Approximation Algorithms for Min-Max Resource Sharing and Malleable Tasks Scheduling

Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

(Dr. rer. nat.)

der Technischen Fakultät

der Christian-Albrechts-Universität zu Kiel

Hu Zhang

Kiel 2004

1. Gutachter: Prof. Dr. Klaus Jansen

2. Gutachter: Prof. Dr. Anand Srivastav

3. Gutachter: Prof. Dr. Denis Trystram

Tag der mündlichen Prüfung: 20.12.2004

Zum Druck genehmigt: 20.12.2004

Acknowledgment

A million thanks to Klaus Jansen, my supervisor, for his invaluable guidance. It was him who introduced me to the field and community of combinatorial optimization and approximation algorithms. Only with his inspiration, his advice and his patience was I able to start my research career. This thesis has gained much from his co-authorship in the corresponding papers. I am very grateful to him for his support, help and friendship.

Many thanks are due to Anand Srivastav, Denis Trystram and Roberto Solis-Oba, for their kind help to me in both my work and my career. Their support, concerning and encouragement has benefited my research and my future much. I wish to thank Denis for his providing me the opportunity of working in Grenoble. I thank Tamás Terlaky, for his offer of the postdoctoral fellowship at the McMaster University. I also thank Igor Averbakh for his kind offer at the University of Toronto.

I am very grateful to Jana Chlebíková, Aleksei Fishkin, Olga Gerber, Ute Iaquinto, Qiang Lu, Marian Margraf, Gitta Marchand, Parvaneh Karimi-Massouleh, Brigitte Preuß, Deshi Ye and Guochuan Zhang, my former and current colleagues in the group "Theory of Parallelism" at the Institute of Computer Science and Applied Mathematics, the Christian-Albrechts-University of Kiel. They have made me to enjoy the good time in Kiel. I would never forget our interesting group activities. My special thanks are due to Jana, Gitta, Deshi and Guochuan for their aid for my finishing this thesis.

I am grateful to thank Wen Chen, Jiansong Deng, Pierre-Francois Dutot, Zhongyong Fan, Michael Grigoriadis, Jianming Guo, Yu Gu, Zhanyao Ha, Rastislav Kralovic, Jean-Francois Lalande, Xiaobo Li, Pu Ma, Michel Syska, Lixin Wu, Nianwei Xing, Min Zhang, Ping Zhang, Ying Zhang, Ning Zhao, Wei Zhong, Jun Zhou, Yan Zhou, and Xianli Zhu, for their friendship. Their help to my work, to my future and to my

daily life is my treasure. It is my pleasure that my thank list is increasing everyday, while it is a big pity that I can not list all the names of people who do help me.

My research was supported in part by the Graduiertenkolleg 357 "Effiziente Algorithmen und Mehrskalenmethoden" supported by Deutsche Forschungsgemeinschaft, by EU Thematic Network APPOL I and II "Approximation and Online Algorithms for Optimization Problems" (IST-1999-14084, IST-2001-32007), by EU Project ARACNE, Research Training Network "Approximation and Randomized Algorithms in Communication Networks" (HPRN-CT-199-00112), by DAAD Project Procope "Scheduling of Malleable Tasks", and by EU Project CRESCCO, Critical Resource Sharing for Cooperation in Complex Systems, IST-2001-33135. I would like to thank the Institute of Computer Science and Applied Mathematics for providing me with the opportunity of working here and the nice facilities and working environment.

I am sincerely grateful to my parents, my grandparents, my uncle, and Mei Chen, for their understanding and encouragement, for their concerning and care, for their love and devotion. I dedicate this thesis to them.

Kiel, December 2004 Hu Zhang

Abstract

This thesis deals with approximation algorithms for problems in mathematical programming, combinatorial optimization, and their applications.

We first study the Convex Min-Max Resource-Sharing problem (the Packing problem as the linear case) with M nonnegative convex constraints on a convex set B, which is a class of convex programming. In general block solvers are required for solving the problems. Based on a Lagrangian decomposition method via a logarithmic potential reduction, we generalize the algorithm by Grigoriadis et al. to the case with only weak approximate block solvers (i.e. with only constant, logarithmic or even worse approximation ratios). In this way we present an approximation algorithm for the Convex Min-Max Resource-Sharing problem that needs at most $O(M(\ln M+\varepsilon^{-2}\ln\varepsilon^{-1}))$ calls to the block solver for any given relative accuracy $\varepsilon \in (0,1)$. It is the first bound independent of the data and the approximation ratio of the block solver for this general Convex Min-Max Resource-Sharing problem. In the case of small ratios we propose an improved approximation algorithm with at most $O(M(\ln M + \varepsilon^{-2}))$ calls to the block solver.

As an application of the Convex Min-Max Resource-Sharing problem, we study the Multicast Congestion problem in communication networks. We are given a graph $G = (V, E)$ to represent a communication network where $|V| = n$ and $|E| = m$, and a set of multicast requests $S_1, \ldots, S_k \subseteq V$. A feasible solution to the Multicast Congestion problem in communication networks is a set of k trees $T_1, \ldots, T_k$ where T_i connects the vertices in S_i. The goal of the Multicast Congestion problem is to find a solution of k trees minimizing the maximum edge congestion (the number of times an edge is used). We develop a randomized asymptotic approximation algorithm for the Multicast Congestion problem in communication networks based on our approximation algorithm for the Convex

Min-Max Resource-Sharing problem. We show that our algorithm is able to overcome the difficulties of an exponential number of variables and a weak block solver for the Steiner tree problem. For any given relative accuracy $\varepsilon \in (0, 1)$ our approximation algorithm delivers a solution with a constant factor times the optimal value with an additive term in $O(m(\ln m + \varepsilon^{-2} \ln \varepsilon^{-1})(k\beta + m \ln \ln(m\varepsilon^{-1})))$ time, where β is the complexity of the approximate solver for the Steiner tree problem.

We study the Minimum Range Assignment problem in static ad-hoc networks with general structure as another application of the Convex Min-Max Resource-Sharing problem, where the transmission distances can violate the triangle inequality. We consider two versions of the Minimum Range Assignment problem, where the communication graph has to fulfill either the h-strong connectivity condition (Min-Range(h-SC)) or the h-broadcast condition (Min-Range(h-B)). Both homogeneous and non-homogeneous cases are studied. By approximating arbitrary edge-weighted graphs by paths, we present probabilistic $O(\log n)$-approximation algorithms for Min-Range(h-SC) and Min-Range(h-B). The result for Min-Range(h-B) matches the lower bound by Rossi for the case that the triangle inequality holds for transmission distances (which is a special case of our model). Furthermore, we show that if the network fulfils certain property and the distance power gradient α is sufficiently small, the approximation ratio is improved to $O((\log \log n)^{\alpha})$. In addition, the results are generalized to the mobile ad-hoc networks (dynamical networks).

We also study the problem of Scheduling Malleable Tasks with Precedence Constraints. We are given m identical processors and n tasks. For each task the processing time is a discrete function of the number of processors allotted to it. In addition, the tasks must be processed according to the precedence constraints. The goal is to minimize the makespan (maximum completion time) of the resulting schedule. The best previous approximation algorithm (that works in two phases) by Lepére et al. has a ratio $3+\sqrt{5} \approx 5.236$. In the first phase a time-cost tradeoff problem is solved approximately. With a binary search procedure, each task is allotted a

number of processors. In the second phase a variant of the list scheduling algorithm is used. In phase one a rounding parameter $\rho = 1/2$ and in phase two an allotment parameter $\mu = (3m - \sqrt{5m^2 - 4m})/2$ is employed, respectively. In the first phase of our algorithm, we formulate a linear program to solve the allotment problem and avoid the binary search procedure. We study the rounding technique carefully and shift the rounding parameter ρ in the second phase. Then we develop a min-max nonlinear program, whose objective value is an upper bound of the approximation ratio of our algorithm. By exploring the structure of the nonlinear program, we set $\rho = 0.43$, and vary the value of μ accordingly to obtain an improved approximation algorithm with a ratio at most $100/43 + 100(\sqrt{4349} - 7)/2451 \approx 4.730598$. We show that our settings of ρ and μ are very close to the asymptotic best choices.

Finally, Based on an interesting model for malleable tasks with continuous processor allotments by Prasanna et al., we define two natural assumptions for malleable tasks: the processing time of any malleable task is non-increasing in the number of processors allotted, and the speedup in concave in the number of processors. We show that under these assumptions the work function of any malleable task is non-decreasing in the number of processors and is convex in the processing time. Furthermore, we propose a two-phase approximation algorithm for the Scheduling Malleable Tasks with Precedence Constraints. In the first phase we solve a linear program to obtain a fractional allotment for all tasks. By rounding the fractional solution with a new technique, each malleable task is assigned a number of processors. In the second phase the variant of the list scheduling algorithm is also employed. In the phases we also use a new rounding parameter ρ and an allotment parameter μ, respectively. By choosing $\rho = 0.26$ and corresponding μ, we show (via a min-max nonlinear program) that the approximation ratio of our algorithm is at most $100/63 + 100(\sqrt{6469} + 13)/5481 \approx 3.291919$. We also show that our result is very close to the best asymptotic one.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A lot of exact and/or approximation algorithms have been proposed for problems either polynomial time solvable ($\mathcal{P}$) or non-deterministic polynomial time solvable ($\mathcal{NP}$). In general for the optimization problems in $\mathcal{P}$, main interests are paid for design of fast algorithms to obtain the optimal solutions with less running times. However, with the increasing demands of large data input, exact algorithms can not always serve due to the limited capacity of current computer systems. Therefore another trend is to find fast approximation algorithms for this class of problems without much loss of quality of solution.

Besides requirements for exact algorithms for optimization problems in $\mathcal{NP}$ in few fields (e.g., bioinformatics), this class of problems attracts more attention for developing approximation algorithms. Denote by $\mathcal{A}$ an approximation algorithm for a problem P. For any instance $\mathcal{I}$ of the problem P, we denote by $\mathcal{OPT}(\mathcal{I})$ the optimal solution and by $\mathcal{ALG}(\mathcal{I})$ the solution delivered by the algorithm $\mathcal{A}$. The quality of the algorithm $\mathcal{A}$ is measured by the *performance ratio* or *approximation ratio* r. In the *worst-case analysis*, for a minimization problem where the goal is to minimize the objective function, the approximation ratio r is defined as follows:

$$r = \sup_{\mathcal{I}} \frac{\mathcal{ALG}(\mathcal{I})}{\mathcal{OPT}(\mathcal{I})}.$$

For a maximization problem where the goal is to maximize the objective function, the approximation ratio r has the following definition:

$$r = \sup_{\mathcal{I}} \frac{\mathcal{OPT}(\mathcal{I})}{\mathcal{ALG}(\mathcal{I})}.$$

The development of approximation algorithms for optimization problems in $\mathcal{NP}$ follows two directions. One direction is to design approximation algorithms with better ratios in order to match or approach the lower bound of the approximation ratio for the problems. The other direction is to design fast approximation algorithms running in less time without too bad approximation ratios. The effort in the first direction has more meaning in theoretical aspect as the approximation algorithms with improved ratio guarantee that the solution developed by the algorithms can not be too bad. Since most results are based on worst-case analysis, the ratios of the approximation algorithms in average case are not clear though there have been some steps in this topic. In practice it can happen that an algorithm with better approximation ratio has worse performance for most instances. Thus regardless of improvement of approximation ratio, some effort is paid to reduce the overall running time of the approximation algorithms. Both directions fit certain requirement in accordance with the applications.

In this thesis, we shall first discuss approximation algorithms for a type of *linear program* or *convex program*. In a convex program, we are given a set of convex constraints and a convex objective function with variables defined over a convex set. The goal is to minimize the objective function. Convex programming, together with its linear case where the objective function is linear, are among the central problems in mathematical programming. They have been widely applied in many areas such as computer science, communications, operational research, industrial engineering, finance and so on. Then we shall show how to apply our algorithms in communication networks to reduce the edge congestion of data flow, which is a crucial issue for the real case of large demands by clients whit limited network hardware resources. Another application studied in this thesis is the problem to reduce energy consumption of mobile devices in wireless communication networks to extend the lifetime of the topological structure of the networks for keeping communication quality. Finally, we will study a kind of *scheduling* problem in $\mathcal{NP}$. A scheduling problem is usually characterized by *processors* (*machines*), *jobs* (*tasks*) and the *objective function*. The processors can be *identical*, *related* or *unrelated*. In addition, in computer science the structure of the parallel processors can be PRAM, line, mesh, hypercube and so on. The jobs can have *due dates* such that they must be executed by the due dates. In the *online* model (where not all information of input is given at the beginning),

jobs can have *release times* such that their information is unknown before the release times. There are also many models of parallel jobs such as *preemptive* jobs and *malleable* jobs. Due to the data flow the jobs can also have *precedence constraints* such that one job can not be executed unless all of its predecessors are completed. If the objective functions are *makespan* (the maximum *completion time*) or sum of weighted completion time (average completion time as a special case), then the goal of the problems are usually to minimize the objective functions. If the objective function is overall throughput, then the goal of the scheduling problem is usually a maximization one. Details of the models and algorithms for scheduling problems can be found in [82].

For linear programming, the *simplex method* discovered by Dantzig [27] has been widely applied in practice. It works very efficiently in many applications and has been used to establish some softwares and libraries. However, simplex method is shown to have a running time exponential in input size (the number of variables or the number of constraints). The first polynomial time algorithm for the linear programming is the *ellipsoid method* developed by Khachiyan [72]. Then the ellipsoid method is applied to combinatorial optimization [49, 68]. Unfortunately, the running time of the ellipsoid method is very large, though polynomial in input size. Another polynomial time algorithm for linear programming and convex programming was proposed by Karmarkar [66], which is faster both theoretically and practically compared with ellipsoid method.

The polynomial running times of both ellipsoid method and Karmarkar's method show that linear programming and convex programming could be in $\mathcal{P}$. However, the running times are quite large. Thus, as mentioned before, when the input size is large, the running time can exceed the capacity of current computer systems. Therefore to solve such kind of problems (which are common in real applications), we need to consider the tradeoff between accuracy and speed. The strategy is to design some fast but approximation algorithms for linear programming or convex programming. The idea is similar to the approximation algorithms for problems in $\mathcal{NP}$. We use the approximation ratio defined as before for measuring the quality of the algorithms.

In this thesis, we consider the following Convex Min-Max Resource-Sharing

problem, which is a special case of linear or convex program:

$$\begin{aligned} \min \quad & \lambda \\ \text{s.t.} \quad & f_m(x) \leq \lambda, \quad m = 1, \ldots, M; \\ & x \in B, \end{aligned}$$

where $f : B \to \mathbb{R}_+^M$ is a vector of M continuous convex functions defined on a non-empty convex compact set $B \subseteq \mathbb{R}^N$.Without loss of generality we assume $\lambda^* > 0$. Otherwise we can solve a system of equations $f(x) = 0$ to obtain the optimal solution. The functions f_m, $m \in \{1, \ldots, M\}$, are called the *coupling constraints*. We use the *Lagrangian decomposition* or *Lagrangian relaxation* method to design approximation algorithms for the Convex Min-Max Resource-Sharing problem. Assume that we are given a weak $c(1 + O(\varepsilon))$-approximate block solver for a given relative accuracy $\varepsilon \in (0, 1)$ which is used to approximately solve the block problem and called as a subroutine, our algorithm can find a $c(1 + \varepsilon)$-approximate solution for the Convex Min-Max Resource-Sharing problem, where $c \geq 1$ is the approximation ratio. The *coordination complexity* (bound on the number of iterations) is $O(M(\ln M + \varepsilon^{-2} \ln \varepsilon^{-1}))$, which is the first result independent of data and approximation ratio while only polynomial in M and ε^{-1} with only weak block solvers. For a special case of small c, we propose a fast approximation algorithm running within only $O(M(\ln M + \varepsilon^{-2}))$ iterations.

