, let's suppose that $(z_t^*(e), y_t^*(e))_{e,t}$ has payed acquisition cost $gr/2$ in the time interval [1,t']. In this case, if we make independent rounding of the fractional solution at the intervals [1,t'],[t'+1,T], the acquisition cost at time t'+1 will be at most gr, so it can be charged at the acquisition cost that $(z_t^*(e), y_t^*(e))_{e,t}$ has payed during [1,t'], i.e we lose at most a constant factor at the acquisition cost. Based on this idea, we present a process that takes as input $(z^t, y^t)_{t\in[t_1,t_2]}$, a feasible solution of the $LP_{MMM}(\mathcal{M}, [t_1, t_2])$, $1 \leq t_1 \leq t_2 \leq T$, which is the restriction of the $LP_{MMM}(\mathcal{M}, [1, T])$ at the interval $[t_1, t_2]$. This process outputs a division of $[t_1, t_2]$ into "epochs" and a vector w_i for each "epoch" i.

Algorithm 7 Division into epochs of $(z^t, y^t)_{t\in[t_1,t_2]}$, for matroid $\mathcal{M}$

```
$r = r_{\mathcal{M}}(\mathcal{S})$
$j = 1$
$s = t_1$
for $t = t_1$ to $t_2$ do
    $b_t(e) = min_{s\leq u\leq t} z_u(e)$
    if $b_t(\mathcal{S}) < \frac{r}{2}$ then
        $epoch_j = [s, t-1]$
        $w_j = b_{t-1}$
        $s = t$
        $j{+}{+}$
return $\{epoch_i, w_i\}_{i=1}^{j}$
```

Observe that w_i is the "fractional intersection" of $\{z_t\}_{t\in epoch_i}$. Now, suppose we want to round $(z^t, y^t)_{t\in[t_1,t_2]}$, which has acquisition cost γ. It is easy to see that if we make the rounding independently in each epoch and after we concatenate the solutions, this concatenation will cost at most 2γ at the acquisition cost. So now it remains to round each epoch. Let's focus on a specific epoch, say epoch i. From construction, $w_i(\mathcal{S}) \geq r_{\mathcal{M}}(\mathcal{S})/2$. Hence, $\{z_t\}_{t\in epoch_i}$ have a lot in common. More specifically, as we will show, there exists a set $A \subseteq \mathcal{S}$, such that $|A| = \Omega(r_{\mathcal{M}}(\mathcal{S}))$ and if the entries of w_i that correspond to elements of A are multiplied by at most a constant factor, the restriction of w_i in the set A becomes a vector that lies inside the base polytope of the restriction of $\mathcal{M}$ in the set A. Also, this set A can be computed in polynomial time. To compute it, first consider the following LP, *that takes as input a matroid and a vector $w \in P_I(\mathcal{M})$ and outputs a vector $x \in P_I(\mathcal{M})$*

$$\begin{array}{ll} \textit{maximize} & x(\mathcal{S}) \\ \textit{subject to} & x(T) \leq r(T), \forall T \subseteq \mathcal{S} \\ & 0 \leq x(e) \leq \alpha w_i(e), \forall e \in E \end{array}$$

for some parameter $\alpha > 2$ that we will fix later. We call this linear program $LP_{extr}(\mathcal{M}, w)$, because it will help us extract the set A. Let x be the optimal solution of $LP_{extr}(\mathcal{M}, w)$. From lemma 2.3.1 there is a unique inclusion wise maximum tight set of x, say $C_x \subseteq \mathcal{S}$. Let $r = r_{\mathcal{M}}(\mathcal{S})$.

Lemma 4.2.1. *If $w(\mathcal{S}) \geq \frac{r}{\beta}$, for a $\beta > 0$, then $r(C_x) \geq \frac{r}{\beta}\frac{\alpha-\beta}{\alpha-1}$.*

Proof. From lemma 2.3.1 every element that belongs to some tight set of x also belongs in C_x. Thus, because of the objective, $x(e) = \alpha w(e)$, $\forall e \in \mathcal{S} \setminus C_x$ (otherwise we could increase x(e) without violating any constraint).

We have that

$$r \geq x(\mathcal{S} \setminus C_x) + x(C_x) \ (1)$$
$$w(\mathcal{S} \setminus C_x) + w(C_x) \geq \frac{r}{\beta} \ (2)$$
$$x(\mathcal{S} \setminus C_x) = \alpha w(\mathcal{S} \setminus C_x) \ (3)$$
$$w(C_x) \leq r(C_x) \ (4)$$
$$x(C_x) = r(C_x) \ (5)$$

(1) holds since $x \in P_I(\mathcal{M})$. (2) holds by assumption. (3) is implied by the observation at the beginning of the proof. (4) holds, since $w \in P_I(\mathcal{M})$. (5) holds by definition. From (1),(3),(5) we get that:

$$r \geq \alpha w(\mathcal{S} \setminus C_x) + r(C_x) \Rightarrow$$
$$\frac{r - r(C_x)}{\alpha} \geq w(\mathcal{S} \setminus C_x) \ (6)$$

From (2), (4) we get that:

$$w(\mathcal{S} \setminus C_x) \geq \frac{r}{\beta} - r(C_x) \ (7)$$

From (6), (7) we get that:

$$\frac{r - r(C_x)}{\alpha} \geq \frac{r}{\beta} - r(C_x) \Rightarrow$$
$$(1 - \frac{1}{\alpha}) r(C_x) \geq \frac{r}{\beta} - \frac{r}{\alpha} \Rightarrow$$
$$r(C_x) \geq \frac{r}{\beta}\frac{\alpha - \beta}{\alpha - 1}$$

□

Algorithm 8 Computation of C_x

```
$A \leftarrow \emptyset$
while $min_{T\subseteq\mathcal{S}}(r(T) - x(T)) = 0$ do
    $T^* \leftarrow argmin_{T\subseteq\mathcal{S}}(r(T) - x(T))$
    $A \leftarrow A \cup T^*$
    $\mathcal{M} \leftarrow \mathcal{M}/T^*$
    $x \leftarrow$ restriction of x in $\mathcal{S} \setminus T^*$
return $A$
```

The algorithm uses an algorithm that minimizes submodular functions (see [37]) as oracle, takes as input the $x \in P_I(\mathcal{M})$ and outputs C_x in polynomial time.

Lemma 4.2.2. *The algorithm outputs C_x.*

Proof. Let $x, \mathcal{M}$ at some iteration of the algorithm such that $x \in P_I(\mathcal{M})$. Let r be the rank function of $\mathcal{M}$. Also, let T be a tight set of x and T^* be its maximum tight set. Now take $\mathcal{M}' = \mathcal{M}/T$ with rank function r' and x' be the restriction of x at $\mathcal{S} \setminus T$. We have that for each $A \subseteq \mathcal{S} \setminus T^*$ we have

$$x'(A) \le r'(A) \Leftrightarrow x(A \cup T) - x(T) \le r(A \cup T) - r(T) \Leftrightarrow x(A \cup T) \le r(A \cup T)$$

Thus, $x' \in P_I(\mathcal{M}/T)$ and A is a tight set of x' iff $T \cup A$ is a tight set of x. Hence, $T^* \setminus T$ is the maximum tight set of x'. The lemma follows.

□

Now, since $w_i \in P_I(\mathcal{M})$ and $w_i(\mathcal{S}) \ge r/2$, by solving the $LP_{extr}(\mathcal{M}, w_i)$ for $\alpha = 3$, we get the optimal solution x_i and thus we have the following corollary:

Corollary 4.2.2.1. *$r(C_{x_i}) \ge \frac{r}{4}$ and C_{x_i} can be computed in polynomial time.*

Now, for $t \in epoch_i$, let

$$\hat{z}_t(e) = \begin{cases} x_i(e) & \text{if } e \in C_x \\ z_t(e) & \text{otherwise} \end{cases}$$

From lemma 1.1, $\{\hat{z}_t\}_{t\in epoch_i}$, restricted in C_{x_i}, are all fractional bases of $\mathcal{M}$ restricted in C_{x_i}. Also $\{\hat{z}_t\}_{t\in epoch_i}$ are all in the spanning set polytope, see [37]. Now, let H_i be the minimum weight basis of $\mathcal{M}$ ***restricted in*** *C_{x_i} with weight function $\sum_{t\in epoch_i} c_t$. For $t \in epoch_i$, let*

$$\tilde{z}_t(e) = \begin{cases} \mathbf{1}\{e \in H_i\} & \text{if } e \in C_{x_i} \\ z_t(e) & \text{otherwise} \end{cases}$$

We have that

$$\sum_{t\in epoch_i} \sum_{e\in C_{x_i}} c_t(e)\tilde{z}_t(e) \le \sum_{t\in epoch_i} \sum_{e\in C_{x_i}} c_t(e)\hat{z}_t(e) \le 3 \sum_{t\in epoch_i} \sum_{e\in C_{x_i}} c_t(e)w_i(e) \le 3 \sum_{t\in epoch_i} \sum_{e\in C_{x_i}} c_t(e)z_t(e)$$

Also, the acquisition cost of $\{\tilde{z}_t\}_{t\in epoch_i}$ is lower or equal to that of $\{z_t\}_{t\in epoch_i}$. What we want now is to transform $\{\tilde{z}_t\}_{t\in epoch_i}$ into a fractional base of $\mathcal{M}$ (remove the fractional circuits) without changing anything inside C_x and without increasing neither the variables nor the acquisition cost.

From now on, if $z \in \mathcal{P}_B(\mathcal{M})$, then $\mathcal{F}_z = \{\chi(T) :\ T \subseteq \mathcal{S}$ and $z(T) = r(T)\}$, i.e $\mathcal{F}_z$ is the set of characteristic vectors of the tight sets of z. Before we proceed, we should note that one of the most important matroid properties is the ability to move from one base to another by replacing elements, one by one, in the base with others outside of it so that the whole time we maintain a base. Is this possible if we want to move from a fractional base to another? Towards answering this question we first introduce a very useful definition.

Definition 23. *Let $z \in \mathcal{P}_B(\mathcal{M})$, $e \in \mathcal{S}$. We define $mts_z(e)$ to be the smallest set $T \subseteq \mathcal{S}$ such that $e \in T$ and $\chi(T) \in F_z$, that is, T is the minimum tight set that contains e. When z is clear from context we will simply write $mts(e)$.*

We remark that $mts(e)$ always exists, since $\mathcal{S}$ itself is tight, and it is always unique, because of lemma 2.3.1. Also note that if $z(e) = 1$ then $mts(e) = e$. Also observe that $\forall e' \in mts(e) - e,\ z(e') > 0$, from the definition of mts(e) and the monotonicity of the rank function. The reason that we present this definition is that the elements of $mts(e) - e$ are the ones that we can decrease, at least slightly, in order to increase $z(e)$ while remaining inside $\mathcal{P}_B(\mathcal{M})$, i.e while maintaining a fractional base. This is shown rigorously via the following proposition.