As applications of the Convex Min-Max Resource-Sharing problem, we consider two important problems in communication networks. First we study the Multicast Congestion problem in communication networks. Given a communication network represented by an undirected graph $G = (V, E)$ with *multicast requests* $S_1, \ldots, S_k \subseteq V$, we shall find for each request S_i a trees in G connecting all vertices in S_i such that the maximum number of times an edge in E is used in all the k trees is minimized. The Multicast Congestion problem in communication networks is in $\mathcal{NP}$ and we develop a randomized asymptotic approximation algorithm with an improved running time. The strategy is to formulate the Multicast Congestion problem as an integer linear program in the form of the Packing problem. Then we can apply our algorithm for the Convex Min-Max Resource-Sharing problem and the Packing problem to solve the linear programming relaxation. Here

the approximate block solver has a ratio c is greater than 1. The other problem is the MINIMUM RANGE ASSIGNMENT problem in *ad-hoc networks*. The ad-hoc network is a type of wireless communication network without infrastructure backbone in which the messages are delivered via intermedia stations. The network topology can vary due to the mobility of stations. A crucial problem in ad-hoc networks is the energy consumption. Higher communication quality leads to more energy consumption while increase of the lifetime of the network requires low energy consumption. Therefore an optimization problem, the MINIMUM RANGE ASSIGNMENT problem, arises with such a tradeoff. In general the MINIMUM RANGE ASSIGNMENT problem is in $\mathcal{NP}$. Furthermore, most existing approximation algorithms are only for networks on one dimensional Euclidean spaces. We propose the first general model of the MINIMUM RANGE ASSIGNMENT problem in ad-hoc networks. In our model the network is represented by an arbitrary undirected graph with n stations where the triangle inequality can be violated. In addition, the number of hops is bounded to keep the communication quality. All previous models are special cases of ours. Furthermore, we develop a probabilistic $O(\log n)$-approximation algorithm for the MINIMUM RANGE ASSIGNMENT problem. To determine the probability distribution a linear program of the form of the PACKING problem is designed. The corresponding block problem has only $O(\log n)$-approximate solvers. Our algorithm for the CONVEX MIN-MAX RESOURCE-SHARING problem and the PACKING problem is employed to obtain a data independent running time. It is worth noting that the lower bound for one class of the MINIMUM RANGE ASSIGNMENT problem is also $O(\log n)$.

We also study the problem of SCHEDULING MALLEABLE TASKS WITH PRECEDENCE CONSTRAINTS, which is a scheduling problem in $\mathcal{NP}$. We are given m identical processors and n parallel jobs. Each job can be executed on any number of available processors. The processing time of each job is a discrete function depending on the number of processors allotted to it. The processing time function and the processor number fulfil certain monotonous assumption. Furthermore, precedence constraints exist among the jobs such that the jobs can not be scheduled at arbitrary time slots. The previous best algorithm [81] gives a 5.236-approximation algorithm where a two-phase strategy is applied. In the first phase each job is allotted a num-

ber of processors by solving a discrete time-cost tradeoff problem. In the second phase a new allotment is generated and a variant of the list scheduling algorithm is employed and a feasible schedule is delivered with a makespan at most 5.236 times the makespan of the optimal schedule. In this thesis we propose improved approximation algorithms based on the algorithm in [81]. The first algorithm we design has an approximation ratio of 5.162. Then we analyze the rounding phase carefully and obtain the second approximation algorithm with a ratio of 4.7306. At last for a special case of the processing time function we develop a 3.2919-approximation algorithm. These algorithms are the best known for the problem of Scheduling Malleable Tasks with Precedence Constraints.

The thesis is organized as follows: In Chapter 2 we study the Convex Min-Max Resource-Sharing problem and its approximation algorithms. Applications of the Convex Min-Max Resource-Sharing problem are shown in Chapter 3, including the Multicast Congestion problem in communication networks and the Minimum Range Assignment problem in ad-hoc networks. Finally, in Chapter 4 we develop approximation algorithms for the problem of Scheduling Malleable Tasks with Precedence Constraints. Parts of this thesis are published or will be published in [60, 61, 114, 110, 115, 62, 63, 19].

Chapter 2

Approximation Algorithms for the Min-Max Resource Sharing Problem

In this chapter we present approximation algorithms based on Lagrangian decomposition via a logarithmic potential reduction to solve a general packing or min-max resource sharing problem with M nonnegative convex constraints over a convex set B. We generalize a method by Grigoriadis et al to the case with weak approximate block solvers (i.e. with only constant, logarithmic or even worse approximation ratios). We show that the algorithm needs at most $O(M(\ln M + \varepsilon^{-2} \ln \varepsilon^{-1}))$ calls to the block solver for a given relative accuracy $\varepsilon \in (0, 1)$, a bound independent of the data and the approximation ratio of the block solver. For small approximation ratios the algorithm needs at most $O(M(\ln M + \varepsilon^{-2}))$ calls to the block solver.

2.1 Introduction

We consider the following general Convex Min-Max Resource-Sharing problem:

$$(P) \qquad \lambda^* = \min\{\lambda | f(x) \leq \lambda e, x \in B\},$$

where $f : B \to \mathbb{R}_+^M$ is a vector of M continuous convex functions defined on a nonempty convex compact set $B \subseteq \mathbb{R}^N$, and e is the vector of all ones. Without loss of generality we assume $\lambda^* > 0$. Otherwise we can solve the system of equations $f(x) = 0$ to obtain the optimal solution. The functions f_m, $m \in \{1, \ldots, M\}$, are the *coupling constraints*. In addition, we denote by $\lambda(x) = \max_{m \in \{1,\ldots,M\}} f_m(x)$ for

any fixed $x \in B$. Therefore the problem can be reformulated as follows: To find an $x^* \in B$ such that

$$(P') \qquad \lambda^* = \lambda(x^*) = \min\{\lambda(x), x \in B\}.$$

If the coupling constraints $f_m(x)$, $m \in \{1, \ldots, M\}$, are linear functions, the Convex Min-Max Resource-Sharing problem is reduced to the Packing problem [45, 46, 89, 109]. The linear functions $f_m(x)$, $m \in \{1, \ldots, M\}$, are the *packing constraints*.

There are many applications of the Convex Min-Max Resource-Sharing problem or the Packing problem. Typical examples include scheduling on unrelated machines, job shop scheduling, network embeddings, Held-Karp bound for TSP, minimum-cost multicommodity flows, maximum concurrent flow, bin covering, spreading metrics, approximation of metric spaces, graph partitioning, multicast congestion in communication networks, and range assignment in static ad-hoc networks [5, 16, 32, 42, 47, 56, 59, 78, 89, 110, 114].

Theoretically the optimal solution of a convex or linear program with the form of (P) could be obtained by polynomial time algorithms. However, the running times are still large, which can be impractical in many applications. Furthermore, in some algorithms the optimal solution is not required and an approximate solution suffices (e.g. [71]). In addition, in some applications the size of (P) can be exponential in the size of input (e.g. [5, 61, 110]). Thus the demand of approximation but fast algorithms for the Convex Min-Max Resource-Sharing problem (P) arises.

Grigoriadis and Khachiyan [45, 46] proposed algorithms to compute an ε-approximate solution to the Convex Min-Max Resource-Sharing problem (P); i.e. for a given accuracy $\varepsilon > 0$,

$$(P_\varepsilon) \qquad \text{compute } x \in B \text{ such that } f(x) \leq (1+\varepsilon)\lambda^* e.$$

The approaches are based on the following Lagrangian duality relation:

$$\lambda^* = \min_{x \in B} \max_{p \in P} p^T f(x) = \max_{p \in P} \min_{x \in B} p^T f(x),$$

where $P = \{p \in \mathbb{R}^M | \sum_{m=1}^{M} p_m = 1, p_m \geq 0\}$ is the set of *price vectors*. For any fixed $x \in B$, we can choose the price vector p as follows: We set $p_k = 1$ and all

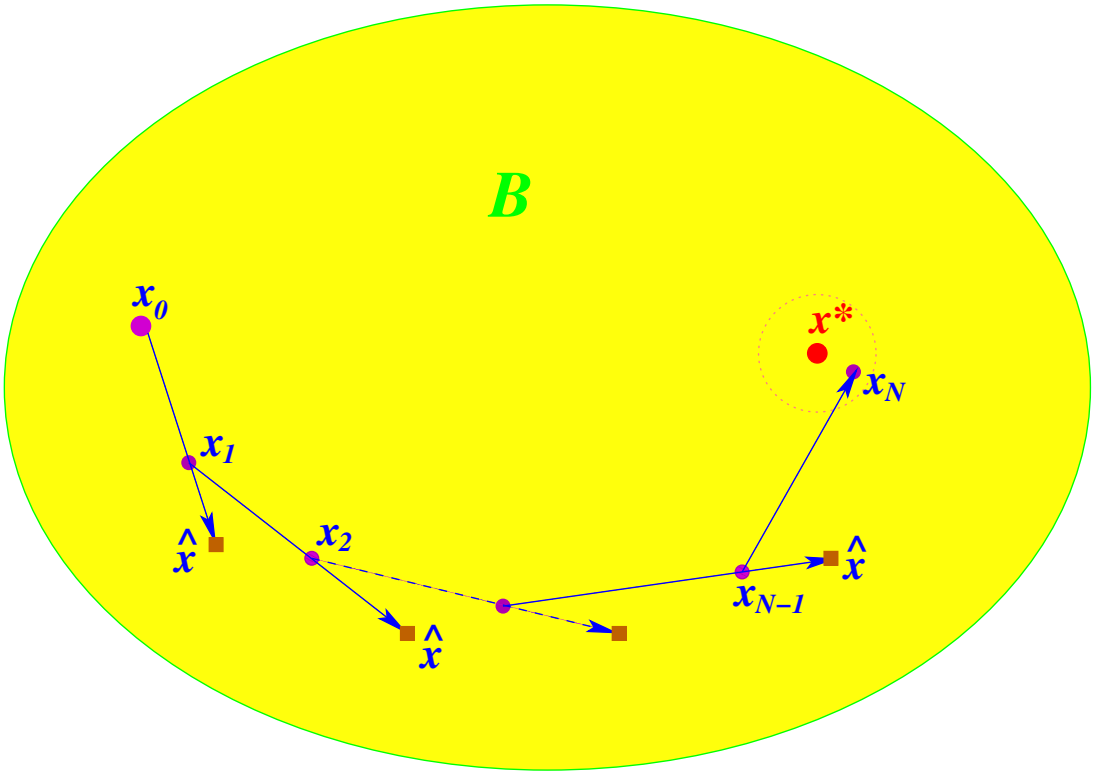


Figure 2.1: Lagrangian decomposition method.

of other components $p_m = 0$ for $m \neq k$, where $f_k(x) = \lambda(x)$. In this way when $x = x^*$ we obtain the optimal value λ^* and the first equation of the duality relation holds. In addition, we can exchange the order of p and x by the duality relation to obtain the second equation. Denoting by $\Lambda(p) = \min_{x \in B} p^T f(x)$, we have that $\Lambda(p) \leq \lambda^* \leq \lambda(x)$ for any pair x and p. Furthermore a pair $x \in B$ and $p \in P$ is optimal, if and only if $\lambda(x) = \Lambda(p)$. The corresponding ε-approximate dual problem has the following form:

$$(D_\varepsilon) \qquad \text{compute } p \in P \text{ such that } \Lambda(p) \geq (1 - \varepsilon)\lambda^*.$$

The *Lagrangian* or *price-directive decomposition* method is an iterative strategy that solves the primal problem (P_ε) and its dual problem (D_ε) by computing a sequence of pairs x and p to approximate the optimal pair x^* and $p(f(x^*))$ from above and below respectively (Figure 2.1). One such Lagrangian decomposition step (that is called also a *coordination step*) consists of the following three substeps:

Step 1 (Computation of the price vector) Using the current $x \in B$ the coordinator computes a price vector $p = p(f(x)) \in P$ corresponding to the coupling constraints.

Step 2 (Block optimization) The coordinator calls a block solver as an oracle to compute an (approximate) solution $\hat{x} \in B$ to the block problem for the computed price vector p.

Step 3 (New iterate) The coordinator makes a move from x to $(1-\tau)x+\tau\hat{x}$ with an appropriate step length $\tau \in (0,1]$. The new iterate is a convex combination of the previous solution x and solution $\hat{x}$ delivered by the block solver.

Grigoriadis and Khachiyan [46] proved that the approximate primal problem (P_ε) and the approximate dual problem (D_ε) can be solved within $O(M(\ln M + \varepsilon^{-2}\ln\varepsilon^{-1}))$ iterations. In each iteration the algorithm calls to a t-approximate block solver that solves the block problem for a given tolerance $t = O(\varepsilon)$:

$$\begin{aligned} ABS(p,t) \qquad & \text{compute } \hat{x} = \hat{x}(p) \in B \\ & \text{such that } p^T f(\hat{x}) \le (1+t)\ \min\{p^T f(y) | y \in B\}. \end{aligned}$$

Their methods use either an exponential or a standard logarithmic potential function [45, 46] and is based on a ternary search procedure that repeatedly scales the functions $f(x)$. Villavicencio and Grigoriadis [109] proposed a modified logarithmic potential function to avoid the scaling phases of $f(x)$ and to simplify the analysis. The number of iterations used in [109] is also bounded by $O(M(\ln M + \varepsilon^{-2}\ln\varepsilon^{-1}))$. Furthermore, Grigoriadis et al [48] studied a Concave Max-Min Resource-Sharing problem (the Covering problem in the linear case), that is orthogonal to the Convex Min-Max Resource-Sharing problem studied here and defined later. They showed that the bound on the number of iterations or block optimization steps is only $O(M(\ln M + \varepsilon^{-2}))$ for the Concave Max-Min Resource-Sharing problem. Therefore it is natural to conjecture that one can improve the number of iterations for the Convex Min-Max Resource-Sharing problem. In fact we reduce the number of iterations for the Convex Min-Max Resource-Sharing problem (P_ε) and the dual problem (D_ε) to $O(M(\ln M + \varepsilon^{-2}))$.

On the other hand, in general the block problem may be hard to approximate [5, 16, 32, 110]. This means that the assumption to have a block solver with accuracy $t = O(\varepsilon)$ is too strict. Therefore we consider in this paper the case that we have only a weak approximate block solver. A (t,c)-approximate block solver is defined

as follows:

$$\begin{aligned} ABS(p,t,c) \qquad & \text{compute } \hat{x} = \hat{x}(p) \in B \\ & \text{such that } p^T f(\hat{x}) \leq c(1+t)\min\{p^T f(y) | y \in B\}, \end{aligned}$$

where $c \geq 1$ is the approximation ratio of the weak approximate block solver. The goal is now to solve the following approximate primal problem (using the weak block solver):

$$(P_{\varepsilon,c}) \qquad \text{compute } x \in B \text{ such that } f(x) \leq c(1+\varepsilon)\lambda^* e.$$

The corresponding approximate dual problem has the form:

$$(D_{\varepsilon,c}) \qquad \text{compute } p \in P \text{ such that } \Lambda(p) \geq \frac{1}{c}(1-\varepsilon)\lambda^*.$$

In this chapter, we will present an approximation algorithm that for any accuracy $\varepsilon \in (0,1)$ solves the problem $(P_{\varepsilon,c})$ in at most

$$N = O(M(\ln M + \varepsilon^{-2}\ln\varepsilon^{-1}))$$

iterations or coordination steps. Each step requires a call to the weak block solver $ABS(p, O(\varepsilon), c)$ and an overhead of $O(M \ln\ln(M\varepsilon^{-1}))$ arithmetic operations. Furthermore for small ratio c such that $\ln c = O(\varepsilon)$ we improve the number of iterations to $O(M(\ln M + \varepsilon^{-2}))$.

2.1.1 Previous work and related results

Plotkin, Shmoys and Tardos [89] considered the feasibility variants of the Packing problem as follows: to find a point $x \in B$ such that $f(x) = Ax \leq (1+\varepsilon)b$ where A is the coefficient matrix with M rows and b is an M-dimensional vector. The problem was solved by Lagrangian decomposition using exponential potential reductions, and the numbers of iterations (calls to the corresponding block solver) in the algorithm is $O(\varepsilon^{-2}\rho\ln(M\varepsilon^{-1}))$, where $\rho = \max_{m\in\{1,\ldots,M\}}\max_{x\in B} a_m^T x/b_m$ is the *width* of B. In their algorithm, the infeasibilities are penalized and approximate feasible solutions to the system of linear inequalities are found. However, here the width ρ is data

dependent and the bound is only pseudo polynomial in the size of input. With similar technique, Young [111] studied also the PACKING problem. He proposed an algorithm that uses $O(\rho'(\lambda^*)^{-1}\varepsilon^{-2}\ln M)$ calls to the block solver, where $\rho' = \max_{1\le m\le M}\max_{x\in B} a_m^T x/b_m - \min_{1\le m\le M}\min_{x\in B} a_m^T x/b_m$ similar to the width and λ^* is the optimal value of the PACKING problem.