Proposition 4.2.2.1. *Let $e' \in mts(e) - e$, where $z(e) < 1$ and e is not a loop. If we start increasing $z(e)$ and decreasing $z(e')$ with the same constant rate, then $z(e)$ remains feasible until e' is no longer in $mts(e)$.*

Proof. Let $mts(e)$ be the minimum tight set of e before we make any changes. Observe that since $z(e) < 1$ and e is not a loop, $mts(e) \neq \{e\}$. At the beginning, all elements of mts(e), except of e (maybe), correspond to positive entries of z. Now, let's fix an element $e' \in mts(e)$. At the beginning, from definition: $\forall T :\ \chi(T) \in \mathcal{F}_z$ and $e \in T \Rightarrow mts(e) \subseteq T \Rightarrow e' \in T$. Hence, since we make the two changes at the same rate, all the tight sets remain tight. Thus, this process will stop at the moment when at least one of these happens: $z(e)$ becomes 1 or $z(e')$ becomes 0 or a set T^* such that $e \in T^*$, $e' \notin T^*$ becomes tight. Let $mts'(e)$ be the minimum tight set of e at that moment. In the first case, $mts'(e) = \{e\} \subseteq mts(e) - e'$. In the second case, $mts'(e) \subseteq mts(e) - e'$. In the the third case, from lemma 2.3.1, $mts'(e) \subseteq T^* \cap mts(e) \subseteq mts(e) - e'$. The proposition follows. □

Now, we return to the question of how to transform $\{\tilde{z}_t\}_{t\in epoch_i}$ into a fractional base of $\mathcal{M}$ without changing anything inside C_x and without increasing neither the variables nor the acquisition cost. For the next lemma we write $mts_t(e)$ instead of $mts_{z_t}(e)$, for $e \in \mathcal{S}, t \in epoch_i$.

Lemma 4.2.3. *Let $\{x_t\}_t \in I$, where I is a time interval, be a feasible solution of the $LP_{MMM}(\mathcal{M}, I)$. Let $e \in \mathcal{S}$ and e is not a loop. There is a $\{x'_t\}_{t\in I}$ such that:*

$$x'_t(e) = 1, \forall t \in I,$$

$$x'_t(e_1) \le x_t(e_1), \forall t \in I, \forall e_1 \neq e$$

and the acquisition cost of $\{x'_t\}_{t\in I}$ is not bigger than that of $\{x_t\}_{t\in I}$.

Proof. First, let's prove the follwing proposition (notation: for the restriction of a vector v on a subset A of the elements, we write $\vec{v}(A)$):

Proposition 4.2.3.1. *Let $\{x_t\}_t \in I$ is a feasible solution of the $LP_{MMM}(\mathcal{M}, I)$ and t is a time step such that $t, t+1 \in I$ and $\vec{x_t}(mts_t(e)) \neq \vec{x_{t+1}}(mts_t(e))$, where by $mts_t(e)$, we mean $mts_{x_t(e)}$. Then, $\exists e' \in \mathcal{S}$, $e' \neq e$ and either*

$$e' \in mts_t(e) \text{ and } x_t(e') > x_{t+1}(e') \text{ or } e' \in mts_{t+1}(e) \text{ and } x_{t+1}(e') > x_t(e')$$

Proof. Suppose $x_t(e) \le x_{t+1}(e)$ (for the other case the argument is identical). Clearly, $x_t(mts_t(e)) = r(mts_t(e))$ and $x_{t+1}(mts_t(e)) \le r(mts_t(e)) \Rightarrow x_t(mts_t(e) - e) + x_t(e) \ge x_{t+1}(mts_t(e) - e) + x_{t+1}(e)$. Thus, the proposition follows from the hypothesis and our assumption. □

Now, we are ready to construct $\{x'_t\}_{t\in I}$, from $\{x_t\}_t \in I$. While $\exists t \in I$ such that $\vec{x_t}(mts_t(e)) \neq \vec{x_{t+1}}(mts_t(e))$ (which implies that not both $x_t(e), x_{t+1}(e)$ are equal to one), then from proposition 4.2.3.1, $\exists e' \in mts_t(e)$, $e' \neq e$, such that $x_t(e') > x_{t+1}(e')$ or $\exists e' \in mts_{t+1}(e)$ such that $x_{t+1}(e') > x_t(e')$. Let's say that the first case occurs. In this case we change x_t (otherwise we do the same thing but we change x_{t+1}). We start increasing $x_t(e)$ and decreasing $x_t(e')$ at the same constant rate until either the $mts_t(e)$ becomes smaller or $x_t(e') = x_{t+1}(e')$ (as long as neither of the two happens, the solution remains feasible as proposition 4.2.2.1 indicates).

Clearly, when this first phase finishes, the $mts_t(e)$ will be the same at all time steps. Now, we simultaneously change all $\{x_t\}_t \in I$: while not all $x_t(e) = 1$, we find an e' such that $mts_t(e') > 0, \forall t \in I$ and we start increasing $x_t(e)$ and decreasing $x_t(e')$ at the same rate, $\forall t \in I$, until the $mts_t(e)$ becomes smaller (again until this happens, the solution remains feasible). It is easy to see that the two phases will terminate after a finite number of steps. At the end, the vector has the properties of the lemma.

□

Corollary 4.2.3.1. *For each i, there is a $\{z'_t\}_{t\in epoch_i}$ such that:*

$$z'_t(e) = \mathbf{1}\{e \in H_i\}, \forall e \in C_{x_i}, t \in epoch_i$$

$$z'_t(e) \le z_t(e), \forall e \in \mathcal{S} \setminus C_{x_i}, t \in epoch_i$$

and the acquisition cost of $\{z'_t\}_{t\in epoch_i}$ in $\mathcal{S} \setminus C_{x_i}$ is not bigger than that of $\{z_t\}_{t\in epoch_i}$ in $\mathcal{S} \setminus C_{x_i}$.

Proof. For each i, let $H_i = \{e_1^i, ..., e_k^i\}$. We iteratively use lemma 4.2.3.1, for $I = epoch_i$, initial solution the $\{z_t\}_{t \in epoch_i}$ and initial matroid the $\mathcal{M}$, for each e_j^i, j=1,...,k. Once we apply this lemma for element e_j^i, we update $\mathcal{M} \leftarrow \mathcal{M}/e_j^i$, we restrict all $\{z_t\}_{t \in epoch_i}$ in $\mathcal{S} - e_j^i$ and reapply the lemma for e_{j+1}^i etc. □

Now, note that since for each epoch i, there is such a $\{z_t'\}_{t \in epoch_i}$, it can be computed in polynomial time via the $LP_{MMM}(\mathcal{M}/C_{x_i}, epoch_i)$ ***with the additional constraints:***

$$z_t'(e) \leq z_t(e), \forall e \in \mathcal{S} \setminus C_{x_i}, t \in epoch_i$$

All this analysis yields the following theorem:

Theorem 4.2.4. *There is a polynomial time algorithm that takes as input a matroid $\mathcal{M} = (\mathcal{S}, \mathcal{I})$, $r = r_{\mathcal{M}}(\mathcal{S})$, a time interval $[t_1, t_2]$ and a $\{z_t\}_{t \in [t_1, t_2]} \in P_B(\mathcal{M}, [t_1, t_2])$, and outputs*

1. *A division of $[t_1, t_2]$ into disjoint epochs $[f_0, f_1], [f_1 + 1, f_2], ..., [f_{k-1} + 1, f_k]$ ($f_0 = t_1, f_k = t_2$), such that for all $i = 1, ..., k-1$, $\{z_t\}_{t \in [t_1, t_2]} \in P_B(\mathcal{M}, [t_1, t_2])$ has at least r/2 acquisition cost inside each $[f_{i-1}, f_i + 1]$.*

2. *An $H_i \in \mathcal{I}$ such that $|H_i| \geq r/4$, $i = 1, ..., k$.*

3. *$\{z_t'\}_{t \in epoch_i} \in P_B(\mathcal{M}/H_i, epoch_i)$, $i = 1, ..., k$.*

We also have that

$$\sum_{t \in epoch_i} w_t(H_i) \leq 3 \sum_{t \in epoch_i} \sum_{e \in span(H_i)} w_t(e) z_t(e), \ \ for\ all\ i = 1, ..., k$$

$$z_t'(e) \ \leq z_t(e), \ \ for\ all\ t \in epoch_i, e \in \mathcal{S} \setminus span(H_i), i = 1, ..., k$$

and finally, the acquisition cost of $\{z_t'(e)\}_{t \in epoch_i, e \in \mathcal{S} \setminus span(H_i)}$ is not greater than the acquisition cost of $\{z_t(e)\}_{t \in epoch_i, e \in \mathcal{S} \setminus span(H_i)}$. We refer to this algorithm as $\mathcal{F}(\mathcal{M}, [t_1, t_2], \{z_t\}_{t \in [t_1, t_2]})$.

Now it is clear to see how the main algorithm will be:

Algorithm 9 Algorithm for MMM with uniform acquisition costs

```
$B_t \leftarrow \emptyset, \ t = 1, ..., T$
Solve $LP_{MMM}(\mathcal{M}, [1, T])$ and get the optimal solution $\{z_t\}_{t \in [1, T]}$.
run $\mathcal{A}(\mathcal{M}, [1, T], \{z_t\}_{t \in [1, T]})$.
return $B_1, ..., B_T$
```

where $\mathcal{A}(\mathcal{M}, [t_1, t_2], \{z_t\}_{t \in [t_1, t_2]})$ is presented below:

Algorithm 10 $\mathcal{A}(\mathcal{M}, [t_1, t_2], \{z_t\}_{t\in[t_1,t_2]})$

```
if $r_{\mathcal{M}}(\mathcal{S}) = 0$, then return
run $\mathcal{F}(\mathcal{M}, [t_1, t_2], \{z_t\}_{t\in[t_1,t_2]})$ and get the epochs (suppose they are k) $\{epoch_i\}_{i=1}^k$, $\{H_i\}_{i=1}^k$, $\{\{z'_t(e)\}_{t\in epoch_i, e\in\mathcal{S}\setminus span(H_i)}\}_{i=1}^k$
for each $t \in [t_1, t_2]$ find $i \in [k]$ such that $t \in epoch_i$ and update $B_t \leftarrow B_t \cup H_i$
for each $i = 1, ..., k$ run $\mathcal{A}(\mathcal{M}/H_i, epoch_i, \{z'_t\}_{t\in epoch_i})$
```

Performance Guarantee

Theorem 4.2.5. *If $\{z_t\}_{t\in[1,T]}$ is the optimal solution of $LP_{MMM}(\mathcal{M}, [1, T])$ and $B_1, B_2, ..., B_T$ is the output of the algorithm then all of the outputed sets are bases of $\mathcal{M}$ and*

$$\sum_{t=1}^{T} w_t(B_t) \leq 3 \sum_{t=1}^{T} \sum_{e\in\mathcal{S}} w_t(e) z_t(e)$$

and

$$\sum_{t=2}^{T} |B_t \setminus B_{t-1}| = O(\log r) \sum_{t=2}^{T} \sum_{e\in\mathcal{S}} max(z_t(e) - z_{t-1}(e), 0)$$

Proof. Let's say that at the beginning of the algorithm $\{z'_t\}_{t\in[T]} = \{z_t\}_{t\in[T]}$. Now, let's consider $\{x_t\}_{t\in[T]}$ which is modified during the course of the algorithm and gradually becomes integral. More specifically,

$$x_t(e) = \mathbf{1}\{e \in B_t\}, \forall e \in span(B_t), t \in [T]$$

and

$$x_t(e) = z'_t(e), \forall e \in \mathcal{S} \setminus span(B_t), t \in [T]$$

from the property 3, of theorem 4.2.4, when the algorithm terminates, $\{x_t\}_{t\in[T]}$ is integral and it is the collection of the characteristic vectors of $B_1, B_2, ..., B_T$. From the property 2 and 3 of the theorem 4.2.4 $x_t \in P_B(\mathcal{M})$, $\forall t \in [T]$ always during the course of the algorithm. Thus, since due to the property 2 of the theorem 4.2.4 the algorithm will terminate (in polynomial time), the returned $B_1, ..., B_T$ are all bases of $\mathcal{M}$.

Now, we argue for the approximation on the holding cost. We actually prove that during the course of the algorithm:

$$\sum_{t\in[T]} \sum_{e\in span(B_t)} w_t(e)x_t(e) \leq 3 \sum_{t\in[T]} \sum_{e\in span(B_t)} w_t(e)z_t(e) \ \mathbf{(1)}$$

and

$$x_t(e) \leq z_t(e), \forall e \in \mathcal{S} \setminus span(B_t), t \in [T] \ \mathbf{(2)}$$

(2) follows directly from theorem 4.2.4. Before we present the proof of **(1)**, notice that the algorithm constructs an "interval tree", where the root is [1,T] and each node's children are the epochs that the algorithm constructs for this interval.

We will prove **(1)** via induction on the number of updates made at the collection of $\{B_t\}_{t\in[T]}$. The base case is trivial to verify (remember that at the beginning of the algorithm $\{z'_t\}_{t\in[T]} = \{z_t\}_{t\in[T]}$). Now, focus on some time that the algorithm updates B_t, by adding to it some set H. This update was made during some call of $\mathcal{A}$. When this call took place, t belonged to some "leaf" of the current "interval tree", say I. The algorithm runs $\mathcal{F}(\mathcal{M} \setminus B_t, I, \{z'_t(e)\}_{e\in\mathcal{M}/span(B_t),t\in I})$. The interval I is devided into epochs (these are added, conceptually, as children of I at the interval tree) and suppose $epoch_i$ is the epoch that contains t. So, along with B_t are updated all $\{B_{t'}\}_{t'\in epoch_i}$ and $H = H_i$ which is independent set of $\mathcal{M}/B_t$. Now, from theorem 4.2.4, for matroid $\mathcal{M}/span(B_t)$, interval I and $\{z'_t(e)\}_{e\in\mathcal{M}/span(B_t),t\in I} \in P_B(\mathcal{M}/span(B_t))$,

$$\sum_{t\in epoch_i} w_t(H_i) \leq 3 \sum_{t\in epoch_i} \sum_{e\in span_{\mathcal{M}/B_t}(H_i)} w_t(e)x_t(e)$$

so, from **(2)**:

$$\sum_{t\in epoch_i} w_t(H_i) \leq 3 \sum_{t\in epoch_i} \sum_{e\in span_{\mathcal{M}/B_t}(H_i)} w_t(e)z_t(e) \ \mathbf{(3)}$$

Now, $B_{t'} \leftarrow B_{t'} \cup H_i$, for all t' in $epoch_i$. Thus, after the update, the lefthand and the righthand side of **(3)** are parts of the lefthand and righthand side of **(1)**, respectively. Now, since this part of the lefthandside of **(1)** of is modified only once during the algorithm (because after this, in $epoch_i$, H_i is contracted from $\mathcal{M}/B_t$) and since the epochs partition I, when the updates of this call of $\mathcal{F}$ finish, **(1)** will still hold. The claim follows.

It remains to show the approximation at the acquisition cost. What we are going to show is that if we have a $\{z_t\}_{t\in[t_1,t_2]} : \ z_t \in P_B(\mathcal{M}), \ \forall t \in [t_1, t_2]$, a matroid $\mathcal{M}$ and we set $B_{t_1}, B_{t_1+1}, ..., B_{t_2} \leftarrow \emptyset$ and then run $\mathcal{A}(\mathcal{M}, [t_1, t_2], \{z_t\}_{t\in[t_1,t_2]})$ which produces, conceptually, an interval tree T with height h(T) and root $[t_1, t_2]$, then the acquisition cost of the returned $B_{t_1}, B_{t_1+1}, ..., B_{t_2}$ is at most $2h(T)g \sum_{t=t_1+1}^{t_2} \sum_{e\in\mathcal{S}} max(z_t(e) - z_{t-1}(e), 0)$. The proof is by induction on h(T). The base case h(T)=0 is trivial. From the way that the epochs are constructed, even if at the borders of $epoch_1, ..., epoch_k$, the returned integral solution changes entirely, the switching cost payed there is at most $2g \sum_{t=t_1+1}^{t_2} \sum_{e\in\mathcal{S}} max(z_t(e) - z_{t-1}(e), 0)$. The property 2 of theorem 4.2.4 and the induction hypothesis complete the induction. So, if T^* is the tree-interval of the algorithm that we run to solve the problem, then from property 2 of theorem 4.2.4, $h(T^*) = O(\log r)$ and we are done. □

Bibliography

[1] Jacob Abernethy et al. ≪A regularization approach to metrical task systems≫. In: *International Conference on Algorithmic Learning Theory.* Springer. 2010, pp. 270–284.

[2] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. ≪Dynamic facility location via exponential clocks≫. In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms.* Society for Industrial and Applied Mathematics. 2015, pp. 708–721.

[3] Matthew Andrews, Michel X Goemans, and Lisa Zhang. ≪Improved bounds for on-line load balancing≫. In: *Algorithmica* 23.4 (1999), pp. 278–301.

[4] Barbara M Anthony and Anupam Gupta. ≪Infrastructure leasing problems≫. In: *International Conference on Integer Programming and Combinatorial Optimization.* Springer. 2007, pp. 424–438.

[5] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. ≪Metrical task systems and the k-server problem on hsts≫. In: *Automata, Languages and Programming* (2010), pp. 287–298.

[6] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization.* Vol. 6. Athena Scientific Belmont, MA, 1997.

[7] Allan Borodin, Nathan Linial, and Michael E Saks. ≪An optimal on-line algorithm for metrical task system≫. In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 745–763.

[8] Thomas H Brylawski. ≪A decomposition for combinatorial geometries≫. In: *Transactions of the American Mathematical Society* 171 (1972), pp. 235–282.

[9] Niv Buchbinder et al. ≪Unified algorithms for online learning and competitive analysis≫. In: *Conference on Learning Theory.* 2012, pp. 5–1.

[10] Gruia Calinescu et al. ≪Maximizing a monotone submodular function subject to a matroid constraint≫. In: *SIAM Journal on Computing* 40.6 (2011), pp. 1740–1766.

[11] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. ≪Dependent randomized rounding for matroid polytopes and applications≫. In: *arXiv preprint arXiv:0909.4348* (2009).

[12] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. ≪Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes≫. In: *CoRR* abs/1105.4593 (2011). URL: http://arxiv.org/abs/1105.4593.

[13] Vasek Chvatal. ≪A greedy heuristic for the set-covering problem≫. In: *Mathematics of operations research* 4.3 (1979), pp. 233–235.

[14] Irit Dinur and David Steurer. ≪Analytical approach to parallel repetition≫. In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. ACM. 2014, pp. 624–633.

[15] Jack Edmonds. ≪Matroids and the greedy algorithm≫. In: *Mathematical programming* 1.1 (1971), pp. 127–136.

[16] Jack Edmonds. ≪Optimum branchings≫. In: *Journal of Research of the national Bureau of Standards B* 71.4 (1967), pp. 233–240.

[17] Jack Edmonds. ≪Submodular functions, matroids, and certain polyhedra≫. In: *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen* 11 (1970).

[18] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. ≪Facility location in evolving metrics≫. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2014, pp. 459–470.

[19] Leah Epstein and Asaf Levin. ≪Robust algorithms for preemptive scheduling≫. In: *Proceedings of the 19th European conference on Algorithms*. Springer-Verlag. 2011, pp. 567–578.

[20] Martin Grötschel, László Lovász, and Alexander Schrijver. ≪The ellipsoid method and its consequences in combinatorial optimization≫. In: *Combinatorica* 1.2 (1981), pp. 169–197.

[21] Albert Gu, Anupam Gupta, and Amit Kumar. ≪The power of deferral: maintaining a constant-competitive Steiner tree online≫. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM. 2013, pp. 525–534.

[22] Anupam Gupta, Kunal Talwar, and Udi Wieder. ≪Changing bases: Multistage optimization for matroids and matchings≫. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2014, pp. 563–575.

[23] Venkatesan Guruswami and Sanjeev Khanna. ≪On the hardness of 4-coloring a 3-colorable graph≫. In: *SIAM Journal on Discrete Mathematics* 18.1 (2004), pp. 30–40.

[24] Makoto Imase and Bernard M Waxman. ≪Dynamic Steiner tree problem≫. In: *SIAM Journal on Discrete Mathematics* 4.3 (1991), pp. 369–384.

[25] Kamal Jain. ≪A factor 2 approximation algorithm for the generalized Steiner network problem≫. In: *Combinatorica* 21.1 (2001), pp. 39–60.

[26] Jeff Kahn and Joseph PS Kung. ≪Varieties of combinatorial geometries≫. In: *Transactions of the American Mathematical Society* 271.2 (1982), pp. 485–499.

[27] Viggo Kann. «Maximum bounded 3-dimensional matching is MAX SNP-complete». In: *Information Processing Letters* 37.1 (1991), pp. 27–35.

[28] Howard Karloff. *Linear programming.* Springer Science & Business Media, 2008.

[29] Narendra Karmarkar. «A new polynomial-time algorithm for linear programming». In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing.* ACM. 1984, pp. 302–311.

[30] Leonid G Khachiyan. «Polynomial algorithms in linear programming». In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72.

[31] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization.* Vol. 46. Cambridge University Press, 2011.