Based on a combinatorial idea of "variable-size increments" for the multicommodity flow problem, Garg and Könemann [42] proposed a $(1+\varepsilon)$-approximation algorithm to solve the PACKING problem within $O(M\varepsilon^{-2}\ln M)$ iterations which is independent of the width. They also use an exponential length function to capture congestion on an edge. Their running time for the maximum multicommodity flow problem was improved by Fleischer [34] with an idea of "round robin increments". In fact this idea was used in [108] to generate a fast algorithm for the PACKING problem (P_ε) that the set B is the product of K nonempty disjoint convex compact sets. The number of iteration is at most $O(K(\ln M)(\varepsilon^{-2} + \ln\min\{K, M\}))$. The similar idea was also applied in [65] for the maximum concurrent flow problem.

As for the case of weak approximate block solver, there are also a few results. For the PACKING problem, the algorithm in [111] can also be applied when only a weak block solve is available. The number of iterations is also at most $O(\rho'(\lambda^*)^{-1}\varepsilon^{-2}\ln M)$. Furthermore, Charikar et al [16] noticed that the result in [89] for the packing problem can be extended also to the case with weak block solvers with the same number $O(\varepsilon^{-2}\rho\ln(M\varepsilon^{-1}))$ of iterations. However, there was no results before for the CONVEX MIN-MAX RESOURCE-SHARING problem with only weak block solvers. It is worth noting that our algorithm is the first one for this problem and the coordination complexity is independent of data ρ, ρ', λ^* and approximation ratio c. It is the first of this kind. In [115] the possibility to slightly improve the running time for a special case was discussed.

A class of problem which is orthogonal to the CONVEX MIN-MAX RESOURCE-SHARING problem is the CONCAVE MAX-MIN RESOURCE-SHARING problem (called the COVERING problem for linear constraints) is as follows:

$$(C) \qquad \lambda^* = \max\{\lambda | f(x) \ge \lambda e, x \in B\},$$

where $f : B \to \mathbb{R}_+^M$ is a vector of M continuous concave functions defined on a

nonempty convex compact set $B \subseteq \mathbb{R}^N$. As mentioned before, Grigoriadis et al [48] proposed an algorithm for the approximate Concave Max-Min Resource-Sharing problem running in $O(M(\ln M + \varepsilon^{-2}))$ iterations. Plotkin et al [89] also studied the feasibility variant of Covering problem: to find a point $x \in B$ such that $f(x) = Ax \leq (1+\varepsilon)b$ where A is the coefficient matrix with M rows and b is an M-dimensional vector. Their algorithm needs $O(M + \rho \ln^2 M + \varepsilon^{-2}\rho \ln(M\varepsilon^{-1}))$ iterations, where $\rho = \max_{m\in\{1,\ldots,M\}} \max_{x\in B} a_m^T x/b_m$ is the width of B similar with the case of Packing problem. As argued before, their algorithm may only lead to pseudo polynomial time approximation algorithms. Jansen and Porkolab [58] studied the Concave Max-Min Resource-Sharing problem with only weak approximate block solvers and showed that at most $O(M(\ln M + \varepsilon^{-2} + \varepsilon^{-3} \ln c)$ coordination steps are necessary. Unfortunately, this bound depends also on c, the approximation ratio of the block solver. Recently, Jansen [54] improved the coordination complexity to also $O(M(\ln M + \varepsilon^{-2} \ln \varepsilon^{-1}))$, which matches the bound of the algorithm in [60] for the Convex Min-Max Resource-Sharing problem and is independent of data.

Another form of the Covering problem is as follows:

$$\lambda^* = \min\{c^T x | Ax \geq b, 0 \leq x \leq u\},$$

where c, $u \in \mathbb{R}_+^M$, $b \in \mathbb{R}_+^N$ and $A \in \mathbb{R}_+^{N\times M}$ have non-negative entries. For the approximate problem Fleischer [35] proposed an algorithm with at most a number of $O(M\varepsilon^{-2} \ln(MC))$ iterations, where $C = \max_{m\in\{1,\ldots,M\}} c_m u_m / \min_{m:c_m u_m>0} c_m u_m$. The bound has been improved by Garg et al [41] to $O(M\varepsilon^{-2} \ln M + \min\{N, \ln \ln C\})$. However, both results are only for the case of t-approximate block solver and the bounds are data dependent.

Besides, the mixed problem with both packing and covering constraints was proposed by Young [112]. His approximation algorithm can find a point $x \in B \subseteq \mathbb{R}_+^N$ such that the packing and covering constraints are violated at most by factors of $1+\varepsilon$ and $1-\varepsilon$, respectively. The number of iterations is at most $O(Md\varepsilon^{-2} \ln M)$ where d is the maximum number of constraints any variable appears in, which is bounded by M. Jansen also studied the mixed packing and covering problem and

improved the bound to $O(M\varepsilon^{-2}\ln(M\varepsilon^{-1}))$ [55]. This is the first result independent of data for the mixed problem.

A detailed survey of the Largrangian based algorithms for convex programming can be found in [73].

2.1.2 Main idea

Our algorithm is based on ideas in [46, 48, 109]. We use the modified logarithmic potential function proposed in [109]. Our algorithm is based on the scaling phase strategy such that the relative error tolerance σ_s in s-th scaling phase approaches the given relative tolerance ε gradually when s increases. The oracle $ABS(p, t, c)$ is called once in each iteration. We found that the stopping rules proposed in [46, 109] are too strict, and that the bound on the number of iterations could be $O(Mc^2(\ln(Mc) + \varepsilon^{-3}\ln c))$ by directly applying the method in [46], or $O(M(\ln M + \varepsilon^{-2} + \varepsilon^{-3}\ln c))$ by the method in [109]. This shows that with only the existing methods the bounds independent of data are not possible. Therefore we analyzed a combination of two stopping rules in order to obtain a running time independent of c. In fact our result is the first one independent of the width ρ (or ρ'), the optimal value λ^* and the approximation ratio c. For certain c small enough, we use an upper bound for the difference of the potential function values $\phi_t(x) - \phi_t(x')$ for arbitrary two iterates $x, x' \in B$ within one scaling phase similar to [48]. This enables us to show that the original method in [109] (with a slightly modified stopping rule) uses only $O(M(\ln M + \varepsilon^{-2}))$ coordination steps for c with $\ln c = O(\varepsilon)$.

2.2 Modified logarithmic potential function

In order to solve the packing problem (P), we use the Lagrangian decomposition method that is based on a special potential function. Similar to the approaches in [45, 46, 109], we use the idea of potential function to relax the coupling constraints. We are able to show that the approximation of minimum value of potential function corresponds to an approximation of λ^*. Thus the original Convex Min-Max Resource-Sharing problem can be replaced by finding a good approximate minimum point of the (smooth) potential function. The modified Karmarkar's potential

function is defined as follows:

$$\Phi_t(\theta, x) = \ln \theta - \frac{t}{M} \sum_{m=1}^{M} \ln(\theta - f_m(x)), \tag{2.1}$$

where $\theta \in \mathbb{R}_+$ and $x \in B$ are variables and $t \in (0, 1)$ is a fixed tolerance parameter (that is used in the approximate block solver $ABS(p, t, c)$). In our algorithm, we shall set values of t from $O(1)$ initially down to $O(\varepsilon)$ at last, where ε is the desired relative accuracy for the solution. The function Φ_t is well-defined for $\lambda(x) < \theta < \infty$ where $\lambda(x) = \max\{f_1(x), \ldots, f_M(x)\}$ and has the *barrier property*: $\Phi_t(\theta, x) \to \infty$ for $\theta \to \lambda(x)$ and $\theta \to \infty$.

We define the reduced potential function as the minimum of $\Phi_t(\theta, x)$ over all $\theta \in (\lambda(x), \infty)$ for a fixed $x \in B$, i.e.

$$\phi_t(x) = \Phi_t(\theta(x), x) = \min_{\lambda(x)<\theta<\infty} \Phi_t(\theta, x). \tag{2.2}$$

Lemma 2.1 *For any fixed $x \in B$, the minimizer $\theta(x)$ of $\Phi_t(\theta, x)$ is the unique root to the following equation:*

$$\frac{t}{M} \sum_{m=1}^{M} \frac{\theta}{\theta - f_m(x)} = 1. \tag{2.3}$$

Proof: For a fixed $x \in B$, the potential function $\Phi_t(\theta, x) \in C^1$ is well defined in $\theta \in (\lambda(x), \infty)$. Due to the barrier property, there must exist a minimum in the interval $(\lambda(x), \infty)$. Its first order partial derivative with respect to θ is:

$$(\Phi_t(\theta, x))'_\theta = \frac{1}{\theta} - \frac{t}{M} \sum_{m=1}^{M} \frac{1}{\theta - f_m(x)}.$$

Then the minimizer $\theta(x)$ is the root of equation $(\Phi_t(\theta, x))'_\theta = 0$, i.e. equation (2.3). Furthermore, the function

$$g(\theta) = \frac{t}{M} \sum_{m=1}^{M} \frac{\theta}{\theta - f_m(x)} = t + \frac{t}{M} \sum_{m=1}^{M} \frac{f_m(x)}{\theta - f_m(x)}$$

is strictly decreasing as $\theta - f_m(x) > 0$. Therefore the root $\theta(x)$ is unique in the interval $(\lambda(x), \infty)$ and the lemma is proved. □

2.2.1 Technical inequalities

In order to analyze the properties of potential functions and our algorithms described in Section 2.3, we need some technical lemmas. First for functions with continuous derivatives we have the following lemma:

Lemma 2.2 *For two functions $f(x)$, $g(x) \in C^1$,*

1. *if $f'(x) \geq g'(x)$ for any $x \in (a, b]$ and $f(b) = g(b)$, then $f(x) \leq g(x)$ for all $x \in (a, b]$, where a can tend to $-\infty$;*

2. *if $f'(x) \geq g'(x)$ for any $x \in [b, c)$ and $f(b) = g(b)$, then $f(x) \geq g(x)$ for all $x \in [b, c)$, where c can tend to ∞.*

Proof: We consider the first claim. Define a new function $h(x) = f(x) - g(x)$. Its first order derivative $h'(x) = f'(x) - g'(x)$ is nonnegative. For any $x \in (a, b)$, there exists a $\xi \in [x, b)$ such that

$$\frac{h(x) - h(b)}{x - b} = h'(\xi).$$

Since $h(b) = f(b) - g(b) = 0$, and $h'(\xi) \geq 0$, we have

$$\frac{f(x) - g(x)}{x - b} \geq 0.$$

Because $x \in (a, b)$, $x - b$ is negative. Therefore we have $f(x) \leq g(x)$ and the first claim is proved.

For the second claim, we define also a function $h(x) = f(x) - g(x)$. Similarly, for any $x \in (b, c)$, there exists a $\xi \in (b, c)$ such that

$$\frac{f(x) - g(x)}{x - b} = \frac{h(x) - h(b)}{x - b} = h'(\xi) \geq 0.$$

Because $x \geq b$, $f(x) \geq g(x)$ in this case. Then the lemma is proved. □

Now we are able to show the following inequalities based on the above properties:

Lemma 2.3 *The logarithmic function fulfils following inequalities:*

1. *For any* $x > 0$, $\ln x \leq x - 1$;

2. *For any* $x \geq -1/2$, $\ln(1 + x) \geq x - x^2$.

Proof: Denote by $f(x) = \ln x$ and $g(x) = x - 1$. Then $f'(x) = 1/x$ and $g'(x) = 1$. It is obvious that $f(1) = g(1) = 0$. For any $x \in (0, 1]$, $f'(x) \geq 1 = g'(x)$. According to Lemma 2.2, $\ln x = f(x) \leq g(x) = x - 1$ for all $x \in (0, 1]$. On the other hand, for any $x \in [1, \infty)$, $f'(x) \leq 1 = g'(x)$. Again, according to the second claim of Lemma 2.2, $\ln x = f(x) \leq g(x) = x - 1$ for all $x \in [1, \infty)$. Therefore for any $x \in (0, \infty)$ we have $\ln x \leq x - 1$ and the first inequality of the lemma is proved.

Now consider the second inequality. Denote by $f(x) = \ln(1+x)$ and $g(x) = x-x^2$. Note that $f(0) = g(0)$. Then $f'(x) = 1/(1+x)$ and $g'(x) = 1-2x$. Solving equation $1/(1 + x) = 1 - 2x$ gives two roots $x_1 = -1/2$ and $x_2 = 0$. Thus $f'(x) \geq g'(x)$ for $x \in [0, \infty)$ while $f'(x) \leq g'(x)$ for $x \in [-1/2, 0]$. With the similar argument and Lemma 2.2, $\ln(1 + x) \geq x - x^2$ for any $x \geq -1/2$. This completes the proof. □

2.2.2 Bounds on the minimizer $\theta(x)$ and the reduced potential function $\phi_t(x)$

We will show the bounds on $\theta(x)$ and $\phi_t(x)$ in the following two lemmas by the definition of $\theta(x)$ similar to those in [109] and [48].

Lemma 2.4 $\lambda(x)/(1 - t/M) \leq \theta(x) \leq \lambda(x)/(1 - t)$ *for any* $x \in B$.

Proof: Using equation (2.3) and the fact that $f_m(x) \leq \lambda(x)$ for all m we obtain the following relations:

$$1 = \frac{t}{M} \sum_{m=1}^{M} \frac{\theta(x)}{\theta(x) - f_m(x)} \leq \frac{t}{M} \sum_{m=1}^{M} \frac{\theta(x)}{\theta(x) - \lambda(x)} = \frac{t\theta(x)}{\theta(x) - \lambda(x)},$$

which implies the right inequality in the Lemma since $t < 1$. The left inequality is also based on (2.3) as

$$\frac{t}{M} \frac{\theta(x)}{\theta(x) - f_m(x)} \leq 1, \qquad m = 1, \ldots, M,$$

and in particular for that m for which $f_m(x) = \lambda(x)$. □

Lemma 2.5 $(1-t)\ln\lambda(x) \le \phi_t(x) \le (1-t)\ln\lambda(x) + t\ln(\exp(1)/t)$ *for any* $x \in B$.

Proof: By the definition (2.1), $\ln\theta(x) = \phi_t(x) + \frac{t}{M}\sum_{m=1}^{M}\ln(\theta(x) - f_m(x))$. Since $f_m(x) \ge 0$ for all m,

$$\ln\theta(x) \le \phi_t(x) + \frac{t}{M}\sum_{m=1}^{M}\ln\theta(x) = \phi_t(x) + t\ln\theta(x).$$

This gives $(1-t)\ln\theta(x) \le \phi_t(x)$ as $t < 1$ and proves the left inequality in the Lemma (using $\lambda(x) \le \theta(x)$ from Lemma 2.4). To show the right inequality of the Lemma, we take logarithms of both sides of (2.3) and obtain:

$$\ln t + \ln\left(\frac{1}{M}\sum_{m=1}^{M}\frac{\theta(x)}{\theta(x) - f_m(x)}\right) = 0.$$

Using the concavity of $\ln(\cdot)$ this implies:

$$\ln t + \frac{1}{M}\sum_{m=1}^{M}\ln\left(\frac{\theta(x)}{\theta(x) - f_m(x)}\right) \le 0.$$

Then we multiply by t and add $\ln\theta(x)$ to both sides. This gives

$$\phi_t(x) = \ln\theta(x) - \frac{t}{M}\sum_{m=1}^{M}\ln(\theta(x) - f_m(x)) \le (1-t)\ln\theta(x) - t\ln t,$$

which becomes

$$\phi_t(x) \le (1-t)\ln\lambda(x) - (1-t)\ln(1-t) - t\ln t,$$

by the right inequality of Lemma 2.4. Finally we have $\ln z \le z - 1$ for all $z > 0$ according to Lemma 2.3. Then it can be proved that $-\ln(1-t) \le t/(1-t)$ using $z = 1/(1-t) > 0$. Substituting this into the above inequality provides the upper bound of $\phi_t(x)$. □

Lemmas 2.4 and 2.5 show (for certain sufficiently small values of t) that the minimum $\theta(x)$ approximates $\lambda(x)$ and that the reduced potential function $\phi_t(x)$ approximates $\ln\lambda(x)$ closely. This gives us the possibility to solve the approximation problem $(P_{\varepsilon,c})$ by minimizing the smooth function $\phi_t(x)$ over $x \in B$ based on these two lemmas.

2.2.3 Price vector function

The price vector $p(x) \in \mathbb{R}_+^M$ is defined as follows similar to [109]:

$$p_m(x) = \frac{t}{M} \frac{\theta(x)}{\theta(x) - f_m(x)}, \qquad m = 1, \ldots, M. \tag{2.4}$$

In view of the definition above, the following lemma holds:

Lemma 2.6 *The price vector $p(x)$ in (2.4) is in the set P, and $p(x)^T f(x) = \theta(x)(1-t)$ for any $x \in B$.*

Proof: Since $\theta(x) \geq f_m(x)$ for all $m = 1, \ldots, M$, all $p_m(x)$ are nonnegative. According to equation (2.3), for any $x \in B$, $\sum_{m=1}^M p_m(x) = 1$. Therefore $p(x) \in P$. Using equation (2.3) and definition (2.4)

$$\begin{aligned} p(x)^T f(x) &= \frac{t}{M} \sum_{m=1}^{M} \frac{\theta(x) f_m(x)}{\theta(x) - f_m(x)} \\ &= \frac{t\theta(x)}{M} \sum_{m=1}^{M} \left(\frac{\theta(x)}{\theta(x) - f_m(x)} - 1 \right) \\ &= \theta(x)(1-t). \end{aligned}$$

□

With the definition (2.4), we can just simply compute the price vector p from (2.4), which is easier to obtain compared with other methods (e.g. using the exponential potential function).

Furthermore, by Lemma 2.6, for t small enough, the dual value $p^T f(x)$ is only slightly less than $\theta(x)$, the minimum of the potential function. This shows that the dual value $p^T f(x)$ is also an approximation of $\lambda(x)$. Therefore the primal problem can also be solved by obtaining approximate dual value and in fact that is what we need to construct our algorithm and the base of stopping rules.

2.3 The approximation algorithm

The minimum dual value $\Lambda(p)$ can be approximated by the dual value $p^T f(\hat{x})$, where $\hat{x}$ is the block solution computed by the (t, c)-approximate block solver for the given

price vector p. Furthermore, to establish the stopping rules of each scaling phase in the approximation algorithm, the value of the duality gap should be estimated in each iteration. For our first stopping rule we use the following parameter ν:

$$\nu = \nu(x, \hat{x}) = \frac{p^T f(x) - p^T f(\hat{x})}{p^T f(x) + p^T f(\hat{x})}. \tag{2.5}$$

If $\nu = O(\varepsilon)$, then the duality gap is also quite small. But for larger ν close to 1, the gap can be extremely large (see also Subsection 2.3.1). Therefore, we define a second parameter w_s. Let σ_s be the relative error tolerance of the s-th scaling phase. Then the parameter w_s is given by:

$$w_s = \begin{cases} \dfrac{1+\sigma_1}{(1+\sigma_1/3)M}, & \text{for the first scaling phase;} \\ \dfrac{1+\sigma_s}{1+2\sigma_s}, & \text{otherwise.} \end{cases} \tag{2.6}$$

Let x_s be the solution of sth scaling phase. Then the two stopping rules used in the s-th scaling phase are:

$$\begin{aligned} &\textit{Rule } 1: && \nu \le \sigma_s/6; \\ &\textit{Rule } 2: && \lambda(x) \le w_s\, \lambda(x_{s-1}). \end{aligned} \tag{2.7}$$

Grigoriadis et al [48, 109] used either only the first stopping rule or the rule $p^T f(x) - p^T f(\hat{x}) \le \sigma_s \theta(x)/2$ that is similar to the first rule. In the case of only weak block solvers available, such stopping rules are not sufficient to obtain a bound on the number of iterations independent of the ratio c. It may happen that the block solver is called more times than what necessary. Therefore we have introduced the second stopping rule to make sure that the scaling phase stops as soon as the solution meets the requirement of the phase. On the other hand the first stopping rule is needed to have always a constant bound on decrement in the potential function (see Subsection 2.3.1).

The algorithm works now as follows. First we apply the scaling phase strategy. In each scaling phase the relative error tolerance σ_s is set. Then based on the known pair of x and p, a solution $\hat{x}$ is delivered by the approximate block solver. Afterwards an appropriate convex combination of the old solution x and the block solution $\hat{x}$

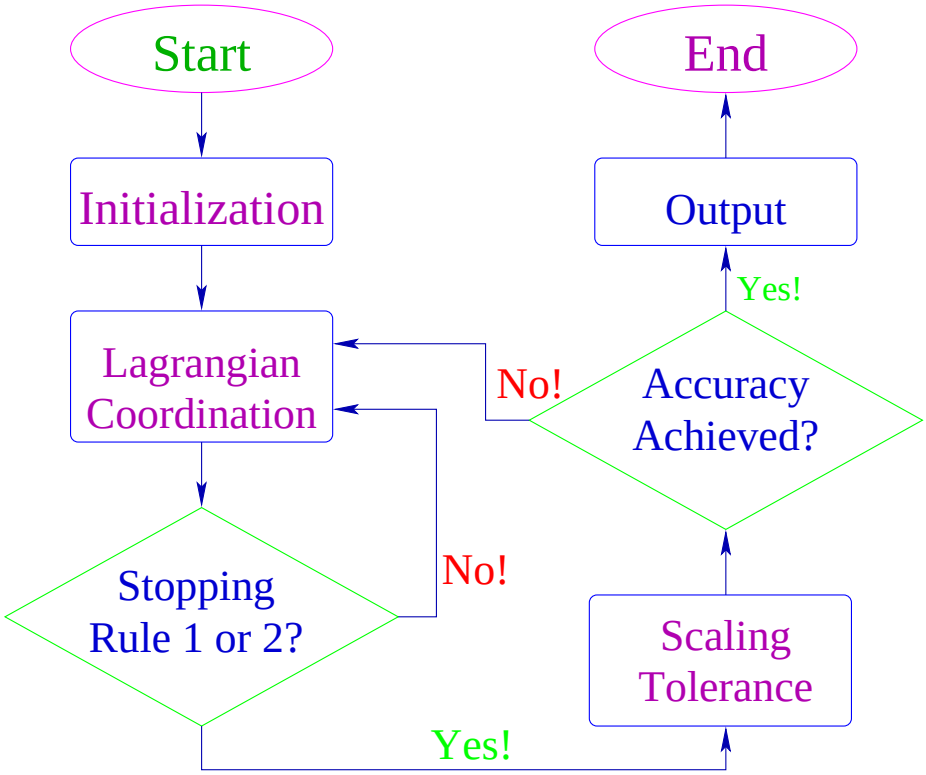


Figure 2.2: Flowchart of Algorithm $\mathcal{L}$.

is computed as the new solution. The iteration stops when the solution satisfies any one of the two stopping rules. After one scaling phase, the error tolerance σ_s is halved and the next scaling phase starts until the error tolerance $\sigma_s \leq \varepsilon$ (Figure 2.2). The solution generated in the last scaling phase solves the primal problem $(P_{\varepsilon,c})$ (see also Subsection 2.3.1).

In our algorithm we set $t = \sigma_s/6$ for the error tolerance in the block solver $ABS(p, t, c)$. To run the algorithm, we need an initial solution $x_0 \in B$ in advance. Here we use as x_0 the solution of the block solver $ABS(e/M, \sigma_0/6, c)$ where the price vector e/M is the vector of all $1/M$'s and the initial error tolerance $\sigma_0 = 1$.

(2.3).

We set the step length τ as:

$$\tau = \frac{t\theta\nu}{2M(p^T f + p^T \hat{f})}. \tag{2.8}$$

We note that the step length τ can be computed also by a line search to minimize the potential value ϕ_t. The algorithm and the analysis remains valid if we use such

Algorithm $\mathcal{L}(f, B, \varepsilon, c)$:
initialize: $s := 0$, $\sigma_0 := 1$, $t := \sigma_0/6$, $p := e/M$, $x_0 := ABS(p, t, c)$;
$finished\_scaling := false$;
while $not(finished\_scaling)$ **do** {*scaling phase*}
  $s := s + 1$, $x := x_{s-1}$, $\sigma_s := \sigma_{s-1}/2$, $t := \sigma_s/6$;
  $finished\_coordination := false$;
  compute w_s from (2.6);
  while $not(finished\_coordination)$ **do** {*coordination step*}
    compute $\theta(x)$ from (2.3) and $p = p(x) \in P$ from (2.4);
    $\hat{x} := ABS(p, t, c)$;
    compute $\nu = \nu(x, \hat{x})$ from (2.5);
    if (*Stopping Rule 1 or 2*) **then**
      $x_s := x$, $finished\_coordination := true$;
    else
      $x := (1 - \tau)x + \tau\hat{x}$ for an appropriate step length $\tau \in (0, 1]$;
    end
  end
  if $(\sigma_s \leq \varepsilon)$ **then** $finished\_scaling := true$;
end

Table 2.1: Algorithm $\mathcal{L}$ for the Convex Min-Max Resource-Sharing problem.

a line search to compute τ (see the similar analysis in [45]).

2.3.1 Analysis of the algorithm $\mathcal{L}$

In this section we verify the convergence of algorithm $\mathcal{L}$ by proving that the algorithm stops in each scaling phase after a finite number of iterations. Furthermore we show that the vector x computed in the final phase solves the primal problem $(P_{\varepsilon,c})$. From now on for convenience we denote $\theta = \theta(x)$, $\theta' = \theta(x')$, $f = f(x)$, $f' = f(x')$ and $\hat{f} = f(\hat{x})$. Before proving the correctness of the approximation algorithm we have the following bound for the initial solution x_0.

Lemma 2.7 *If x_0 is the solution of $ABS(e/M, t, c)$ with $t = 1/6$, then $\lambda(x_0) \leq (7/6)cM\lambda^*$.*

Proof: Since x_0 is the solution of the block solver $ABS(e/M, t, c)$ we have

$$\sum_{m=1}^{M} f_m(x_0)/M \;=\; e^T f(x_0)/M \;\leq\; (1+t)c\Lambda(e/M) \;=\; (7/6)c\Lambda(e/M).$$

Using the fact $\lambda^* = \max_{p\in P} \Lambda(p)$, the right side of the inequality above is bounded by $(7/6)c\lambda^*$. Therefore $f_m(x_0) \leq (7/6)cM\lambda^*$ for each $m = 1, \ldots, M$. This implies $\lambda(x_0) = \max_{1\leq m\leq M} f_m(x_0) \leq (7/6)cM\lambda^*$. □

Lemma 2.8 *If algorithm $\mathcal{L}$ stops, then the computed $x \in B$ solves $(P_{\varepsilon,c})$.*

Proof: We shall consider both stopping rules. If the first stopping rule is satisfied, then using the definition of ν and $\nu \leq t$ we have

$$(1-t)p^T f \leq (1+t)p^T \hat{f}.$$

According to Lemma 2.6, the inequality above, and the value $p^T f(\hat{x})$ of the solution $\hat{x}$ computed by the block solver $ABS(p, t, c)$, we have

$$\theta = \frac{p^T f}{1-t} \leq \frac{1+t}{(1-t)^2} p^T \hat{f} \leq c\left(\frac{1+t}{1-t}\right)^2 \Lambda(p).$$

Using the fact that $\Lambda(p) \leq \lambda^*$ and $t = \sigma_s/6$, we obtain $\theta \leq c(1+\sigma_s)\lambda^*$. Then by Lemma 2.4, $f_m(x_s) \leq \lambda(x_s) \leq \theta(x_s) \leq c(1+\sigma_s)\lambda^*$.

If the second stopping rule is satisfied, then we consider two further cases. If the rule is satisfied in the first phase $s = 1$, then according to the value of w in formula (2.6),

$$\lambda(x_1) \leq \frac{1+\sigma_1}{(1+\sigma_1/3)M} \lambda(x_0).$$

Since x_0 is the solution of $ABS(e/M, t, c)$ we have $\lambda(x_0) \leq c(1+\sigma_0/6)M\lambda^*$. This implies that $\lambda(x_1) \leq c(1+\sigma_1)\lambda^*$.

Now let us assume (by induction) that $\lambda(x_{s-1}) \leq c(1+\sigma_{s-1})\lambda^*$ after $s-1$st scaling phase. Then if the second stopping rule is satisfied in phase s, then according to the definition of w and $\sigma_s = \sigma_{s-1}/2$, we have

$$\lambda(x_s) \leq \frac{(1+\sigma_s)}{(1+2\sigma_s)}\lambda(x_{s-1}) \leq (1+\sigma_s)c\lambda^*.$$

Therefore in both cases we have $\lambda(x_s) \leq (1+\sigma_s)c\lambda^*$ for any $s > 0$. Since $\sigma \leq \varepsilon$ when the algorithm $\mathcal{L}$ halts, the solution $x = x_s$ computed in the last phase solves $(P_{\varepsilon,c})$. □

In the next lemma we show that the reduced potential function ϕ_t decreases boundedly by a constant factor in each coordination step. This enables us later to prove an upper bound for the number of iterations.

Lemma 2.9 *For any two consecutive iterates x, $x' \in B$ within a scaling phase of algorithm $\mathcal{L}$,*

$$\phi_t(x') \leq \phi_t(x) - \frac{t\nu^2}{4M}.$$

Proof: Due to the convexity of the function f_m and the definition of the price vector p_m in (2.4), we get

$$\begin{aligned} \theta - f'_m &\geq \theta - (1-\tau)f_m - \tau\hat{f}_m \\ &= (\theta - f_m)\left(1 + \tau\frac{f_m - \hat{f}_m}{\theta - f_m}\right) \\ &= (\theta - f_m)\left(1 + \frac{\tau M}{t\theta}p_m(f_m - \hat{f}_m)\right). \end{aligned} \tag{2.9}$$

By the non-negativity of f_m and the definition of τ in (2.8),

$$\left|\frac{\tau M}{t\theta}p_m(f_m - \hat{f}_m)\right| \leq \frac{\tau M}{t\theta}p_m(f_m + \hat{f}_m) \leq \frac{\tau M}{t\theta}(p^T f + p^T \hat{f}) = \frac{\nu}{2} \leq \frac{1}{2},$$

where the fact that $\nu \leq 1$ is used. Since $\theta - f_m$ is positive by Lemma 2.4 we have $f'_m < \theta$ for $m = 1, \ldots M$, i.e., $\lambda(x') < \theta$. From the definition $\phi_t(y) = \Phi_t(\theta(y), f(y))$,

$$\phi_t(x') = \Phi_t(\theta', f') = \min_{\xi \geq \lambda(x')} \Phi_t(\xi, f') \leq \Phi_t(\theta, f') = \ln\theta - \frac{t}{M}\sum_{m=1}^{M}\ln(\theta - f'_m), \tag{2.10}$$

for the particular point $x' \in B$. By substituting (2.9) in (2.10) and using the definition (2.1) of the potential function,

$$\begin{aligned}\phi_t(x') &\leq \ln\theta - \frac{t}{M}\sum_{m=1}^{M}\ln(\theta - f_m) - \frac{t}{M}\sum_{m=1}^{M}\ln\left(1+\tau\frac{f_m-\hat{f}_m}{\theta-f_m}\right)\\ &= \phi_t(x) - \frac{t}{M}\sum_{m=1}^{M}\ln\left(1+\tau\frac{f_m-\hat{f}_m}{\theta-f_m}\right), \qquad (2.11)\end{aligned}$$

where the last term is well defined. Now we use the second inequality in Lemma 2.3:

$$\begin{aligned}\ln\left(1+\tau\frac{f_m-\hat{f}_m}{\theta-f_m}\right) &= \ln\left(1+\frac{M\tau}{t\theta}p_m(f_m-\hat{f}_m)\right)\\ &\geq \frac{M\tau}{t\theta}p_m(f_m-\hat{f}_m) - \left(\frac{M\tau}{t\theta}p_m(f_m-\hat{f}_m)\right)^2,\end{aligned}$$

for all $m = 1, \ldots, M$. Hence, (2.11) becomes:

$$\begin{aligned}\phi_t(x') &\leq \phi_t(x) - \tau\frac{p^T(f-\hat{f})}{\theta} + \frac{M\tau^2}{t\theta^2}\sum_{m=1}^{M}(p_m(f_m-\hat{f}_m))^2\\ &\leq \phi_t(x) - \frac{\tau\nu}{\theta}(p^Tf + p^T\hat{f}) + \frac{M\tau^2}{t\theta^2}(p^Tf+p^T\hat{f})^2\\ &= \phi_t(x) - \frac{\tau\nu}{\theta}(p^Tf + p^T\hat{f}) + \frac{\tau\nu}{2\theta}(p^Tf+p^T\hat{f})\\ &= \phi_t(x) - \frac{\tau\nu}{2\theta}(p^Tf + p^T\hat{f})\\ &= \phi_t(x) - \frac{t\nu^2}{4M},\end{aligned}$$

where the definitions of ν and τ are employed. □

Now we are ready to estimate the complexity of algorithm $\mathcal{L}$.