[32] Nicole Megow et al. «The power of recourse for online MST and TSP». In: *Automata, Languages, and Programming* (2012), pp. 689–700.

[33] Adam Meyerson. «The parking permit problem». In: *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on.* IEEE. 2005, pp. 274–282.

[34] Chandrashekhar Nagarajan and David P Williamson. «Offline and online facility leasing». In: *International Conference on Integer Programming and Combinatorial Optimization.* Springer. 2008, pp. 303–315.

[35] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. «An analysis of approximations for maximizing submodular set functions—I». In: *Mathematical Programming* 14.1 (1978), pp. 265–294.

[36] James G Oxley. *Matroid theory.* Vol. 3. Oxford University Press, USA, 2006.

[37] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency.* Vol. 24. Springer Science & Business Media, 2002.

[38] Paul D Seymour. «Decomposition of regular matroids». In: *Journal of combinatorial theory, Series B* 28.3 (1980), pp. 305–359.

[39] Hadas Shachnai, Gal Tamir, and Tami Tamir. «A theory and algorithms for combinatorial reoptimization». In: *Latin American Symposium on Theoretical Informatics.* Springer. 2012, pp. 618–630.

[40] Maxim Sviridenko, Jan Vondrák, and Justin Ward. «Optimal approximation for submodular and supermodular optimization with bounded curvature». In: *Mathematics of Operations Research* (2017).

[41] William Thomas Tutte. «Lectures on matroids». In: *J. Res. Nat. Bur. Standards Sect. B* 69.1-47 (1965), p. 468.

[42] Vijay V Vazirani. *Approximation algorithms.* Springer Science & Business Media, 2013.

[43] Hassler Whitney. ≪On the abstract properties of linear dependence≫. In: *American Journal of Mathematics* 57.3 (1935), pp. 509–533.

[44] Geoff Whittle. ≪A characterization of the matroids representable over GF (3) and the rationals≫. In: *Journal of Combinatorial Theory, Series B* 65.2 (1995), pp. 222–261.

[45] David P Williamson and David B Shmoys. *The design of approximation algorithms.* Cambridge university press, 2011.

[46] Laurence A Wolsey. ≪An analysis of the greedy algorithm for the submodular set covering problem≫. In: *Combinatorica* 2.4 (1982), pp. 385–393.

Κεφάλαιο 1

Εισαγωγή

Στη συνδυαστική βελτιστοποίηση έχουμε προβλήματα που μοντελοποιούν χρονικά στατικές εφαρμογές. Ωστόσο, στην πράξη χρειάζεται να λυθούν στιγμιότυπα του συνδυαστικού προβλήματος βελτιστοποίησης ενώ αυτό αλλάζει με την πάροδο του χρόνου. Φυσικά, μπορεί κανείς να λύσει το πρόβλημα ανεξάρτητα σε κάθε βήμα του χρόνου. Παρ᾽ όλα αυτά, η αλλαγή της λύσης σε διαδοχικά βήματα χρόνου συχνά κοστίζει ένα **κόστος μετάβασης**. Σκεφτείτε, για παράδειγμα, το πρόβλημα που αντιμετωπίζει ένας πωλητής που χρειάζεται να προμηθευτεί ένα προϊόν από k διαφορετικούς παραγωγούς για να ικανοποιήσει τη ζήτηση. Σε μια δεδομένη ημέρα, θα μπορούσε να πάρει τιμές από κάθε έναν από τους παραγωγούς και να επιλέξει τους k φθηνότερους. Καθώς οι τιμές αλλάζουν, αυτό το σύνολο των k φθηνότερων παραγωγών ενδέχεται να αλλάξει. Ωστόσο, υπάρχει ενα σταθερό κόστος για την έναρξη ή/και τη λήξη μιας σχέσης με κάθε νέο παραγωγό. Ο στόχος του πωλητή είναι να ελαχιστοποιήσει το άθροισμα των δύο αυτών κοστών: το *κόστος απόκτησης* $a(e)$ που πληρώνεται κάθε φορά που ξεκινά μια σχέση με τον παραγωγό e και ένα *κόστος κράτησης* ανά περίοδο $c_t(e)$ που πληρώνεται για να γίνει η αγορά τη στιγμή t από τον παραγωγό e, αθροιζόμενα στις T περιόδους. Παρατηρήστε ότι αυτό το πρόβλημα είναι ένα παράδειγμα διατήρησης μιας βάσης ενός k-ομοιόμορφου Matroid. Η εύρεση της βέλτιστης λύσης είναι επίσης τετριμμένη για matroids, δεδομένου ότι ο άπληστος αλγόριθμος είναι ο βέλτιστος στην περίπτωση αυτή. Έτσι, είναι φυσικό να αναρωτηθούμε αν είναι επίσης εύκολο να λυθεί το πρόβλημα στην χρονομεταβαλλόμενή του γενίκευση για γενικά matroids. Για παράδειγμα, μπορεί να θέλουμε να διατηρήσουμε ένα δένδρο που συνδέει ένα δεδομένο γράφημα και σε κάθε βήμα, η ακμή e κοστίζει $c_t(e)$ και ένα κόστος απόκτησης $a(e)$ πρέπει να πληρώνεται κάθε φορά που μια νέα ακμή e εισέρχεται στο δέντρο.

Το πρόβλημα της Χρονομεταβαλλόμενης Βελτιστοποίησης σε Μητροειδή (MMM) ορίστηκε από τους Gupta, Talwar και Wieder [13]. Οι συγγραφείς απέδειξαν ότι όταν το κόστος απόκτησης είναι μη ομοιόμορφο, που σημαίνει ότι αν αλλάξουμε διαφορετικά στοιχεία, ενδέχεται να πληρώσουμε διαφορετικό κόστος απόκτησης, τότε η λογαριθμική προσέγγιση είναι βέλτιστη, εκτός εάν $P = NP$. Ωστόσο, η αναγωγή, που παρουσιάζουν, βασίζεται σε μεγάλο βαθμό στην ανομοιομορφία των κοστών απόκτησης. Για το λόγο αυτό, επικεντρωνόμαστε στην περίπτωση ενιαίου κόστους απόκτησης και κάνουμε ένα πρώτο βήμα προς έναν αλγόριθμο σταθερής προσέγγισης, παρουσιάζοντας έναν αλγόριθμο που έχει σταθερή προσέγγιση στο κόστος κράτησης και λογαριθμική προσέγγιση στο κόστος απόκτησης, μια εγγύηση που οι προηγούμενοι αλγόριθμοι δεν είχαν. Επιπλέον, στο [13], οι Gupta et al. αποδεικνύουν ότι το πρόβλημα στην περίπτωση που έχουμε matroid διαμέρισης βρίσκεται στο P, ακόμη και αν το κόστος απόκτησης δεν είναι ομοιόμορφο και εξαρτάται από το χρόνο. Παρουσιάζουν επίσης το ίδιο αποτέλεσμα για την περίπτωση $T = 2$, ακόμη και αν τα δύο matroids είναι διαφορετικά. Αποδεικνύουμε ότι τα φυσικά LP για αυτά τα προβλήματα είναι ακέραια, παρουσιάζοντας τις πρώτες αποδείξεις integrality για LP για προβλήματα χρονομεταβαλλόμενης βελτιστοποίησης.

Μαζί με το έργο των Gupta et al. [13], η εργασία σχετίζεται με διάφορες γραμμές έρευνας. Στην online περίπτωση, το πρόβλημα MMM είναι επίσης μια ειδική περίπτωση του κλασσικού προβλήματος Metrical Task Systems [6], [1], [5]. Προσπαθώντας να ενοποιήσουν την θεωρία μάθησης και την ανταγωνιστική ανάλυση, οι Buchbinder et al. [7] μελέτησαν ένα πρόβλημα πολύ παρόμοιο με το δικό μας. Στη δυναμική συντήρηση δέντρων Steiner [12], [15], [14], ο στόχος είναι να διατηρηθεί ένα σχεδόν βέλτιστο Steiner δέντρο σε κάθε χρονική στιγμή (όπου προστίθενται τερματικά) ενώ αλλάζουν μερικές ακμές σε κάθε χρονικό βήμα. Στη δυναμική εξισορρόπηση φορτίου [11], [3] πρέπει να διατηρηθεί μια καλή λύση ενώ μετακινείτε ένας μικρός αριθμός εργασιών.

Στην offline περίπτωση, οι Shachnai et al. [18] μελετούν προβλήματα ¨επαναβελτιστοποίησης¨: δεδομένης μιας αρχικής λύσης και μιας νέας στιγμής, θέλουν να εξισορροπήσουν το κόστος μετάβασης και το κόστος στη νέα περίπτωση. Επίσης έχει γίνει αρκετή έρευνα σχετικά με τα προβλήματα δανεισμού [4], [17], [16]: αυτά είναι προβλήματα βελτιστοποίησης όπου τα στοιχεία μπορούν να λαμβάνονται για ένα διάστημα οποιουδήποτε μήκους, όπου το κόστος είναι κοίλο στα μήκη.

1.1 Προσεγγιστικοί αλγόριθμοι

Η κλάση πολυπλοκότητας **P** περιέχει το σύνολο των προβλημάτων που μπορούν να λυθούν σε πολυωνυμικό χρόνο. Από θεωρητική άποψη, αυτό περιγράφει την κλάση των ¨εύκολων προβλημάτων¨, δηλαδή τα προβλήματα που μπορούν να λυθούν αποδοτικά. Η κλάση **NP** είναι το σύνολο των προβλημάτων που μπορούν να λυθούν με μη ντετερμινιστικό τρόπο σε πολυωνυμικό χρόνο ή ισοδύναμα, τα προβλήματα για τα οποία μια λύση μπορεί να επαληθευτεί οτι είναι σωστή σε πολυωνυμικό χρόνο. Το NP περιέχει πολλά ενδιαφέροντα προβλήματα που συχνά προκύπτουν στην πράξη, αλλά υπάρχει καλός λόγος να πιστεύουμε οτι $\mathbf{P} \neq \mathbf{NP}$. Δηλαδή, δε φαίνεται να είναι πιθανή η ύπαρξη αλγορίθμων για την αποδοτική επίλυση NP προβλημάτων βελτιστοποίησης. Γι΄ αυτό συχνά καταφεύγουμε σε ευρυστικές μεθόδους για την επίλυση αυτών των προβλημάτων. Κάποιες ευρυστικές μέθοδοι καταφέρνουν να βρούν μια βέλτιστη λύση, αν και μπορεί να πάρει εκθετικό χρόνο για να συμβεί αυτό. Άλλες τρέχουν πάντα σε πολυωνυμικό χρόνο, αν και μπορεί να μην επιστρέψουν μια βέλτιστη λύση. Οι Προσεγγιστικοί Αλγόριθμοι εμπίπτουν στην τελευταία κατηγορία. Ωστόσο, αν και δεν βρίσκουν μια βέλτιστη λύση, εγγυώνται την ποιότητα της λύσης που επιστρέψανε.

Ορισμός 1.1. Ένας α-προσεγγιστικός αλγόριθμος για ένα πρόβλημα βελτιστοποίησης είναι ένας πολυωνυμικός αλγόριθμος που για όλα τα στιγμιότυπα του προβλήματος παράγει μια λύση της οποίας η αντικειμενική τιμή απέχει το πολύ κατά έναν παράγοντα α απο την αντικειμενική τιμή μιάς βέλτιστης λύσης.

1.2 Βασικές Τεχνικές Σχεδίασης Προσεγγιστικών Αλγορίθμων

Σε αυτό το κεφάλαιο δίνουμε μια πολύ σύντομη επισκόπηση των βασικών τεχνικών για το σχεδιασμό προσεγγιστικών αλγορίθμων, βασισμένες στον Γραμμικό Προγραμματισμό. Η παρουσίασή μας βασίζεται στα βιβλία [20] και [19], όπου μπορεί κανείς να βρει μια λεπτομερή παρουσίαση του θέματος.

Για την παρουσίαση αυτή, θα βασιστούμε σε ένα παράδειγμα δύσκολου προβλήματος συνδυαστικής βελτιστοποίησης, το Set Cover.

Ορισμός 1.2. Στο πρόβλημα Set Cover, δίνεται ένα σύνολο στοιχείων $\mathcal{E} = \{e_1, ..., e_n\}$, μερικά υποσύνολα αυτών των στοιχείων $S_1, S_2, ..., S_m$ όπου κάθε $S_j \subseteq \mathcal{E}$ και ένα μη αρνητικό βάρος $w_j \geq 0$, για κάθε υποσύνολο S_j. Ο στόχος είναι να βρεθεί μια συλλογή των υποσυνόλων, ελάχιστου βάρους, που να καλύπτει το σύνολο $\mathcal{E}$. Δηλαδή, θέλουμε να βρούμε ένα $I \subseteq [m]$ που ελαχιστοποιεί το $\sum_{j\in I} w_j$ με $\cup_{j\in I} S_j = \mathcal{E}$.

Ενα γραμμικό πρόγραμμα για το Set Cover είναι το εξής:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j: e_i \in S_j} x_j \geq 1, \quad i = 1, ..., n \\ & x_j \geq 0, \quad j = 1, ..., m \end{aligned} \tag{1.1}$$

Το Set Cover είναι NP-hard και γι' αυτό σχεδιάζουμε προσεγγιστικούς αλγορίθμους για την επίλυσή του. Οι βασικές τεχνικές σχεδίασης, που χρησιμοποιούμε σε αυτήν την εργασία, εφαρμοζόμενες στο Set Cover είναι οι εξής:

1. **Deterministic Rounding**: Έχοντας λύσει το LP, επιλέγουμε όλα τα σύνολα j για το οποία ισχύει $x_j \geq 1/f$, όπου $f = \max_{e\in\mathcal{E}} |j \in [m] : e \in S_j|$. Αποδεικνύεται οτι ο αλγόριθμος είναι f-προσεγγιστικός.

2. **Randomized Rounding**: Έχοντας λύσει το LP, επιλέγουμε ένα σύνολο j με πιθανότητα x_j. Επαναλαμβάνουμε το πείραμα $2 \ln n$ φορές και επιλέγουμε ένα σύνολο αν αυτό επιλέχθηκε σε κάποιο απ' τα πειράματα. Ο αλγόριθμος επιστρέφει εφικτή λύση με μεγάλη πιθανότητα και αν αυτό συμβεί, είναι $O(\log n)$-προσεγγιστικός.

3. **Greedy Algorithm**: Ο αλγόριθμος διατηρεί μια συλλογή $I \subseteq \{S_1, ..., S_m\}$ και σε κάθε επανάληψη προσθέτει στο I το σύνολο με το ελάχιστο $\frac{w_j}{|S_j \setminus I|}$. Ενας τρόπος να αποδειχθεί οτι ο αλγόριθμος είναι $O(\log n)$-προσεγγιστικός είναι η μέθοδος Dual-Fitting.

Κεφάλαιο 2

Μητροειδή

Τα Μητροειδή (Matroids) μελετήθηκαν πρώτη φορά από τον Whitney το 1935 και έχουν στόχο να συλλάβουν αφηρημένα την έννοια της εξάρτησης. Ο ορισμός του Whitney γενικεύει μια εκπληκτική ποικιλία συνδυαστικών δομών, όπως τα συνδετικά δέντρα.

Ορισμός 2.3. Ενα ζευγάρι $\mathcal{M} = (\mathcal{S}, \mathcal{I})$ είναι Matroid αν $\mathcal{I}$ είναι μια μή κενή συλλογή υποσυνόλων του $\mathcal{S}$ που ικανοποιεί τις ακόλουθες ιδιότητες:

1. $\emptyset \in \mathcal{I}$
2. $A \in \mathcal{I}$ και $B \subseteq A \Rightarrow B \in \mathcal{I}$
3. $A, B \in \mathcal{I}$ και $|A| > |B| \Rightarrow \exists x \in B \setminus A$ τέτοιο ώστε $A \cup \{x\} \in \mathcal{I}$

Ενα σύνολο $A \subseteq \mathcal{S}$ θα λέγεται ανεξάρτητο αν $A \in \mathcal{I}$, αλλιώς θα λέγεται εξαρτημένο. Ένα μεγιστικό σύνολο $A \in \mathcal{I}$ θα λέγεται βάση του $\mathcal{M}$. Επίσης, για κάθε υποσύνολο A του $\mathcal{S}$ ορίζουμε $r(A) = \max_{B \subseteq A:\ B \in \mathcal{I}} |B|$.

Βασικά αποτελέσματα

Αν τα στοιχεία του $\mathcal{S}$ έχουν βάρη τότε υπάρχει άπληστος αλγόριθμος που υπολογίζει μια βάση μέγιστου βάρους. Επίσης, υπάρχει πολυωνυμικός αλγόριθμος για τον υπολογισμό ενός μεγίστου βάρους κοινού ανεξάρτητου συνόλου δύο Matroids (το λεγόμενο πρόβλημα two Matroid Intersection). Επιπλέον, τα δύο κλασικά γραμμικά προγράμματα για τα δύο αυτά προβλήματα επιστρέφουν πάντα ακέραια λύση. Τέλος, το πρόβλημα k Matroid Intersection: εύρεση κοινού ανεξάρτητου συνόλου k Μητροειδών, με μέγιστο πληθάριθμο είναι NP-hard, για $k \geq 3$. Ωστόσο, χρησιμοποιώντας μια μέθοδο που ονομάζεται "iterative rounding", μπορεί να σχεδιαστεί αλγόριθμος προσέγγισης $(k-1)$. Το έργο του Jack Edmonds [9], [8] έδειξε τα προαναφερθέντα αποτελέσματα.

Κεφάλαιο 3

Χρονομεταβαλλόμενη Βελτιστοποίηση σε Μητροειδή

Μπορούμε τώρα να ορίσουμε το πρόβλημα της Χρονομεταβαλλόμενης Βελτιστοποίησης σε Μητροειδή (MMM). Τα αποτελέσματα που μελετάμε στο κεφάλαιο αυτό παρουσιάστηκαν στη δημοσίευση των Gupta, Talwar, Wieder "Changing bases: Multistage optimization for matroids and matchings" [13].

Ορισμός 3.4. Ένα στιγμιότυπο του προβλήματος MMM αποτελείται από ένα matroid $\mathscr{M} = (E, \mathcal{I})$, με $r(E) = r$, ένα κόστος $a(e) \geq 0$ για κάθε $e \in E$ και για κάθε βήμα $t \in [T]$, το κόστος κράτησης $c_t(e) \geq 0$. Ο στόχος είναι να βρεθούν βάσεις $\{B_t \in \mathcal{I}\}_{t \in [T]}$ για να ελαχιστοποιηθεί το

$$\sum_t (c_t(B_t) + a(B_t \setminus B_{t-1}))$$

όπου ορίζουμε $B_0 := \emptyset$.

Κύρια Αποτελέσματα

Αρχικά, στο [13] οι συγγραφείς παρουσιάζουν κάποια θετικά και αρνητικά αποτελέσματα στο πρόβλημα MMM. Πρώτον, αποδεικνύουν πώς η επέκταση του άπληστου αλγορίθμου για το πρόβλημα της βάσης μεγίστου βάρους, σε αυτή τη χρονικά εξελισσόμενη γενίκευση, που αναλύεται μέσω Dual-Fitting, δίνει έναν αλγόριθμο προσέγγισης $\log T$. Στη συνέχεια, παρουσιάζουν έναν randomized rounding αλγόριθμο, ο οποίος κάνει randomized rounding σε κάθε χρονική στιγμή, αλλά η τυχαιότητα μοιράζεται μεταξύ όλων των χρονικών στιγμών. Ο αλγόριθμος πετυχαίνει προσέγγιση $O(\log rT)$ και μπορεί να τροποποιηθεί για να δώσει προσέγγιση $O(\log r \frac{a_{max}}{a_{min}})$ (a_{max}, a_{min}: το μέγιστο και το ελάχιστο κόστος απόκτησης αντίστοιχα). Επιπρόσθετα, δείχνουν μια ακριβή αναγωγή από το Set Cover, η οποία δείχνει ότι η λογαριθμική προσέγγιση είναι βέλτιστη. Συνεχίζοντας με τα αρνητικά αποτελέσματα, δείχνουν ότι είναι NP-hard να προσεγγίσουμε καλύτερα από $\Omega(T)$ το MMM με διαφορετικά matroids αν $T \geq 3$. Τέλος, αποδεικνύουν οτι στο πρόβλημα της Χρονομεταβαλλόμενης Βελτιστοποίησης, στην περίπτωση των Perfect Matchings, η δυσκολία αυξάνεται κατά πολύ: για κάθε σταθερό $\epsilon > 0$, δεν υπάρχει δυνατότητα προσέγγισης $O(n^{1-\epsilon})$.