Theorem 2.1 *For a given relative accuracy $\varepsilon \in (0, 1]$, algorithm $\mathcal{L}$ finishes with a solution x that satisfies $\lambda(x) \leq c(1+\varepsilon)\lambda^*$ and performs a total of*

$$N = O(M(\ln M + \varepsilon^{-2}\ln\varepsilon^{-1}))$$

coordination steps.

Proof: If algorithm $\mathcal{L}$ finishes after a finite number of iterations with a solution x, then using Lemma 2.8 $\lambda(x) \leq c(1+\varepsilon)\lambda^*$. First we calculate a bound on the number N_s of coordination steps performed in sth scaling phase. Afterwards we obtain a bound for all scaling phases. This shows also that algorithm $\mathcal{L}$ halts and proves the theorem.

Let x, x' denote the initial and final iterates of sth scaling phase. In addition let $\bar{x}$ be the solution after $\overline{N}_s = N_s - 1$ iterations in the same scaling phase. During the phase from x to $\bar{x}$ we have $\nu > t$; otherwise the phase would finish earlier. Furthermore (by the same reason) the objective value $\lambda(\bar{x}) > w\lambda(x)$. Then Lemma 2.9 provides

$$\phi_t(x) - \phi_t(\bar{x}) \geq \overline{N}_s \frac{t\nu^2}{4M} \geq \overline{N}_s \frac{t^3}{4M}.$$

By the left and right inequality of Lemma 2.5 we have $(1-t)\ln\lambda(\bar{x}) \leq \phi_t(\bar{x})$ and $\phi_t(x) \leq (1-t)\ln\lambda(x) + t\ln(\exp(1)/t)$, respectively. Hence,

$$\overline{N}_s \frac{t^3}{4M} \leq (1-t)\ln\frac{\lambda(x)}{\lambda(\bar{x})} + t\ln\frac{\exp(1)}{t}.$$

This gives directly a bound for N_s,

$$N_s = \overline{N}_s + 1 \leq 4Mt^{-3}\left((1-t)\ln\frac{\lambda(x)}{\lambda(\bar{x})} + t\ln\frac{\exp(1)}{t}\right) + 1.$$

Next we shall bound the term $\lambda(x)/\lambda(\bar{x})$. For the first scaling phase we have $\lambda(\bar{x}) > \frac{1+\sigma_1}{(1+\sigma_1/3)M}\lambda(x)$. In this case

$$\frac{\lambda(x)}{\lambda(\bar{x})} \leq \frac{(1+\sigma_1/3)M}{1+\sigma_1} < \frac{2M}{1+\sigma_1}.$$

Since $\sigma_1 = 1/2$ and $t = 1/12$ during the first scaling phase, there are only $O(M\ln M)$ coordination steps in the first phase. For the other scaling phases, we have $\lambda(\bar{x}) > \frac{1+\sigma_s}{1+2\sigma_s}\lambda(x)$. Therefore,

$$\frac{\lambda(x)}{\lambda(\bar{x})} \leq \frac{1+2\sigma_s}{1+\sigma_s} = 1 + \frac{\sigma_s}{1+\sigma_s}.$$

Then according to the elementary inequality $\ln(1+u) \leq u$ for any $u \geq 0$,

$$\ln \frac{\lambda(x)}{\lambda(\bar{x})} \leq \ln(1 + \frac{\sigma_s}{1+\sigma_s}) \leq \frac{\sigma_s}{1+\sigma_s} < \sigma_s.$$

Substituting this in the bound for N_s above and using $t = \sigma_s/6$, we have the expression

$$N_s = O(M\sigma_s^{-2} \ln \sigma_s^{-1}).$$

Finally, the total number of coordination steps N is obtained by summing the N_s over all scaling phases:

$$N = O(M(\ln M + \sum_s \sigma_s^{-2} \ln \sigma_s^{-1})).$$

Since $\sigma_{s+1} = \sigma_s/2$ in the $(s+1)$st scaling phases, the sum above is further bounded by

$$\begin{aligned} \sum_s \sigma_s^{-2} \ln \sigma_s^{-1} &= O(\sum_{q=0}^{\lceil \log \varepsilon^{-1} \rceil} 2^{2q} \ln(2^q)) \\ &= O(\log \varepsilon^{-1} \sum_{q=0}^{\lceil \log \varepsilon^{-1} \rceil} 2^{2q}) \\ &= O(\varepsilon^{-2} \log \varepsilon^{-1}), \end{aligned}$$

which provides the claimed bound. □

2.3.2 Faster algorithm for small approximation ratio c

In this section we analyze the case that the approximation ratio c of the block solver is small enough, i.e., $\ln c = O(\varepsilon)$. In this case we can propose another algorithm $\mathcal{L}'$. In the algorithm $\mathcal{L}'$, we only use the parameter ν in (2.5) to design the stopping rule. Then the stopping rule is as follows:

$$\nu \leq \sigma_s/6. \tag{2.12}$$

This stopping rule is similar to that in [109]. By this means the algorithm $\mathcal{L}'$ is as in Table 2.2. Here the step length τ is set same as in (2.8).

Algorithm $\mathcal{L}'(f, B, \varepsilon, c)$:
initialize: $s := 0$, $\sigma_0 := 1$, $t := \sigma_0/6$, $p := e/M$, $x_0 := ABS(p, t, c)$;
$finished\_scaling := false$;
while $not(finished\_scaling)$ **do** {*scaling phase*}
  $s := s + 1$, $x := x_{s-1}$, $\sigma_s := \sigma_{s-1}/2$, $t := \sigma_s/6$;
  $finished\_coordination := false$;
  compute w_s from (2.6);
  while $not(finished\_coordination)$ **do** {*coordination step*}
    compute $\theta(x)$ from (2.3) and $p = p(x) \in P$ from (2.4);
    $\hat{x} := ABS(p, t, c)$;
    compute $\nu = \nu(x, \hat{x})$ from (2.5);
    if (*Stopping Rule* (2.12)) **then**
      $x_s := x$, $finished\_coordination := true$;
    else
      $x := (1 - \tau)x + \tau\hat{x}$ for an appropriate step length $\tau \in (0, 1]$;
    end
  end
  if $(\sigma_s \leq \varepsilon)$ **then** $finished\_scaling := true$;
end

Table 2.2: Algorithm $\mathcal{L}'$ for the Convex Min-Max Resource-Sharing problem with small ratio.

2.3.3 Analysis of the algorithm $\mathcal{L}'$

First as for the correctness we have the following lemma:

Lemma 2.10 *If algorithm $\mathcal{L}'$ stops, then the computed pair $x \in B$ and $p \in P$ solves $(P_{\varepsilon,c})$ and $(D_{\varepsilon,c})$.*

The proof is very similar to the first part of that of Lemma 2.8, with only the first stopping rule employed. Furthermore we prove an upper bound on the difference of the reduced potential function for any two arbitrary points x and x' in B. The proof is similar to one in [48] for the general covering problem.

Lemma 2.11 *For any two iterates x, $x' \in B$ within a scaling phase with $\lambda(x) > 0$ and $\lambda(x') > 0$,*

$$\phi_t(x) - \phi_t(x') \le (1-t)\ln\frac{p^T f}{p^T f'} \le (1-t)\ln\frac{p^T f}{\Lambda(p)},$$

where $p = p(x)$ is defined by (2.4).

Proof: Denote $\Lambda = \Lambda(p) = \min_{x\in B} p^T f(x)$. Using the definition of price vector $p = p(x)$ in (2.4) and Lemma 2.6,

$$\begin{aligned}
\phi_t(x) - \phi_t(x') &= \ln\frac{\theta}{\theta'} - \frac{t}{M}\sum_{m=1}^{M}\ln(\frac{\theta - f_m}{\theta' - f'_m}) \\
&= \ln\frac{\theta}{\theta'} + \frac{t}{M}\sum_{m=1}^{M}\ln(\frac{\theta' - f'_m}{\theta - f_m}) \\
&= \ln\frac{\theta}{\theta'} + \frac{t}{M}\sum_{m=1}^{M}\ln(\frac{M}{t\theta}p_m(\theta' - f'_m)) \\
&= \ln\frac{\theta}{\theta'} + t\ln\frac{M}{t\theta} + \frac{t}{M}\sum_{m=1}^{M}\ln(p_m(\theta' - f'_m)) \\
&\le \ln\frac{\theta}{\theta'} + t\ln\frac{M}{t\theta} + t\ln(\frac{1}{M}p^T(\theta' e - f')) \\
&= \ln\frac{\theta}{\theta'} + t\ln\frac{1}{t\theta} + t\ln(\theta' - p^T f') \\
&\le \max_{\xi>\Lambda}\{\ln\frac{\theta}{\xi} + t\ln\frac{1}{t\theta} + t\ln(\xi - p^T f')\} \\
&= (1-t)\ln\frac{(1-t)\theta}{p^T f'} \\
&= (1-t)\ln\frac{p^T f}{p^T f'} \\
&\le (1-t)\ln\frac{p^T f}{\Lambda(p)},
\end{aligned}$$

where the concavity of $\ln(\cdot)$ is used to obtain the first inequality above. □

For the complexity of the algorithm $\mathcal{L}'$ we have the following theorem.

Theorem 2.2 *In the case of $\ln c = O(\varepsilon)$, algorithm $\mathcal{L}'$ performs a total of*

$$N = O(M(\ln M + \varepsilon^{-2}))$$

coordination steps.

Proof: We still consider here the cases of the first and the other scaling phase separately. Let us use the notation in Theorem 2.1. For each scaling phase (similar to the proof of Theorem 2.1) we have

$$\phi_t(x) - \phi_t(x') \geq N_s \frac{t^3}{4M}.$$

Then according to Lemma 2.5 we have $(1-t)\ln\lambda^* \leq (1-t)\ln\lambda(x') \leq \phi_t(x')$ and $\phi_t(x) \leq (1-t)\ln\lambda(x) + t\ln(\exp(1)/t)$. Substituting them into the above inequality we get

$$N_s \frac{t^3}{4M} \leq (1-t)\ln\frac{\lambda(x)}{\lambda^*} + t\ln\frac{\exp(1)}{t}.$$

Now consider the first scaling phase. Since $\lambda(x) \leq 2cM\lambda^*$ and $t = 1/12$, the number of iterations for the first phase is $N_1 = O(M\ln(cM)) = O(M(\ln c + \ln M))$. Then the assumption $\ln c = O(\varepsilon)$ yields $N_1 = O(M\ln M)$. For the other scaling phases (using Lemma 2.11) we have

$$\phi_t(x) - \phi_t(x') \leq (1-t)\ln\frac{p^T f}{\Lambda(p)}$$

where p is the price vector corresponding to x. In the $(s-1)$st scaling phase $t' = 2t$ and the stopping rule was satisfied for x, i.e., $\nu(x,\hat{x}) \leq t' = 2t$. By the definition of ν we have $(1-\nu)p^T f = (1+\nu)p^T\hat{f}$. Since $\hat{x}$ is the solution to the block problem, it follows that $p^T\hat{f} \leq (1+2t)c\Lambda(p)$. By $\nu \leq 2t$ and $t \leq 1/12$,

$$\frac{p^T f}{\Lambda(p)} \leq \frac{(1+\nu)}{(1-\nu)}(1+2t)c \leq \frac{1+2t}{1-2t}(1+2t)c \leq (1+10t)c.$$

Combining both inequalities and using the inequality $\ln(1+u) \leq u$,

$$\phi_t(x) - \phi_t(x') \leq (1-t)(\ln(1+10t) + \ln c) \leq 10t + \ln c.$$

Now we combine this inequality with the upper bound for N_s and get:

$$N_s \leq \frac{4M}{t^3}(10t + \ln c).$$

Using the assumption that $\ln c = O(\varepsilon)$, $t = \sigma_s/6$ and $\varepsilon \leq 2\sigma_s$ for each scaling phase we obtain $N_s = O(M\sigma_s^{-2} + M\varepsilon\sigma_s^{-3}) = O(M\sigma_s^{-2})$. In addition, we have

$$\sum_s \sigma_s^{-2} = O(\sum_{q=0}^{\lceil \log \varepsilon^{-1} \rceil} 2^{2q}) = O(\varepsilon^{-2}).$$

Then summing N_s over all scaling phase, the total number N of coordination steps (similar to the proof of Theorem 2.1) is bounded by $O(M(\ln M + \varepsilon^{-2}))$. □

2.4 Further analyses

We have already proposed an algorithm $\mathcal{L}$ for solving the primal problem $(P_{\varepsilon,c})$ in at most $O(M(\ln M + \varepsilon^{-2}\ln\varepsilon^{-1})))$ iterations. In addition, we also addressed a faster algorithm $\mathcal{L}'$ for both primal problem and dual problem for a special case that $\ln c \leq O(\varepsilon)$. In this section, we will study more details of the algorithms and show the bound on the number of iterations by using our algorithms to solve the dual problem. In addition, we also study the influence of numerical error such that the solutions delivered by our algorithms fulfil the desired accuracy requirement.

2.4.1 Analysis of the dual problem

In this subsection we will consider the dual problem. As we mentioned before, in the general case we can obtain the solution to the primal problem in $O(M(\ln M + \varepsilon^{-2}\ln\varepsilon^{-1}))$ iterations. Now we analyze the dual value of the solution of our algorithms. For the general case the following theorem holds:

Theorem 2.3 *The number of iterations to solve the dual problem $(D_{\varepsilon,c})$ by $\mathcal{L}'$ is at most*

$$\begin{cases} O(M(\ln M + \varepsilon^{-2})), & \textit{if } \ln c \le O(\varepsilon + \varepsilon^3 \ln M) \textit{or} c \ge 12M/\varepsilon; \\ O(M\varepsilon^{-2} \ln M), & \textit{if } O(\varepsilon + \varepsilon^3 \ln M) \le \ln c \le O(\varepsilon \ln M); \\ O(M\varepsilon^{-3} \ln M), & \textit{if } O(\varepsilon \ln M) \le \ln c \le O(\ln M); \\ O(M\varepsilon^{-3} \ln(M\varepsilon^{-1})), & \textit{if } O(M) \le c \le 12M/\varepsilon. \end{cases}$$

Proof: According to the proof of Theorem 2.2, for general $c \ge 1$, to solve both the primal problem and the dual problem, the number of iterations in the first phase $s = 1$ is at most $N_1 = O(M(\ln M + \ln c))$. Furthermore, the number of iterations in the s-th phase is bounded by $N_s = O(M(\sigma_s^{-2} + \sigma_s^{-3} \ln c))$. Since

$$\begin{aligned} \sum_s \sigma_s^{-2} &= O(\sum_{q=0}^{\lceil \log \varepsilon^{-1} \rceil} 2^{2q}) = O(\varepsilon^{-2}), \\ \sum_s \sigma_s^{-3} &= O(\sum_{q=0}^{\lceil \log \varepsilon^{-1} \rceil} 2^{3q}) = O(\varepsilon^{-3}). \end{aligned}$$

Thus the total number of iterations to solve both problems is

$$N = O(M(\ln M + \sum_s (\sigma_s^{-2} + \sigma_s^{-3} \ln c))) = O(M(\ln M + \varepsilon^{-2} + \varepsilon^{-3} \ln c)). \quad (2.13)$$

We need to consider various cases depending on the relations between c, M and ε.

In the case $\ln c \le O(\varepsilon + \varepsilon^3 \ln M)$, as showed in (2.13), algorithm $\mathcal{L}'$ can solve both problems in $O(M(\ln M + \varepsilon^{-2}))$ iterations.

In the case $O(\varepsilon + \varepsilon^3 \ln M) \le \ln c \le O(\varepsilon \ln M)$, substituting the bound on $\ln c$ to (2.13) gives the upper bound $O(M\varepsilon^{-2} \ln M)$ on the number of iterations of algorithm $\mathcal{L}'$ to obtain the dual solution.

In the case $O(\varepsilon \ln M) \le \ln c \le O(\ln M)$, in $O(M\varepsilon^{-3} \ln M)$ iterations the dual solution can be attained with the same approach.

In the case $O(M) \le c \le 12M/\varepsilon$, the number of iterations can be bounded by $O(M\varepsilon^{-3} \ln(M\varepsilon^{-1}))$.