Κεφάλαιο 4

MMM σε ειδικές περιπτώσεις

4.1 Ομοοιόμορφα Κόστη Απόκτησης

Όπως είπαμε, στη γενική περίπτωση η λογαριθμική προσέγγιση είναι βέλτιστη. Γι΄ αυτό το λόγο, στην ενότητα αυτή, μελετάμε το MMM, στην περίπτωση που τα κόστος απόκτησης είναι το ίδιο για όλα τα στοιχεία. Για απλότητα της παρουσίασης, θα ασχοληθούμε με το πρόβλημα της Χρονομεταβαλλόμενης Βελτιστοποίησης σε Συνδετικά Δέντρα (MSTM), το οποίο παρ΄όλο που είναι ειδική περίπτωση του MMM, περιλαμβάνει όλες τις δυσκολίες του και όσες αποδείξεις και αλγορίθμους παρουσιάζουμε μπορούν να γενικευτούν για το MMM. Το συγκεκριμένο πρόβλημα, για ομοιόμορφα κόστη, είναι APX-hard [Fotakis, Lampis, Paschos, Plevrakis '17]. Σε ό,τι αφορά το αλγοριθμικό κομμάτι, ο Randomized Rounding αλγόριθμος πετυχαίνει $O(\log n)$ προσέγγιση στο κόστος απόκτησης και $O(\log n)$ προσέγγιση στο κόστος κράτησης. Επίσης αξίζει να σημειωθεί οτι όλοι οι αλγόριθμοι που έχουν παρουσιαστεί μέχρι στιγμής για προβλήματα Χρονομεταβαλλόμενης Βελτιστοποίησης και βασίζονται σε Γραμμικά Προγράμματα κάνουν Randomized Rounding [10],[13],[2]. Σε αυτή τη διπλωματική παρουσιάζουμε τον πρώτο αλγόριθμο που κάνει Deterministic Rounding για ένα πρόβλημα Χρονομεταβαλλόμενης Βελτιστοποίησης, εν προκειμένω το MSTM και πετυχαίνει $O(\log n)$ προσέγγιση στο κόστος απόκτησης και 3 προσέγγιση στο κόστος κράτησης. Αρχικά, ξεκινάμε με τον ορισμό του προβλήματος:

Ορισμός 4.5. Ένα στιγμιότυπο του προβλήματος MSTM αποτελείται από έναν γράφο $G = (V, E)$, ένα κόστος απόκτησης $g \geq 0$ και για κάθε βήμα $t \in [T]$, το κόστος κράτησης $c_t(e) \geq 0$. Ο στόχος είναι να βρεθούν συνδετικά δέντρα $\{\mathcal{T}_t\}_{t\in[T]}$ για να ελαχιστοποιηθεί το

$$\sum_t (c_t(\mathcal{T}_t) + g|\mathcal{T}_t \setminus \mathcal{T}_{t-1}|)$$

Πρέπει να σημειωθεί οτι μπορούμε να θεωρήσουμε, χωρίς βλάβη της γενικότητας οτι τη χρονική στιγμή $t = 1$ δεν πληρόνουμε τίποτα για την απόκτηση των ακμών του $\mathcal{T}_1$, αφου μπορούμε να ενσωματώσουμε αυτά τα κόστη στο c_1: $c_1(e) \leftarrow c_1(e) + g$, $\forall e \in E$. Συνεχίζουμε με την αναφορά κάποιων βασικών γνώσεων πάνω στο πολύτοπο των δασών και στο πολύτοπο των συνδετικών δέντρων ενός γράφου.

Κλασματικά Δάση και Κλασματικά Συνδετικά Δέντρα

Το Γραμμικό Πρόγραμμα για το πρόβλημα του συνδετικού δέντρου ελαχίστου κόστους είναι το ακόλουθο:

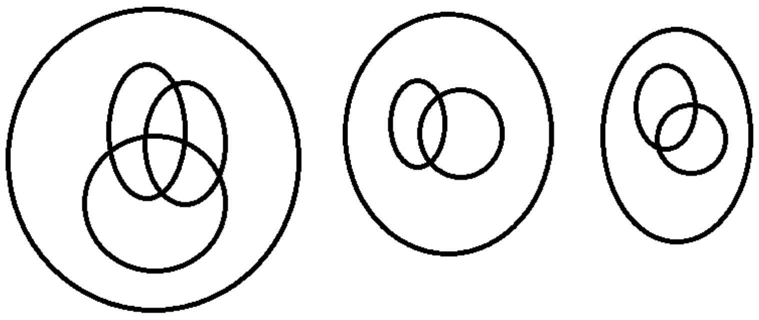
$$\begin{array}{ll} \text{minimize} & \sum_{e\in E} c(e)x(e) \\ \text{subject to} & x(E) = n-1 \\ & x(E(S)) \leq |S|-1, \quad \forall S \subseteq V \\ & x(e) \geq 0,\ \forall e \in E \end{array}$$

Ο Edmonds έδειξε στο [8] οτι αυτό το LP ειναι ακέραιο.
Έστω $G = (V, E)$ και $x \in \mathbb{R}_+^{|E|}$.

- Θα λέμε οτι το x είναι **κλασματικό συνδετικό δέντρο** αν και μόνο αν $x(E(S)) \leq |S| - 1$, $\forall S \subseteq V$ και $x(E) = n - 1$.
- Θα λέμε οτι το x είναι **κλασματικό δάσος** αν και μόνο αν $x(E(S)) \leq |S| - 1$, $\forall S \subseteq V$.
- Θα λέμε οτι το x έχει έναν **κλασματικό κύκλο** αν και μόνο αν το x δεν είναι κλασματικό δάσος.

Έστω οτι το x είναι ένα κλασματικό δέντρο του G.

- Θα λέμε οτι το $S \subseteq V$ είναι **σύνολο ισότητας** στο x αν και μόνο αν $x(E(S)) = |S| - 1$. Σε αυτήν την περίπτωση θα λέμε οτι το x είναι **κλασματικά συνδεδεμένο στο** S.
- Έστω $\mathcal{F}$ το σύνολο των συνόλων ισότητας του x. Στο [8] αποδεικνύεται οτι τα μεγιστικά, ως προς την διάταξη που ορίζει το $\subseteq$, σύνολα του $\mathcal{F}$ είναι ξένα μεταξύ τους, όπως φαίνεται στο παρακάτω παράδειγμα.



Ονομάζουμε αυτά τα σύνολα **κλασματικά συνεκτικές συνιστώσες του** x.

Το Γραμμικό Πρόγραμμα

Το ακόλουθο LP επεκτείνει το γραμμικό πρόγραμμα για το συνδετικό δέντρο ελαχίστου κόστους, στη χρονομεταβαλλόμενη γενίκευση (MSTM) που μελετάμε.

$$\begin{array}{ll} \text{minimize} & \sum_t \sum_e c_t(e)x_t(e) + g\sum_t \sum_e y_t(e) \\ \text{subject to} & x_t(E) = n-1, \quad \forall t \\ & x_t(E(S)) \leq |S|-1, \quad \forall S \subseteq V,\ \forall t \\ & x_t(e) \geq 0,\ \forall e \in E,\ \forall t \\ & y_t(e) \geq x_t(e) - x_{t-1}(e),\ \forall e \in E,\ \forall t \\ & y_t(e) \geq 0,\ \forall e \in E,\ \forall t \end{array}$$

Παρατηρείστε οτι αν $\{x_t, y_t\}_t$ η βέλτιστη λύση του LP, χωρίς βλάβη της γενικότητας έχουμε: $y_t(e) = max(x_t(e) - x_{t-1}(e), 0)\ \forall t, e$. Ο αλγόριθμος αποτελείται απο 4 βήματα.

Χωρισμός σε εποχές

Αρχικά, λύνουμε το LP το οποίο μας επιστρέφει T κλασματικά συνδετικά δέντρα $x_1, x_2, ..., x_T$. Θέτουμε $z^1_t(e) = min_{1\leq t'\leq t}x_{t'}(e)$. Έστω t_1 το μέγιστο t τέτοιο ώστε $z^1_t(E) = \sum_e z^1_t(e) \geq \frac{n-1}{2}$. Ορίζουμε σαν πρώτη εποχή, το διάστημα $[1, t_1]$. Παρατηρείστε οτι μέσα σε αυτό το διάστημα, η λύση $x_1, ..., x_{t_1}$ πληρώνει κόστος απόκτησης τουλάχιστον $g\frac{n-1}{2}$. Στη συνέχεια, θέτουμε $z^2_t(e) = min_{t_1+1\leq t'\leq t}x_{t'}(e)$. Έστω t_2 το μέγιστο t τέτοιο ώστε $z^2_t(E) = \sum_e z^2_t(e) \geq \frac{n-1}{2}$. Ορίζουμε σαν δεύτερη εποχή, το διάστημα $[t_1+1, t_2]$. Μέσα σε αυτό το διάστημα, η λύση $x_{t_1+1}, ..., x_{t_2}$ πληρώνει κόστος απόκτησης τουλάχιστον $g\frac{n-1}{2}$. Συνεχίζουμε με τον ίδιο τρόπο έως ότου χωρίσουμε όλο το $[1, T]$ σε εποχές. Ακολουθεί ένα παράδειγμα της παραπάνω διαδικάσιας (T=13).

$$\underbrace{x_1\ x_2\ x_3\ x_4}_{epoch_1}\ \underbrace{x_5\ x_6\ x_7}_{epoch_2}\ \underbrace{x_8\ x_9\ x_{10}\ x_{11}\ x_{12}}_{epoch_3}\ \underbrace{x_{13}}_{epoch_4}$$

Συνεπακόλουθα, άμα κάνουμε rounding **ανεξάρτητα σε κάθε εποχή**, τότε χάνουμε το πολύ έναν παράγοντα 2 στο κόστος απόκτησης.

Rounding

Μπορούμε να επικεντρωθούμε πλέον μόνο στην πρώτη εποχή $[1, t_1]$ (για τις υπόλοιπες κάνουμε τα ίδια, **ανεξάρτητα**). Ορίζουμε $z(e) = z^1(e)$, $\forall e$. Παρατηρείστε οτι το z είναι κλασματικό δάσος. Επίσης το γεγονός οτι $z(E) \geq \frac{n-1}{2}$ δείχνει οτι τα κλασματικά συνδετικά δέντρα $x_1, ..., x_{t_1}$ δε διαφέρουν πολύ. Προσέξτε οτι αν ήταν ίδια ($z(E) = n-1$) τότε το rounding θα ήταν τετριμμένο, καθώς για κάθε χρονική στιγμή του $[1, t_1]$ θα επιλέγαμε το συνδετικό δέντρο ελαχίστου κόστους, οπου η συνάρτηση κόστους θα ήταν η $c = \sum_{t\in[1,t_1]} c_t$. Επίσης, αν κάποιο $S \subseteq V$ το S είναι σύνολο ισότητας για το z, τότε τα $x_1, ..., x_{t_1}$ θα ήταν ίδια μέσα στο S και κλασματικά συνδεδεμένα μέσα σε αυτό. Άρα, υπολογίζοντας το συνδετικό δέντρο ελαχίστου κόστους **$\mathcal{T}$ μέσα στο S**, με συνάρτηση κόστους $c = \sum_{t\in[1,t_1]} c_t$ και θέτοντας $x_i(e) = \mathbf{1}\{e \in \mathcal{T}\}$, $\forall i \in [1, t_1], e \in S$ δεν αυξάνεται το κόστος. Βέβαια, δεν υπάρχει κάποιος λόγος να έχει το z κάποιο σύνολο ισότητας. Ωστόσο, δεδομένου οτι $z(E) \geq \frac{n-1}{2}$ φαντάζει λογικό να έχει ένα σύνολο που είναι "**σχεδόν σύνολο ισότητας**", με την έννοια οτι αν αυξήσουμε λίγο τις μεταβλητές ακμών που βρίσκονται στο εσωτερικό αυτού του συνόλου, θα το κάνουμε σύνολο ισότητας. Για να πετύχουμε κάτι τέτοιο χρησιμοποιούμε το ακόλουθο γραμμικό πρόγραμμα:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} z_{new}(e) \\ \text{subject to} \quad & z_{new}(E(S)) \leq |S| - 1, \quad \forall S \subseteq V \\ & 0 \leq z_{new}(e) \leq 3z(e), \ \forall e \in E \end{aligned}$$

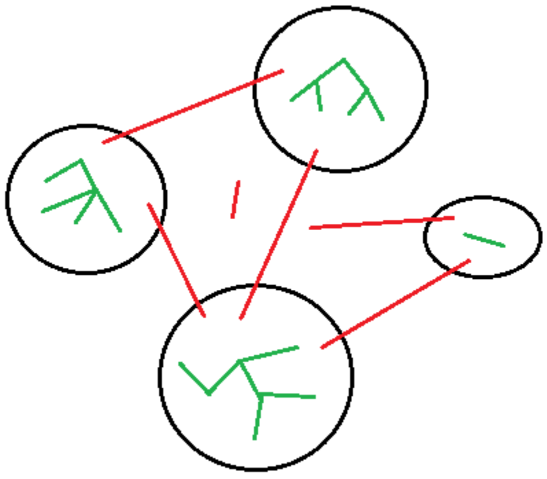
Έστω z_{new} η βέλτιστη λύση του παραπάνω LP. Εκ κατασκευής, το z_{new} είναι κλασματικό δάσος. Έστω $S_1, ..., S_k \subseteq V$ οι κλασματικά συνεκτικές συνιστώσες του. Αποδεικνύεται οτι

$$\sum_{i=1}^{k} (|S_i| - 1) \geq \frac{n-1}{4} \quad \mathbf{(1)}$$

Έστω $\Pi = \cup_{i=1}^{k} E(S_i)$ (πράσινες ακμές). Αλλάζουμε όλα τα $\{x_t(e)\}_{t \in [1,t_1], e \in \Pi}$ ώς εξής: $x_t(e) \leftarrow z_{new}(e)$, $\forall e \in \Pi$. Εκ κατασκευής του z_{new}, με αυτήν την αλλαγή χάνουμε το πολύ έναν παράγοντα 3 στο κόστος κράτησης και το κόστος απόκτησης ενδέχεται να μειώθηκε κι όλας στο εσωτερικό του $[1, t_1]$. Τώρα όμως, μέσα σε κάθε ένα απο τα $S_1, ..., S_k$, τα $x_1, ..., x_{t_1}$ είναι ίδια και κλασματικώς συνδεδεμένα! Άρα βρισκόμαστε στην ειδική περίπτωση που το rounding (μέσα στα S_i) είναι τετριμμένο. Πιο συγκεκριμένα, αν H_i είναι το συνδετικό δέντρο ελαχίστου κόστους μέσα στο S_i, με συνάρτηση κόστους $c = \sum_{t=1}^{t_1} c_t$, τότε θέτοντας για κάθε i:

$$x_t(e) \leftarrow \mathbf{1}\{e \in H_i\}, \ \forall e \in E(S_i), \ \forall t \in [1, t_1]$$

δεν αυξάνεται κανένα κόστος. Το παρακάτω σχήμα δείχνει τη μορφή του ενός απο τα x_i μετά τις αλλαγές που κάναμε. Οι κύκλοι είναι τα S_i και οι πράσινες ακμές είναι τα στοιχεία του Π. Στις πράσινες ακμές, μετά τις αλλαγές που κάναμε, τα x_i είναι ακέραια και ταυτίζονται. Στις κόκκινες ακμές ενδεχομένως να είναι κλασματικά και διαφορετικά μεταξύ τους. Το σύνολο των



κόκκινων ακμών θα το γράφουμε Κ (κόκκινες).

Αφαίρεση κλασματικών κύκλων

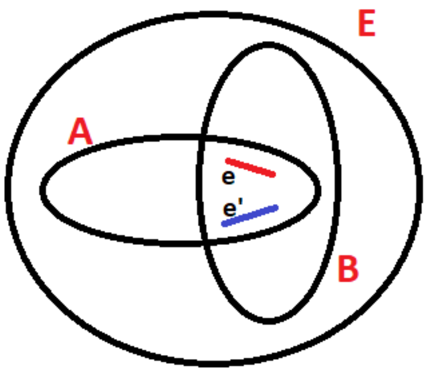
Παρ' όλο που καταφέραμε να κάνουμε round μεγάλο κομμάτι των $x_1, ..., x_{t_1}$, τα τελευταία έχουν πλέον κλασματικούς κύκλους που πρέπει να αφαιρεθούν χωρίς να αυξηθεί το κόστος. Αυτό το πετυχαίνουμε με το ακόλουθο λήμμα:

Λήμμα 4.1. *Μόνο μειώνοντας μεταβλητές $\{x_t(e)\colon e \in K, t \in [1, t_1]\}$, μπορούμε να κάνουμε τα $x_1, ..., x_{t_1}$ κλασματικά συνδετικά δέντρα, χωρίς να αυξήσουμε το κόστος απόκτησης.*

Για την απόδειξη του λήμματος αυτού χρειαζόμαστε μία έννοια ανταλλαξιμότητας ακμών στην κλασματική περίπτωση. Πιό συγκεκριμένα, αν μια ακμή δεν ανήκει σε ένα ακέραιο δέντρο, τότε οι ακμές που ανήκουν στο μονοπάτι του δέντρου που ενώνει τα άκρα της είναι ανταλλάξιμες με αυτή. Χρειαζόμαστε μια τέτοια έννοια στην περίπτωση των κλασματικών συνδετικών δέντρων, την οποία την ονομάζουμε ¨*κλασματική ανταλλαξιμότητα*¨. Άπαξ και εισάγουμε αυτήν την έννοια, η απόδειξη του λήμματος 4.1 είναι ευθύγραμμη. Ας επικέντρωθούμε λοιπόν σε ένα κλασματικά συνδετικό δέντρο, το x_1. Έστω μια ακμή e της οποίας θέλουμε να αυξήσουμε τη μεταβλητή, $x_1(e) < 1$. Ο λόγος που δε μπορούμε να κάνουμε κάτι τέτοιο χωρίς να παραβιάσουμε τους περιορισμούς που καθιστούν το x_1 κλασματικά συνδετικό δέντρο είναι η ύπαρξη περιορισμών που αντιστοιχούν σε σύνολα ισότητας. Έστω $\mathcal{F} = \{S \subseteq V : \ x_1(E(S)) = |S| - 1\}$. Στο [8] αποδεικνύεται οτι

$$A, B \in \mathcal{F} \text{ και } A \cap B \neq \emptyset \Rightarrow A \cap B \in \mathcal{F}$$

Αυτό σημαίνει οτι υπάρχει ένα σύνολο ακμών που ονομάζουμε $mts(e)$, το οποίο περιέχει τη e και για κάθε σύνολο ακμών S που περιέχει την e ισχύει οτι $mts(e) \subseteq S$. Ο λόγος που αυτό ισχύει συνοψίζεται στο ακόλουθο σχήμα οπου οι κύκλοι αποτελούν σύνολα ισότητας για το x_1 που περιέχουν την e.



εφόσον $x_1(mts(e)) = |mts(e)| - 1$ και $x_1(e) < 1$, υπάρχει $e' \in mts(e)$ με $x_1(e') > 0$. Εξ ορισμού, αν $A \in \mathcal{F}$ και $e \in A$, τότε $e' \in A$. Για το λόγο αυτό μπορούμε να αρχίσουμε να αυξάνουμε το $x_1(e)$ και να μειώνουμε το $x_1(e')$ με τον ίδιο ρυθμό, έως ότου προστεθεί νέο σύνολο στο $\mathcal{F}$.

Σύνθλιψη και Αναδρομή

Πλέον τα $x_1, ..., x_T$ είναι κλασματικά συνδετικά δέντρα. Συνθλίβουμε όλες τις πράσινες ακμές που ανήκουν στα x_t, $t \in [1, t_1]$ και έτσι προκύπτουν τα $x'_1, ..., x'_{t_1}$ που είναι κλασματικά συνδετικά δέντρα του $G/\cup_i S_i$. Τώρα, επαναλαμβάνουμε όλη τη διαδικασία (χωρισμός σε εποχές κλπ) πάνω στα $x'_1, ..., x'_{t_1}$.

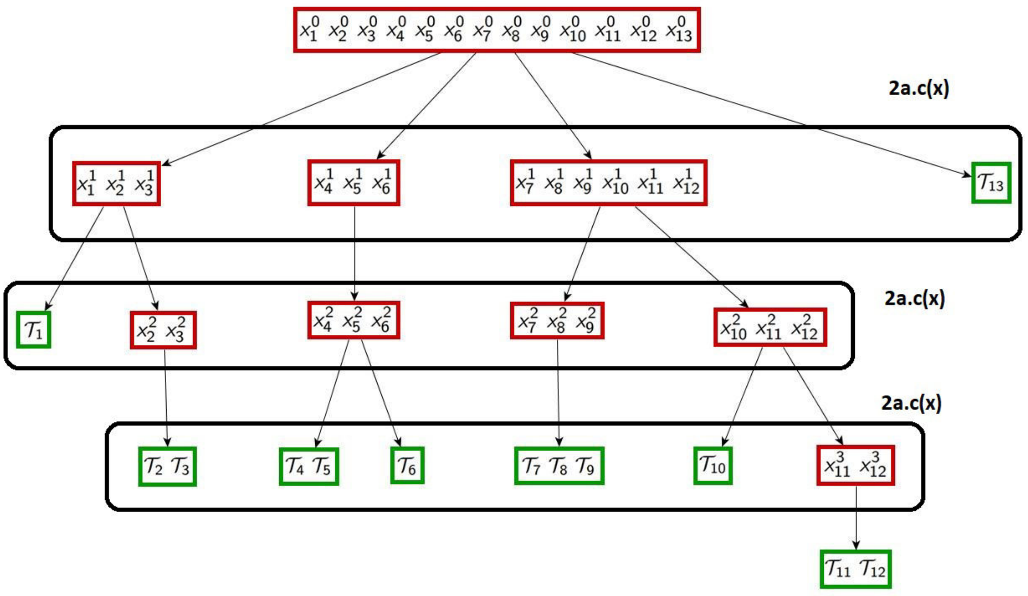
Λόγος Προσέγγισης

Εκ κατασκευής:

$$\sum_{t\in[1,t_1]} \sum_{i=1}^{k} c_t(H_i) \leq 3 \sum_{t\in[1,t_1]} \sum_{e\in\Pi} c_t(e)x_t(e)$$

Ταυτόχρονα, όλες οι μεταβλητές που αντιστοιχούν σε κόκκινες ακμές μειώθηκαν ή έμειναν ίδιες. Συνεπώς, επαγωγικά, έχουμε προσέγγιση 3 στο κόστος κράτησης της $x_1, ..., x_T$.