In the case $c \ge 12M/\varepsilon$, the pair x and p obtained by any of our algorithms is also the solution to both the primal and the dual problem, and the number of iteration

is also $O(M(\ln M + \varepsilon^{-2}))$. To prove this, using the definition of price vector and the bounds for $\theta(x)$,

$$p_m(x) = \frac{t}{M}\frac{\theta(x)}{\theta(x) - f_m(x)} \geq \frac{t}{M}\frac{\frac{\lambda(x)}{1-t/M}}{\frac{\lambda(x)}{1-t} - f_m(x)} \geq \frac{t}{M}\frac{1-t}{1-t/M} \geq \frac{t}{M}(1-t).$$

Hence, if the algorithm stops (i. e. $\varepsilon/12 \leq t \leq \varepsilon/6$), then $p_m(x) \geq \varepsilon(1-\varepsilon)/(12M)$. This implies

$$\begin{aligned} \Lambda(p) &= \min_x p^T f(x) \geq \min_x \frac{\varepsilon}{12M}(1-\varepsilon)\sum_{m=1}^{M} f_m(x) \\ &\geq \frac{\varepsilon}{12M}(1-\varepsilon)\min_x \lambda(x) \geq \frac{1}{c}(1-\varepsilon)\lambda^*. \end{aligned}$$

The analysis shows that the solution to the dual problem is easier to obtain than that to the primal problem for large c.

Therefore, the overall bound for the number of iterations to obtain the solution to the dual problem is $O(M\varepsilon^{-3}\ln(M\varepsilon^{-1}))$. □

2.4.2 Analysis of accuracy

In the analysis above we assumed that the price vector $p \in P$ can be computed exactly from (2.4). However, in practice this is not true. The reason is that to compute the price vector, the value of $\theta(x)$ is needed in (2.4). However, $\theta(x)$ is the root of (2.3), which only can be computed approximately by numerical methods in general. Thus we need to study the influence of the numerical error to the accuracy of the delivered solution by our algorithms. The goal is to control the numerical error so that the solution of our algorithms is still a $c(1 + O(\varepsilon))$-approximation of the optimal solution. A similar argument below how to resolve this problem can be found in [45, 46, 48, 109].

Theorem 2.4 *Solving the equation (2.3) with an absolute error of $O(\varepsilon^2/M)$ can obtain a $c(1 + O(\varepsilon))$-approximate solution to the primal problem.*

Proof: Suppose $\tilde{p}$ is an approximation of the exact value of p with the relative accuracy $\delta \in (0, 1/2)$. Then $(1-\delta)p \leq \tilde{p} \leq (1+\delta)p$. For this fixed p, let $z = \arg\min_{x \in B} p^T f(x)$, i.e. $p^T f(z) = \min_{x \in B} p^T f(x) = \Lambda(p)$ (z is a vector corresponding to the minimum dual value $\Lambda(p)$). Then

$$\Lambda(\tilde{p}) = \min_{x \in B} \tilde{p}^T f(x) \leq \tilde{p}^T f(z) \leq (1+\delta) p^T f(z) = (1+\delta)\Lambda(p).$$

Let $\hat{x}$ be the solution delivered by $ABS(\tilde{p}, t, c)$. Then we get the following bound:

$$\begin{aligned} p^T f(\hat{x}) &\leq \frac{\tilde{p}^T f(\hat{x})}{1-\delta} \leq \frac{c(1+t)\Lambda(\tilde{p})}{1-\delta} \\ &\leq \frac{1+\delta}{1-\delta} c(1+t)\Lambda(p) \leq c\ (1+t)(1+4\delta)\Lambda(p) \end{aligned}$$

for any $\delta \leq 1/2$. In other words, the solution $\hat{x}$ from $ABS(\tilde{p}, t, c)$ has a relative accuracy of $O(t+\delta) = O(\sigma) \to O(\varepsilon)$ to the exact price vector p for $\delta = O(t) \to O(\varepsilon)$. This means that we just need to compute every p_m to the relative accuracy of $\delta = O(\varepsilon)$ in (2.4).

Next we calculate the error bound for θ in solving (2.3) such that $\tilde{p}$ can achieve the required accuracy. It can be verified that $\theta(sf) = s\theta(f)$ and $p(sf) = p(f)$ for any positive scalar s. In this way, it is possible that the vector f is pre-scaled locally just for the computation of $\theta(f)$ so that $\lambda(f) = 1$. According to Lemma 2.4, $\theta(f) \in [M/(M-t), 1/(1-t)]$ and therefore $|\theta(f) - 1| = O(t)$. Let $\tilde{\theta}(f)$ be the approximation of θ to compute $\tilde{p}$. Our goal is to calculate the absolute error bound Δ between $\tilde{\theta}(f)$ and $\theta(f)$ such that the approximate price vector $\tilde{p}$ has the relative accuracy δ. From (2.4), $\tilde{p}_m = t\tilde{\theta}(f)/(M(\tilde{\theta}(f) - f_m))$. Thus according to the definition of relative error of p_m we have the follows:

$$\left| \frac{\tilde{p}_m - p_m}{p_m} \right| = \left| \frac{\tilde{\theta}(f)}{\theta(f)} \frac{\theta(f) - f_m}{\tilde{\theta}(f) - f_m} - 1 \right| = O(t). \tag{2.14}$$

According to the definition of Δ, $|\tilde{\theta}(f) - \theta(f)| \leq \Delta$. Substituting this into (2.14) we need the following two upper bounds:

$$\begin{aligned}\frac{\theta(f)+\Delta}{\theta(f)}\frac{\theta(f)-f_m}{\theta(f)-\Delta-f_m}-1 &= O(t);\\ 1-\frac{\theta(f)-\Delta}{\theta(f)}\frac{\theta(f)-f_m}{\theta(f)+\Delta-f_m} &= O(t).\end{aligned}$$

Let us consider the first expression. It can be written as

$$\frac{(2\theta(f)-f_m)\Delta}{\theta(f)(\theta(f)-\Delta-f_m)}=O(t).$$

Since $2\theta(f)-f_m \le 2\theta(f)=2+O(t)$, $\theta(f)-\Delta-f_m \ge M/(M-t)-1-\Delta = O(t/M)-\Delta$ and $\theta(f)\ge 1$, we need that

$$(2+O(t))\Delta/(O(t/M)-\Delta)=O(t).$$

Therefore a sufficient condition to the first bound is: $\Delta = O(t^2/M) \to O(\varepsilon^2/M)$. The second upper bound can be simplified as

$$\frac{(2\theta(f)-f_m)\Delta}{\theta(f)(\theta(f)+\Delta-f_m)}=O(t).$$

With the same bounds for the $2\theta(f)-f_m$ and $\theta(f)$, together with $\theta(f)+\Delta-f_m = O(t/M)+\Delta$, we have the same inequality $(2+O(t))\Delta/(O(t/M)-\Delta)=O(t)$. Thus the sufficient condition for the second bound is also $\Delta = O(t^2/M) \to O(\varepsilon^2/M)$. The proof is completed. □

To compute the value of $\theta(f) \in [M/(M-t), 1/(1-t)]$ approximately, we can use binary search. Since the length of the interval is $O(t) \to O(\varepsilon)$, $O(\ln(M\varepsilon^{-1}))$ steps are necessary to achieve the accuracy. This requires $O(\ln(M\varepsilon^{-1}))$ computations of the sum $\sum_{m=1}^{M} 1/(\theta(f)-f_m)$ per coordination step of algorithm $\mathcal{L}$. Therefore $O(M\ln(M\varepsilon^{-1}))$ arithmetic operations per iteration or $O(\ln M \ln(M\varepsilon^{-1}))$ on a parallel machine with $M/\ln M$ processors are necessary. On the other hand, one can also use Newton's method to compute the root $\theta(f)$ approximately. Since Newton's method is quadratically convergent [12], $O(M\ln\ln(M\varepsilon^{-1}))$ operations are necessary or $O(\ln M \ln\ln(M\varepsilon^{-1}))$ on a parallel machine. In this way the sequential running time of each coordination step is roughly linear in M.

2.5 Improvement for slow block solvers

We notice that the running time of the whole algorithm depends not only on the coordination complexity (the number of iterations) but also the running time of the block solver. Therefore we need also consider the case that a good block solver is unavailable. A possible case is that the block problem has only an algorithm with a running time depending on input value power to a function of ε^{-1} required, for instance, $O(n^{1/\varepsilon})$. In this case, to reduce the overall running time of our algorithm, one useful approach is to design a $c(1+\varepsilon)$-approximation algorithm for the min-max resource sharing problem with an approximate block solver $ABS(p, t', c)$, where $t' \geq t$. In Section 2.3 we have already addressed a $c(1+\varepsilon)$-approximation algorithm $\mathcal{L}$ with a $c(1+\varepsilon/6)$-approximate block solver. In this section we will develop some $c(1+\varepsilon)$-approximation algorithms $\mathcal{F}$ for the original min-max resource sharing problem with only an approximate block solver $ABS(p, \varepsilon_1/6, c)$, where $\varepsilon_1 = (43 - \sqrt{1849 - 1176\varepsilon})/12 \geq 49\varepsilon/43 > \varepsilon$ for any positive ε. The number of iterations is then bounded by $O(M(\ln M + \varepsilon^{-4} \ln \varepsilon^{-1}))$. This bound is further improved to $O(M(\ln M + \varepsilon^{-2} \ln \varepsilon^{-1}))$ in another algorithm $\mathcal{F}'$ with an $ABS(p, \varepsilon_4/6, c)$ for a $\varepsilon_2 \in (\varepsilon, \varepsilon_1)$.

2.5.1 Fast approximation algorithm $\mathcal{F}$ for $(P_{\varepsilon,c})$

In this section, based on the algorithm $\mathcal{L}$, we will propose a fast approximation algorithm $\mathcal{F}$ only for $(P_{\varepsilon,c})$ with $ABS(p, \varepsilon_3/6, c)$, where $\varepsilon_3 = (43 - \sqrt{1849 - 1176\varepsilon})/12 \geq 49\varepsilon/43$.

The algorithm works similarly to $\mathcal{L}$. The scaling phase strategy is employed, too. In each scaling phase a relative error tolerance σ_s is set. Lagrangian decomposition method is applied same as before. We have two stopping rules here and the iterative procedure in one scaling phase stops if any one of them is fulfilled. Then the error tolerance σ_s is halved and the new scaling phase starts in the same way as described above, until the error tolerance $\sigma_s \leq \varepsilon$. The solution x_s delivered in the last scaling phase solves $(P_{\varepsilon,c})$.

We also estimate the duality gap to construct the stop rule. For our first stopping

rule a parameter ν is defined as follows (same as [109] and Section 2.3):

$$\nu = \nu(x, \hat{x}) = \frac{p^T f(x) - p^T f(\hat{x})}{p^T f(x) + p^T f(\hat{x})}. \tag{2.15}$$

If $\nu = O(\varepsilon)$, then the duality gap is small. However, in the case that ν is large and close to 1, the gap may be extremely large (See Section 2.3). To obtain a better bound on the number of iterations, we define another parameter to connect the function value with the solution of previous scaling phase. Let σ_s be the relative error tolerance of the s-th scaling phase. Then similar to Algorithm $\mathcal{L}$, the parameter w_s is defined as follows:

$$w_s = \begin{cases} \dfrac{1+\sigma_1}{(1+\sigma_0/6)M}, & \text{for the first scaling phase;} \\ \dfrac{1+\sigma_s}{1+2\sigma_s}, & \text{otherwise.} \end{cases} \tag{2.16}$$

Let x_s be the solution of s-th scaling phase. Then the two stopping rules used in the s-th scaling phase are:

$$\begin{aligned} &\textit{Rule } 1: && \nu \le \sigma_s'^2/36; \\ &\textit{Rule } 2: && \lambda(x) \le w_s\, \lambda(x_{s-1}), \end{aligned} \tag{2.17}$$

where $\sigma' = k_\varepsilon \sigma$ and the parameter $k_\varepsilon = (55+\sqrt{1849-1176\varepsilon})/98 < 1$. The stopping rules here are similar to those in $\mathcal{L}$. But the latter are just for the case of solving the problem with an $ABS(p, \varepsilon/6, c)$.

We set $t = \sigma_s'/6$ for the error tolerance in the block solver $ABS(p, t, c)$ in our algorithm. We apply the solution of the block solver $ABS(e/M, 1/6, c)$ as initial solution x_0, where the price vector e/M is still the vector of all $1/M$'s and the initial error tolerance $\sigma_0 = 1$.

Similar to [109] and Algorithm $\mathcal{L}$, the step length is set as

$$\tau = \frac{t\theta(x)\nu}{2M(p(x)^T f(x) + p(x)^T f(\hat{x}))}. \tag{2.18}$$

Algorithm $\mathcal{F}(f, B, \varepsilon, c)$:
initialize: $s := 0$, $\sigma_1 = \sigma_0 := 1$, $\sigma_1' = \sigma_0' := k_\varepsilon \sigma_0$, $t := \sigma_0'/6$, $p := e/M$;
$x_0 := ABS(p, t, c)$, $finished\_scaling := false$;
while $not(finished\_scaling)$ **do** {*scaling phase*}
  $s := s + 1$, $x := x_{s-1}$ and $finished\_coordination := false$;
  compute w_s from (2.16);
  while $not(finished\_coordination)$ **do** {*coordination step*}
    compute $\theta(x)$ from (2.3) and $p = p(x) \in P$ from (2.4);
    $\hat{x} := ABS(p, t, c)$;
    compute $\nu = \nu(x, \hat{x})$ from (2.15);
    if (*Stopping Rule 1 or 2*) **then**
      $x_s := x$ and $finished\_coordination := true$;
    else
      $x := (1 - \tau)x + \tau\hat{x}$ for an appropriate step length $\tau \in (0, 1]$;
    end
  end
  $\sigma_{s+1} := \sigma_s/2$, $\sigma_{s+1}' := k_\varepsilon \sigma_{s+1}$ and $t := \sigma_{s+1}'/6$;
  if $(\sigma_{s+1} \leq \varepsilon/2)$ **then** $finished\_scaling := true$;
end

Table 2.3: Algorithm $\mathcal{F}$ for the Convex Min-Max Resource-Sharing problem.

2.5.2 Analysis of the algorithm $\mathcal{F}$

We are going to analyze the algorithm $\mathcal{F}$ is this section. We will show the correctness, i.e., to prove that the solution x_s of the last scaling phase is a solution to $(P_{\varepsilon,c})$. Then we will prove that the bound on the number of iterations such that the algorithm stops is polynomial only in M and ε. From now on we denote $\theta = \theta(x)$, $\theta' = \theta(x')$, $f = f(x)$, $f' = f(x')$ and $\hat{f} = f(\hat{x})$. First we can obtain the following bound on the function value of the initial solution x_0, similar to that in [109] and Section 2.3:

Lemma 2.12 *If x_0 is the solution of $ABS(e/M, t, c)$ with $t = 1/6$, then $\lambda(x_0) \leq (7/6)cM\lambda^*$.*

Now we will prove the correctness of $\mathcal{F}$ by showing that at the end of the s-th scaling phase the solution satisfies $\lambda(x) \leq c(1+\sigma_s)\lambda^*$ and at the end $(P_{\varepsilon,c})$ is solved.

Theorem 2.5 *If algorithm $\mathcal{F}$ stops, then for any $\varepsilon \in (0,1]$ the computed solution $x \in B$ fulfils $(P_{\varepsilon,c})$ with $ABS(p, \varepsilon_3/6, c)$, where $\varepsilon_3 = (43 - \sqrt{1849 - 1176\varepsilon})/12$.*

Proof: We will consider both stopping rules, and try to get bounds on $\lambda(x)$ in each scaling phase. If the first stopping rule is satisfied, then by the definition of ν we have

$$(1-t^2)p^T f \leq (1+t^2)p^T \hat{f}.$$

Thus Lemma 2.6, the above inequality, and the definition of block solver $ABS(p,t,c)$ yield

$$\begin{aligned} \theta &= \frac{p^T f}{1-t} \leq \left(\frac{1+t^2}{1-t^2}\cdot\frac{1-t}{1+t}\right)\frac{1+t}{(1-t)^2}p^T\hat{f} \\ &\leq \left(\frac{1+t^2}{1-t^2}\cdot\frac{1-t}{1+t}\right)c\left(\frac{1+t}{1-t}\right)^2\Lambda(p). \end{aligned}$$

We observe that the first factor is strictly less than 1 for any $t > 0$. In fact we have the following bound on it:

$$\begin{aligned} \frac{1+t^2}{1-t^2}\cdot\frac{1-t}{1+t} &= 1 - \frac{2t}{1+2t+t^2} \leq 1 - \frac{2t}{(1+1/6)^2} \\ &= 1 - \frac{72}{49}t = 1 - \frac{12}{49}\sigma_s'. \end{aligned}$$

The inequality comes from the largest $t = 1/6$ in the first scaling phase. In addition, using the fact that $t = \sigma_s'/6$ for each scaling phase, we have

$$\left(\frac{1+t}{1-t}\right)^2 \leq 1 + \sigma_s'.$$

This bound is almost tight with considering the value of $t = \sigma_s'/6$. Together with the fact $\Lambda(p) \leq \lambda^*$, we obtain $\theta \leq (1 - 12\sigma_s'/49)c(1+\sigma_s')\lambda^*$. Then by Lemma 2.4, $f_m(x_s) \leq \lambda(x_s) \leq \theta(x_s) \leq (1-12\sigma_s'/49)c(1+\sigma_s')\lambda^* \leq c(1+\sigma_s)\lambda^*$.

If the second stopping rule is satisfied, then we consider two further cases. If the rule is satisfied in the first phase $s = 1$, then according to the value of w_s in formula (2.16),

$$\lambda(x_1) \leq \frac{1+\sigma_1}{(1+\sigma_0/6)M}\lambda(x_0).$$

Since x_0 is the solution of $ABS(e/M, 1/6, c)$ we have $\lambda(x_0) \leq c(1+\sigma_0/6)M\lambda^*$. This implies that $\lambda(x_1) \leq c(1+\sigma_1)\lambda^*$.

Now assume (by induction) that $\lambda(x_{s-1}) \leq c(1+\sigma_{s-1})\lambda^*$ after $(s-1)$-st scaling phase. If the second stopping rule is satisfied in $(s-1)$-st phase, then according to the definition of w_s and the relation $\sigma_s = \sigma_{s-1}/2$, we have

$$\lambda(x_s) \leq \frac{(1+\sigma_s)}{(1+2\sigma_s)}\lambda(x_{s-1}) \leq c(1+\sigma_s)\lambda^*.$$

Therefore in both cases we have $\lambda(x_s) \leq c(1+\sigma_s)\lambda^*$ for any s and the solution $x = x_s$ computed in the last phase solves $(P_{\varepsilon,c})$.

□

The remaining task is to find the bound on the number of iterations of the algorithm $\mathcal{F}$ to attain the solution. In order to do so, in the next lemma we show that the decrease of the reduced potential function ϕ_t in each iteration is lower-bounded by a parameter depending only on t, ν and M. This helps us to prove an upper bound on the number of iterations.

Lemma 2.13 *For any two consecutive iterates x, $x' \in B$ within a scaling phase of algorithm $\mathcal{F}$,*

$$\phi_t(x') \leq \phi_t(x) - \frac{t\nu^2}{4M}.$$

From the above bound we are able to obtain the bound on the number of iterations of algorithm $\mathcal{F}$.

Theorem 2.6 *For a given relative accuracy tolerance $\varepsilon \in (0, 1]$, algorithm $\mathcal{F}$ delivers a solution x satisfying $\lambda(x) \leq c(1+\varepsilon)\lambda^*$ with $ABS(p, \varepsilon_3/6, c)$ in $N = O(M(\ln M + \varepsilon^{-4}\ln\varepsilon^{-1}))$ coordination steps.*

The proof is similar to that of Theorem 2.6 but a little different technique for the stopping rules.

Remark: The running time here is worse than that of Algorithm $\mathcal{L}$. However, a block solver $ABS(p, \varepsilon/6, c)$ is required in Algorithm $\mathcal{L}$ while here we only need a $ABS(p, \varepsilon_3/6, c)$.

Similar to the special case of small c ($\ln c = O(\varepsilon)$) discussed in Section 2.3.2, we here can also design a faster algorithm $\bar{\mathcal{F}}$ with only the first stopping rule. It can be proved that $\bar{\mathcal{F}}$ can solve both primal and dual problems with a better bound on the number of iterations. Therefore we have the following result:

Corollary 2.1 *If $c \leq 1/k_\varepsilon$, the algorithm $\bar{\mathcal{F}}$ can generate a pair x and p solving both (P_ε) and (D_ε) with only the weak approximate block solver $ABS(p, \varepsilon_3/6, c)$ within $O(M(\ln M + \varepsilon^{-4}))$ iterations.*

Here for the numerical errors we have the same argument as in Section 2.3.

2.5.3 Better running time

The number of iterations of the algorithm for primal problem in $\mathcal{L}$ is bounded by $O(M(\ln M + \varepsilon^{-2} \ln \varepsilon))$, which is better than the bound in Theorem 2.6. Here we also get such an algorithm with this technique.

We can develop an algorithm $\mathcal{F}'$ by slight modification of the stopping rules. Suppose $r \in (0, 1)$ is a constant. Here a function $h(r)$ is defined as:

$$h(r) = \begin{cases} \dfrac{2r(1-r)}{3(1+r)^2}, & \text{if } r \geq \dfrac{3}{4}; \\ \dfrac{6(1-r)}{7(6-r)}, & \text{otherwise.} \end{cases} \tag{2.19}$$

In addition, we define a parameter by $k_{r,\varepsilon} = 1 - h(r)\varepsilon_4$, where

$$\varepsilon_4 = \frac{(1-h(r)) - \sqrt{(1-h(r))^2 - 4h(r)\varepsilon}}{2h(r)} > \frac{\varepsilon}{1-h(r)}.$$

Define $\sigma'_s = k_{r,\varepsilon}\sigma_s$. Then the stopping rules of $\mathcal{F}'$ are as follows:

$$\begin{aligned} &\textit{Rule } 1: && \nu \leq r\sigma'_s/6; \\ &\textit{Rule } 2: && \lambda(x) \leq w_s\, \lambda(x_{s-1}), \end{aligned} \tag{2.20}$$

Lemma 2.12 is still valid for algorithm $\mathcal{F}'$. Similar to that for $\mathcal{F}$, we have the following theorem:

Theorem 2.7 *If algorithm $\mathcal{F}'$ stops, then for any $\varepsilon \in (0,1]$ the solution x delivered satisfies $(P_{\varepsilon,c})$ with $ABS(p, \varepsilon_4/6, c)$.*

As for the running time, we have also the same bound on increase of reduced potential function for $\mathcal{F}'$ as in Lemma 2.13. To find the bound on number of iterations of algorithm $\mathcal{F}'$, we can just apply the similar argument to the proof of Theorem 2.6. Since here r is a constant in $(0,1)$, we have the following theorem:

Theorem 2.8 *For a given relative accuracy $\varepsilon \in (0,1]$, the number of coordination steps of algorithm $\mathcal{F}'$ is bounded by $N = O(M(\ln M + \varepsilon^{-2} \ln \varepsilon^{-1}))$.*

This bound exactly matches the bound of Algorithm $\mathcal{L}$. But here we just need a weaker block solver.

Remark: We find that if we design the first stopping rule as $\nu \leq v$ for any $v < t$, we can always have a $\varepsilon' > \varepsilon$ for $ABS(p, \varepsilon'/6, c)$ called in algorithm. A reasonable choice, $v = t^q$ for larger q, can generate a larger ε'. Unfortunately this kind of improvement is very limited and the running time increases considerable for the bound on the number of iterations is $O(M(\ln M + \varepsilon^{-2q} \ln \varepsilon^{-1}))$.

Chapter 3

Applications for the Min-Max Resource Sharing Problem

In this chapter, as applications of the Convex Min-Max Resource-Sharing problem, we shall study the Multicast Congestion problem in communication networks and the Minimum Range Assignment problem in ad-hoc networks. We formulate both problems to certain linear programs with the form of Packing problem. Then we can apply the Algorithm $\mathcal{L}$ in Section 2.3 for these problems. We show also the block problems have only weak solvers with approximation $c > 1$. With these applications we are able to demonstrate how to find the block problem for a given problem such that the Algorithm $\mathcal{L}$ can be used.

3.1 The Multicast Congestion problem in communication networks

We are given a graph $G = (V, E)$ to represent a communication network where $|V| = n$ and $|E| = m$ and a set of multicast requests $S_1, \ldots, S_k \subseteq V$. A feasible solution to the Multicast Congestion problem in communication networks is a set of k trees $T_1, \ldots, T_k$ where T_i connects the vertices in S_i. The goal of the Multicast Congestion problem is to find a solution of k trees minimizing the maximum *edge congestion* (the number of times an edge is used). In this section we present a randomized asymptotic approximation algorithm for the Multicast Congestion problem in communication networks based on the Algorithm $\mathcal{L}$ in Section 2.3 and the algorithm in [5]. We show that our algorithm is able to overcome the difficulties of an exponential number of variables and a weak block solver for the Steiner tree problem.

For any given error tolerance $\varepsilon \in (0,1)$ our approximation algorithm delivers a solution with a constant ratio in $O(m(\ln m + \varepsilon^{-2}\ln\varepsilon^{-1})(k\beta + m\ln\ln(m\varepsilon^{-1})))$ time, where β is the running time of the approximate solver for the Steiner tree problem.

3.1.1 The problem and known results

We consider the MULTICAST CONGESTION problem in communication networks. Let an undirected graph $G = (V, E)$ represent a communication network with n vertices and m edges. Each vertex of G represents a processor which is able to receive, duplicate or deliver packets of data. A *multicast request* is a subset S of vertices (called *terminals*), which are connected such that they can receive copies of the same data packet from the source simultaneously. In order to fulfill a request, one subtree of G spanning S is generated, called a *S-tree*. In the MULTICAST CONGESTION problem in communication networks we are given G and a set of multicast requests $S_1, S_2, \ldots, S_k \subseteq V$. A feasible solution is a set of k trees $T_1, \ldots, T_k$, where each tree T_i connects the terminals S_i for the i-th multicast request, called S_i-tree. The *congestion* of an edge in a solution is the number of S_i-trees which use the edge. The goal of the MULTICAST CONGESTION problem in communication networks is to find a solution of S_i-trees minimizing the maximum edge congestion.

A simple example to reduce the maximum edge congestion in a communication network with requests is illustrated in Figure 3.1 and 3.2. In this example we are given a graph with a vertex set $V = \{1, \ldots, 6\}$. The edge set is $E = \{(1,2),(2,3),(3,4),(1,5),(4,6),(5,6),(2,5),(3,6)\}$. There are two requests $S_1 = \{1,4,5,6\}$ and $S_2 = \{3,5\}$. In Figure 3.1 two trees $T_1 = \{(1,5),(5,6),(4,6)\}$ and $T_2 = \{(3,6),(5,6)\}$ are generated to connect S_1 and S_2. The maximum edge congestion is 2 and it occurs on edge $(5,6)$. In Figure 3.2 we still use the same tree T_1 as in Figure 3.1 for the request S_1. However, we generate $T_2 = \{(2,3),(2,5)\}$. Therefore the maximum edge congestion is reduced to 1 (this is also the optimal solution).

The MULTICAST CONGESTION problem in communication networks is a generalization of the standard routing problem of finding integral paths with minimum congestion where each request consists of only two terminals. It is in fact a generalization of the problem of finding edge disjoint paths. This problem is $\mathcal{NP}$-hard [67]

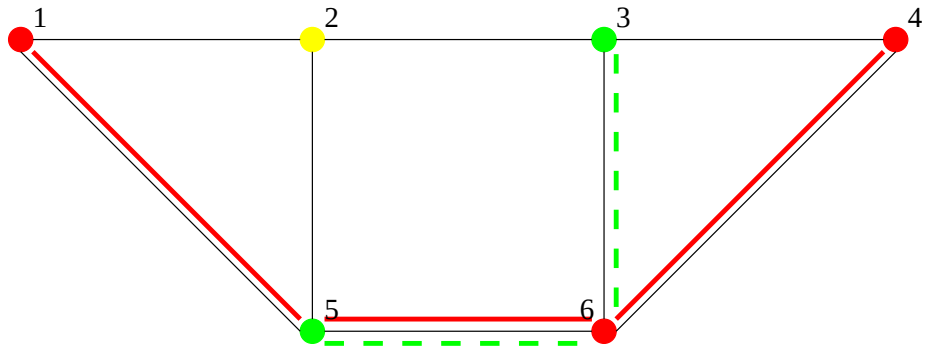


Figure 3.1: An example to reduce maximum edge congestion (high congestion).

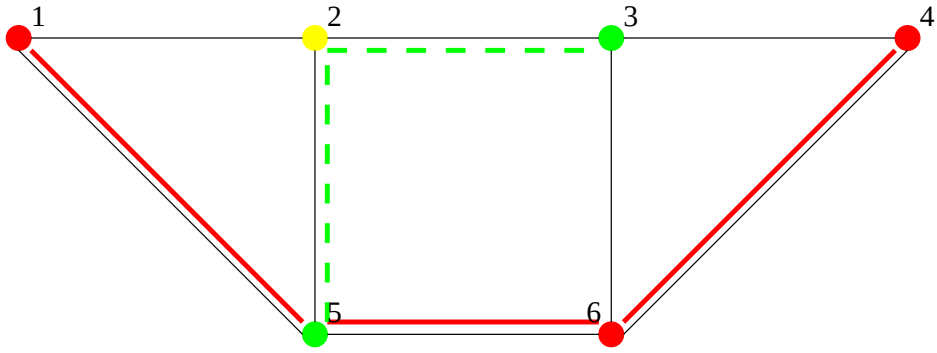


Figure 3.2: An example to reduce maximum edge congestion (low congestion).

and hence the MULTICAST CONGESTION problem is also $\mathcal{NP}$-hard.

A related problem is the *Steiner tree problem*, a classical problem in graph theory. Given a graph $G = (V, E)$, a set $S \subseteq V$ of terminals and a *length function* (*cost*) on the edges, a *Steiner tree* T is a subtree spanning all vertices in S. The vertices of T may be in $V \setminus S$. The goal of the Steiner tree problem in graphs is to find a minimum Steiner tree, i.e., a Steiner tree with minimal total edge length. Compared with the MULTICAST CONGESTION problem, in the Steiner tree problem, there is only a single multicast with a slightly different objective function. Furthermore, we will show that in our algorithm, the Steiner tree problem is exactly the block problem of the MULTICAST CONGESTION problem and our algorithm depends on the approximate solver of it. Unfortunately, this problem is also proved $\mathcal{NP}$-hard [67]. We review the results of approximation algorithms and inapproximibility for the Steiner tree problem briefly in the Subsection 3.1.2. Details of the Steiner tree problem can be found in [38, 95, 30].

Since the MULTICAST CONGESTION problem is $\mathcal{NP}$-hard, interests turn to the

approximation algorithms. In [100] a routing problem in the design of a certain class of VLSI circuits was studied as a special case of the MULTICAST CONGESTION problem. The goal is to reduce the edge congestion of a two-dimensional rectilinear lattice with a specific finite set of trees. Solving the relaxation of the integer linear program and applying randomized rounding yield a randomized algorithm where the congestion is bounded by $\mathcal{OPT} + O(\sqrt{\mathcal{OPT} \ln(n^2/\varepsilon)})$ with probability $1 - \varepsilon$ when $\mathcal{OPT}$ is sufficiently large, where $\mathcal{OPT}$ is the optimal value.

Vempala and Vöcking [107] proposed an approximation algorithm for the MULTICAST CONGESTION problem based on the idea of randomized rounding of the solution to the linear relaxation of their integer linear program, which is related to the dual program of our method. However, the linear program in their model has an exponential number of constraints. Thus they had to apply a separation oracle to solve it, which increases the complexity. Afterwards they decomposed the fractional solution for each multicast into a set of paths instead of trees in [100]. Within a number of $O(\ln n)$ iterations of the rounding procedure, an $O(\ln n)$-approximate solution can be delivered in time $O(n^6\alpha^2 + n^7\alpha)$ where α involves the number k and some other logarithmic factors.

Carr and Vempala [15] proposed a randomized asymptotic algorithm for the MULTICAST CONGESTION problem with a constant approximation ratio. They analyzed the solution to the linear programming relaxation by the ellipsoid method, and showed that it is a convex combination of S_i-trees. By picking a tree with probability equal to its convex multiplier, one can obtain a solution with congestion bounded by $2\exp(1)c \cdot \mathcal{OPT} + O(\ln n)$ with probability at least $1 - 1/n$, where $c > 1$ is the approximation ratio of the solver for the Steiner tree problem applied as an oracle. The algorithm needs $\tilde{O}(n^7)$ time including k as a multiplication factor.