Το ακόλουθο δέντρο δείχνει ένα παράδειγμα εκτέλεσης του αλγορίθμου.


$x_1^0\ x_2^0\ x_3^0\ x_4^0\ x_5^0\ x_6^0\ x_7^0\ x_8^0\ x_9^0\ x_{10}^0\ x_{11}^0\ x_{12}^0\ x_{13}^0$
2a.c(x)
$x_1^1\ x_2^1\ x_3^1$
$x_4^1\ x_5^1\ x_6^1$
$x_7^1\ x_8^1\ x_9^1\ x_{10}^1\ x_{11}^1\ x_{12}^1$
$\mathcal{T}_{13}$
2a.c(x)
$\mathcal{T}_1$
$x_2^2\ x_3^2$
$x_4^2\ x_5^2\ x_6^2$
$x_7^2\ x_8^2\ x_9^2$
$x_{10}^2\ x_{11}^2\ x_{12}^2$
2a.c(x)
$\mathcal{T}_2\ \mathcal{T}_3$
$\mathcal{T}_4\ \mathcal{T}_5$
$\mathcal{T}_6$
$\mathcal{T}_7\ \mathcal{T}_8\ \mathcal{T}_9$
$\mathcal{T}_{10}$
$x_{11}^3\ x_{12}^3$
$\mathcal{T}_{11}\ \mathcal{T}_{12}$

Στην αρχή, το LP επιστρέφει τα $\{x_i^0\}_{i=1}^{13}$. Στη συνέχεια, χωρίζουμε το χρόνο σε 4 εποχές και πληρώνουμε το ¨τίμημα του ανεξάρτητου rounding¨, που είναι το πολύ $2a.c(x)$ ($a.c(x)$: το κόστος απόκτησης που πληρώνει η $\{x_i^0\}_{i=1}^{13}$). Τρέχουμε τα βήματα του αλγορίθμου όπως τα εξηγήσαμε πρίν και σε κάθε εποχή $[t, t']$ παίρνουμε τα νέα κλασματικά συνδετικά δέντρα $\{x_i^1\}_{i=t}^{t'}$ πάνω στις αντίστοιχες κόκκινες ακμές. Παρατηρείστε οτι στην τελευταία εποχή κάνουμε κατευθείαν όλο το rounding αφού έχουμε μόνο ένα κλασματικά συνδετικό δέντρο (περίπτωση $z(E) = n - 1$). Λόγω του οτι στο στάδιο αφαίρεσης των κλασματικών κύκλων δεν αυξήσαμε το κόστος απόκτησης, σε κάθε επίπεδο του δέντρου πληρώνουμε το πολύ $2a.c(x)$ για κόστος απόκτησης. Συνεπώς, το κόστος απόκτησης της λύσης $\{\mathcal{T}_i\}_{i=1}^{13}$ είναι το πολύ $2a.c(x)$ επί το ύψος του δέντρου εκτέλεσης του αλγορίθμου. Λόγω **(1)** το ύψος αυτό είναι $O(\log n)$. Εν τέλει έχουμε προσέγγιση 3 στο κόστος κράτησης και προσέγγιση $O(\log n)$ στο κόστος απόκτησης.

4.2 Ακέραια ΓΠ για ειδικές περιπτώσεις του ΜΜΜ

Σε αυτήν την ενότητα παρουσιάζουμε δύο ακόμα αποτελέσματα αυτής της διπλωματικής. Δείχνουμε οτι τα ΓΠ για την ειδική περίπτωση του ΜΜΜ, όπου τα Μητροειδή είναι Μητροειδή Διαμέρισης, και για την περίπτωση Τ=2, είναι ακέραια.

Χρονομεταβαλλόμενη Βελτιστοποίηση σε Μητροειδή Διαμέρισης

Ενα είδος μητροειδούς είναι τα Μητροειδή Διαμέρισης:

Ορισμός 4.6. ***Μητροειδές Διαμέρισης***: *Έστω ένα σύνολο $\mathcal{S}$ και $S_1, S_2, ..., S_n$ μία διαμέριση του $\mathcal{S}$ και $k_1, k_2, ..., k_n$ μη αρνητικοί ακέραιοι. Έστω $\mathcal{I} = \{T \subseteq \mathcal{S} : \ |T \cap S_i| \leq k_i, \ \forall i \in [n]\}$.*

Παρατηρείστε οτι για να λυθεί το πρόβλημα της Χρονομομεταβαλλόμενης Βελτιστοποίησης σε Μητροειδή Διαμέρισης, αρκεί να λυθεί για $n = 1$. Το ΓΠ γι' αυτό το πρόβλημα έχει την εξής μορφή:

$$\begin{array}{ll} \text{minimize} & \sum_{t=1}^{T}\sum_{e\in\mathcal{S}} c_t(e)y_t(e) + \sum_{t=2}^{T}\sum_{e\in\mathcal{S}} a_t(e)z_t(e) \\ \text{subject to} & z_t(\mathcal{S}) = k, \quad \forall t \in [T] \\ & y_t(e) \geq z_t(e) - z_{t-1}(e), \quad \forall e \in \mathcal{S}, \quad t = 2, ..., T \\ & 0 \leq z_t(e) \leq 1, \ \forall e \in \mathcal{S}, \quad \forall t \in [T] \\ & y_t(e) \geq 0, \ \forall e \in \mathcal{S}, \quad t = 2, ..., T \end{array}$$

Παρατηρείστε οτι επιτρέπουμε τα κόστη απόκτησης να εξαρτώνται απο το χρόνο. Στο [13], οι συγγραφείς δείχνουν οτι αυτό το πρόβλημα είναι στο P. Εμείς δείχνουμε το εξής θεώρημα:

Θεώρημα 4.1. *Το Γραμμικό Πρόγραμμα για το πρόβλημα της Χρονομομεταβαλλόμενης Βελτιστοποίησης σε Μητροειδή Διαμέρισης είναι ακέραιο.*

T=2

Στην περίπτωση του MSTM, οπου Τ=2, το ΓΠ παίρνει την εξής μορφή:

$$\begin{array}{ll} \text{minimize} & \sum_e c_1(e)x_1(e) + \sum_e c_2(e)x_2(e) + \sum_e a(e)y(e) \\ \text{subject to} & x_t(E) = n - 1, \quad \forall t \in \{1, 2\} \\ & x_t(E(S)) \leq |S| - 1, \quad \forall S \subseteq V, \ \forall t \in \{1, 2\} \\ & x_t(e) \geq 0, \ \forall e \in E, \ \forall t \in \{1, 2\} \\ & y(e) \geq x_2(e) - x_1(e), \ \forall e \in E \\ & y(e) \geq 0, \ \forall e \in E, \end{array}$$

Στο [13], οι συγγραφείς δείχνουν οτι αυτό το πρόβλημα είναι στο P. Εμείς δείχνουμε το εξής θεώρημα:

Θεώρημα 4.2. *Το Γραμμικό Πρόγραμμα για το πρόβλημα της Χρονομομεταβαλλόμενης Βελτιστοποίησης σε Συνδετικά Δέντρα, με T=2, είναι ακέραιο.*

Για την ακρίβεια δείχνουμε οτι αυτό ισχύει στην περίπτωση που έχουμε δύο Μητροειδή και αυτά μπορεί να είναι διαφορετικά.

Bibliography

[1] Jacob Abernethy et al. ≪A regularization approach to metrical task systems≫. In: *International Conference on Algorithmic Learning Theory*. Springer. 2010, pp. 270–284.

[2] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. ≪Dynamic facility location via exponential clocks≫. In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2015, pp. 708–721.

[3] Matthew Andrews, Michel X Goemans, and Lisa Zhang. ≪Improved bounds for on-line load balancing≫. In: *Algorithmica* 23.4 (1999), pp. 278–301.

[4] Barbara M Anthony and Anupam Gupta. ≪Infrastructure leasing problems≫. In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2007, pp. 424–438.

[5] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. ≪Metrical task systems and the k-server problem on hsts≫. In: *Automata, Languages and Programming* (2010), pp. 287–298.

[6] Allan Borodin, Nathan Linial, and Michael E Saks. ≪An optimal on-line algorithm for metrical task system≫. In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 745–763.

[7] Niv Buchbinder et al. ≪Unified algorithms for online learning and competitive analysis≫. In: *Conference on Learning Theory*. 2012, pp. 5–1.

[8] Jack Edmonds. ≪Matroids and the greedy algorithm≫. In: *Mathematical programming* 1.1 (1971), pp. 127–136.

[9] Jack Edmonds. ≪Submodular functions, matroids, and certain polyhedra≫. In: *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen* 11 (1970).

[10] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. ≪Facility location in evolving metrics≫. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2014, pp. 459–470.

[11] Leah Epstein and Asaf Levin. ≪Robust algorithms for preemptive scheduling≫. In: *Proceedings of the 19th European conference on Algorithms*. Springer-Verlag. 2011, pp. 567–578.

[12] Albert Gu, Anupam Gupta, and Amit Kumar. ≪The power of deferral: maintaining a constant-competitive Steiner tree online≫. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM. 2013, pp. 525–534.

[13] Anupam Gupta, Kunal Talwar, and Udi Wieder. ≪Changing bases: Multistage optimization for matroids and matchings≫. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2014, pp. 563–575.

[14] Makoto Imase and Bernard M Waxman. ≪Dynamic Steiner tree problem≫. In: *SIAM Journal on Discrete Mathematics* 4.3 (1991), pp. 369–384.

[15] Nicole Megow et al. ≪The power of recourse for online MST and TSP≫. In: *Automata, Languages, and Programming* (2012), pp. 689–700.

[16] Adam Meyerson. ≪The parking permit problem≫. In: *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*. IEEE. 2005, pp. 274–282.

[17] Chandrashekhar Nagarajan and David P Williamson. ≪Offline and online facility leasing≫. In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2008, pp. 303–315.

[18] Hadas Shachnai, Gal Tamir, and Tami Tamir. ≪A theory and algorithms for combinatorial reoptimization≫. In: *Latin American Symposium on Theoretical Informatics*. Springer. 2012, pp. 618–630.

[19] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[20] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.