Baltz and Srivastav [4] studied the problem and proposed a formulation based on the ideas of Klein et al [75] to solve the concurrent multicommodity flow problem with uniform capacities. The integer linear program has an exponential number of variables. Then they constructed a combinatorial LP-algorithm to obtain a polynomial number of S_i-trees for each multicast S_i, in order to avoid the ellipsoid method or the separation algorithm. Furthermore a randomized rounding technique was applied. The solution of their algorithm is bounded by

$$r = \begin{cases} (1+\varepsilon)c \cdot \mathcal{OPT} + (1+\varepsilon)(\exp(1)-1)\sqrt{c \cdot \mathcal{OPT} \ln m}, & \text{if } c \cdot \mathcal{OPT} \geq \ln m, \\ (1+\varepsilon)c \cdot \mathcal{OPT} + \dfrac{(1+\varepsilon)\exp(1)\ln m}{1+\ln(\dfrac{\ln m}{c \cdot \mathcal{OPT}})}, & \text{otherwise.} \end{cases} \tag{3.1}$$

For the case $c \cdot \mathcal{OPT} \geq \ln m$ the bound is in fact $(1+\varepsilon)\exp(1)c \cdot \mathcal{OPT}$ and otherwise it is $(1+\varepsilon)c \cdot \mathcal{OPT} + O(\ln m)$. The running time is $O(\beta n k^3 \ln^3(m\varepsilon^{-1})\varepsilon^{-9} \cdot \min\{\ln m, \ln k\})$ where β is the time to approximately solve the Steiner tree problem. With a 2-approximate solver of the Steiner tree problem, the total running time of the algorithm is $\tilde{O}(n(m + n \ln n))$ for constant k and ε, which is an improvement compared to [15, 107].

Here we present a randomized asymptotic approximation algorithm for the Multicast Congestion problem in communication networks. Based on the idea in [4] an integer linear program is constructed. Then we apply the approximation algorithm for the Convex Min-Max Resource-Sharing problem in Section 2.3 to solve the linear programming relaxation. Finally randomized rounding is used to obtain a feasible solution to the original integer linear program. We show that the block problem to solve the linear programming relaxation is the Steiner tree problem. The approximation ratio hence is the same as that in [4] or [15] if their rounding techniques are applied respectively. However, we improve the running time to $O(m(\ln m + \varepsilon^{-2}\ln\varepsilon^{-1})(k\beta + m\ln\ln(m\varepsilon^{-1})))$, which is better for the case of large k or small ε^{-1}.

Recently Baltz and Srivastav [5] proposed three approximation algorithms for the Multicast Congestion problem in communication networks based on algorithms for Convex Min-Max Resource-Sharing problem and the multicommodity flow problem by [42, 89, 98]. The fastest one has an improved running time of $O(k(m+\beta)\varepsilon^{-2}\ln k \ln m)$. They also did some implementation to explore the behaviour of the algorithms with typical instances. They found that the algorithm in [42] is very impractical. In addition, they also presented a heuristic which can find near-optimal solution within a few iterations.

Main ideas. The bottleneck of the problem is to solve the linear programming relaxation with an exponential number of variables. The Algorithm $\mathcal{L}$ in Section 2.3 has a coordination complexity $O(m(\ln m + \varepsilon^{-2} \ln \varepsilon^{-1}))$ depending only on m and ε^{-1}. Thus we apply it for the Multicast Congestion problem. In addition, we prove that the block problem is the Steiner tree problem. Since the block problem is proved hard to be approximated, only a weak block solver can be employed with an approximation ratio $c > 1$. However, the Algorithm $\mathcal{L}$ can solve this kind of problems within a data-independent number of iteration (calls to the weak block solver). Thus we are able to find an approximate solution with the same approximation ratio as the solver of the Steiner tree problem but with a running time polynomial in only m, n and ε^{-1}.

Related results. The algorithms in [89] can also be applied to solve the linear packing problem with an approximate block solver. In fact Charikar et al [16] noticed that the algorithm in [89] can also be generalized to the case of a weak block solver. If their method is applied to solve the linear programming relaxation, it can be estimated that the width ρ here is bounded by $O(k)$. However, as their algorithm is for the decision version of the problem, an approximate solution to the optimization problem is obtained by the binary search. So a number of $\ln(k\varepsilon^{-1})$ steps are necessary. Hence the overall running time is $O(\varepsilon^{-2}k^2\beta \ln(m\varepsilon^{-1}) \ln(k\varepsilon^{-1})$. Thus our result is better for the case of large k. In addition, the algorithm in [111] can also be used to solve the packing problem with weak block solver. If it is applied here, a similar analysis shows that the overall running time to solve the multicast congestion problem is $O(k^2\varepsilon^{-2}\beta \ln m)$.

3.1.2 The Steiner tree problem

The *Steiner tree* problem (definition in Subsection 3.1.1) is the block problem of the Multicast Congestion problem and our algorithm depends on the solver for the Steiner tree problem (see also Subsection 3.1.3). However, the following conclusion holds [67]:

Proposition 3.1 *The Steiner tree problem in graphs is $\mathcal{NP}$-hard, even for unweighted graphs.*

In this way, much interest has turned to approximation algorithms for the Steiner tree problem.

A natural way is to use a *minimum spanning tree* instead of a minimum Steiner tree. It is worth noting that if the leaves of the minimum spanning tree are same as the terminals of the minimum Steiner tree, then the two trees are identical. In addition, it is well-known that there exist polynomial-time algorithms for the minimum spanning tree problem. This approach was attributed to Moore (cf [43]) and has been rediscovered many times. To solve the minimum spanning tree problem, the algorithms in [76] and [96] based on the greedy idea can be applied. The former starts with a forest, all trees in which are connected gradually to form a single tree at last. The latter begins with arbitrary vertex spanning to all vertices by a labeling and spanning procedure. The complexity is $O(m + n \ln n)$ where $m = |E|$ and $n = |V|$. In this way, an approximation algorithms for the Steiner tree problem can be obtained with an approximation ratio $c = 2$ and the running time of the algorithm is $O(m + n \ln n)$ [85, 36].

In 1990, Zelikovsky [113] studied the case of 3-Steiner trees. A full Steiner tree is a Steiner tree where all terminals are leaves. Then a general Steiner tree can be decomposed into a certain number of full components which are connected on the non-terminal vertices in the tree. A 3-Steiner tree is a Steiner tree where each full component has at most 3 terminals. With the analysis of this concept and a greedy algorithm, Zelikovsky proposed an 11/6-approximation algorithm, which is the first one with approximation ratio less than 2. The running time of the algorithm is $O(mn + s^4)$ where $s = |S|$ is the number of terminals.

The idea was generalized to a k-Steiner tree and a 1.734-approximation algorithm was proposed for large k [9]. Afterwards several other approximation algorithms for the Steiner tree problem are based on Zelikovsky's idea. Prömel and Steger [94] addressed a randomized 5/3-approximation algorithm instead of the simple greedy idea. Later another concept, loss, was introduced in the analysis, which can be used to measure the length required in the connection of a full component. Thus a 1.644-approximation algorithm was constructed [69] and improved to 1.598 [51]

soon.

Almost all above algorithms choose certain full components and keep them for the overall solution. However, it is possible that some later but better full components disagree with a previously accepted one (two components disagree if they share at least two terminals). Noting this shortage, Robins and Zelikovsky [102] designed the best known approximation algorithm for the Steiner tree problem by the improvement of updating of accepted full components to achieve the maximal ratio of gain to loss. The approximation ratio is $1 + (\ln 3)/2 \approx 1.550$.

The details of the development of approximation algorithms for the Steiner tree problem can be found in the surveys [38, 95, 30].

On the other hand, negative results have also been proved for the approximation algorithms. According to the PCP-theory with the reduction of the problem Node-Cover-B to the Steiner tree problem in graphs, the following result holds [3, 10]:

Proposition 3.2 *There exists a constant $\bar{c} > 1$ such that no polynomial-time approximation algorithm for the Steiner tree problem in graphs has approximation ratio less than $\bar{c}$ unless $\mathcal{P}=\mathcal{NP}$.*

Thus another interesting topic is to find the lower bound on the approximation ratio $\bar{c}$. The previous best lower bound was $136/135 \approx 1.0074$ due to Thimm [105] by an inapproximability bound for Max-E3-Lin-2. The idea is to design a reduction to linear equations modulo 2 with three variables per equation. Recently Chlebík and Chlebíková [18] improved the lower bound to $96/95 \approx 1.0105$, which is the best known negative result.

In Subsection 3.1.3 the relation between the Multicast Congestion problem in communication networks and the Steiner tree problem is shown. The hardness of the Steiner tree problem leads to the hardness of the Multicast Congestion problem. Then the approximation ratio of our algorithm is same as the ratio of the approximate solver for the Multicast Congestion problem.

3.1.3 Approximation algorithm for the Multicast Congestion problem in communication networks

In this subsection we develop a randomized polynomial time approximation algorithm for the Multicast Congestion problem based on the algorithms in [5].

Given an undirected graph $G = (V, E)$, where $|V| = n$ and $|E| = m$, and subsets $S_1, \ldots, S_k \subseteq V$, we will find a set of subtrees, each spanning a subset, to minimize the maximum edge congestion. According to the idea in [75] of the concurrent multicommodity flow problem with uniform capacities, an integer linear program can be formulated for the Multicast Congestion problem.

Let $\mathcal{T}_i$ be the set of all S_i-trees for $i \in \{1, \ldots, k\}$. Here the cardinality of $\mathcal{T}_i$ may be exponentially large. Define by $x_i(T)$ a variable indicating whether the tree $T \in \mathcal{T}_i$ is chosen in solution for the multicast request S_i. The an integer linear program for the minimum multicast congestion problem is

$$\begin{array}{lll} \min & \lambda & \\ \text{s.t.} & \sum_{T \in \mathcal{T}_i} x_i(T) = 1, & \text{for all } i \in \{1, \ldots, k\}; \\ & \sum_{i=1}^{k} \sum_{T \in \mathcal{T}_i \ \&\ e \in T} x_i(T) \leq \lambda, & \text{for all } e \in E; \\ & x_i(T) \in \{0, 1\}, & \text{for all } i \text{ and all } T \in \mathcal{T}_i. \end{array} \tag{3.2}$$

Here λ is the maximum congestion over all edges. The first set of constraints means that there is one and only one tree in $\mathcal{T}_i$ is selected for the request S_i. The second set of constraints means that on every edge the congestion is not more than λ. As the usual technique, we shall consider the linear programming relaxation of (3.2) to obtain a fractional solution, where $x_i(T) \in [0, 1]$.

The difficulty to solve the linear programming relaxation of (3.2) is the exponential number of variables $x_i(T)$. In fact this is the bottleneck of solving the Multicast Congestion problem in communication networks. However, we find that the linear programming relaxation of (3.2) is in fact a Packing problem mentioned in Chapter 2, which is the linear case of the Convex Min-Max Resource-Sharing problem. Thus we can use the Algorithm $\mathcal{L}$ in Section 2.3. Here the packing constraints are

$$f_e(x) = \sum_{i=1}^{k} \sum_{T \in \mathcal{T}_i \ \&\ e \in T} x_i(T) \leq \lambda.$$

From the linear programming relaxation of (3.2) we are able to decompose the domain of variables $B = B_1 \times \ldots \times B_k$ where k is the number of multicast requests (subsets) S_i and the block B_i is as follows:

$$B_i = \{(x_i(T))|T \in \mathcal{T}_i, \sum_{T\in\mathcal{T}_i} x_i(T) = 1, x_i(T) \geq 0\}, \qquad \text{for } i = 1, \ldots, k, \tag{3.3}$$

where $(x_i(T))$ represents the vector whose components are $x_i(T)$ for all $T \in \mathcal{T}_i$. The number of components of vector $(x_i(T))$ can be exponentially large.

We now consider the block problem, which is to minimize the value of $p^T f(x)$ for a given price vector p. According to the block structure of the set B, the optimization of $p^T f(x)$ over $x \in B = B_1 \times \ldots \times B_k$ can be conducted independently over all blocks B_i. In addition, each block B_i is a simplex according to (3.3). Since all constraints are linear in variables, the optimal value of the objective function over such a simplex is achieved at one of the vertices.

For a given $p \in P$, the block problem is as follows:

$$\begin{aligned}
\min_{x\in B} p^T f(x) &= \min_{x\in B} \sum_{e\in E} \left(p_e \cdot \sum_{i=1}^{k} \sum_{T\in\mathcal{T}_i \& e\in T} x_i(T) \right) \\
&= \min_{x\in B} \sum_{e\in E} \sum_{i=1}^{k} \sum_{T\in\mathcal{T}_i \& e\in T} p_e x_i(T) \\
&= \min_{x\in B} \sum_{i=1}^{k} \sum_{e\in E} \sum_{T\in\mathcal{T}_i \& e\in T} p_e x_i(T) \\
&= \sum_{i=1}^{k} \min_{x\in B} \sum_{e\in E} \sum_{T\in\mathcal{T}_i \& e\in T} p_e x_i(T) \\
&= \sum_{i=1}^{k} \min_{x\in B} \sum_{T\in\mathcal{T}_i} \sum_{e\in T} p_e x_i(T) \\
&= \sum_{i=1}^{k} \min_{T\in\mathcal{T}_i} \sum_{e\in T} p_e.
\end{aligned} \tag{3.4}$$

The last equality holds as in each block B_i, $i = 1, \ldots, k$, the variable $x_i(T)$ is in a $|\mathcal{T}_i|$ dimensional simplex and we can choose a tree $T \in \mathcal{T}_i$ which has the minimal value of the sum of price vectors on its edges $\sum_{e\in T} p_e$ and set the corresponding $x_i(T) = 1$ (and other components are zeros) to achieve the minimum. Here p_e is the price vector component of $e \in E$ which can be calculated in advance in Algorithm

$\mathcal{L}$ by (2.4) according to the previous value of $x_i(T)$. Now the goal of the block problem is to find, for each $i = 1, \ldots, k$, a tree $T \in \mathcal{T}_i$ that minimizes $\sum_{e \in T} p_e$. Since $\mathcal{T}_i$ contains all trees connecting vertices in S_i, regarding the price vector as the weight functions defined on edge set E, the block problem is for each $i = 1, \ldots, k$, to find a tree connecting S_i (allowing vertices in $V \setminus S$) such that the sum of edge weight of the tree is minimized. Therefore for each $i = 1, \ldots, k$, we have a Steiner tree problem (as reviewed in Subsection 3.1.2) with terminal set S_i, and the weights assigned on edges are given by the price vector p.

However, as we showed in the Subsection 3.1.2, there exists a constant $\bar{c} > 1$ such that there is no polynomial time algorithm to solve the Steiner tree problem with an approximation ratio less than $\bar{c}$. So we are only able to find approximate solvers for the Steiner tree problem with ratios $c > 1$, i.e., weak block solvers for the Multicast Congestion problem in communication networks. Thus if the algorithms for the Packing problemin [42, 46, 109] are applied here, the running times are data dependent. To avoid this, we here use the Algorithm $\mathcal{L}$ in Section 2.3 to solve the Multicast Congestion problem with an approximate solver for the Steiner tree problem.

In addition, in the overall running time we need to consider the numerical overhead to compute the approximate value of p according to (2.4) and (2.3). As studied in Subsection 2.4.2, in each iteration we need $O(M \ln(M\varepsilon^{-1}))$ time by bisection method or $O(M \ln\ln(M\varepsilon^{-1}))$ time by the Newton's method. Since in each iteration we need to call the approximate solver for the Steiner tree problem k times for all blocks $B_1, \ldots, B_k$, the overall running time is $O(m(\ln m + \varepsilon^{-2} \ln \varepsilon^{-1})(k\beta + m \ln\ln(m\varepsilon^{-1})))$, where β is the running time of the approximate solver for the Steiner tree problem.

After the linear programming relaxation of 3.2 is approximately solved by Algorithm $\mathcal{L}$, we use the same rounding technique as in [4, 99] or in [15]. Hence we immediately have the following theorem:

Theorem 3.1 *For $OPT \geq \Omega(\ln m)$ (i.e. $OPT \geq \ln(mc^{-1})$) there exists an asymptotic polynomial time algorithm for the multicast congestion problem in multicast communication network via solving the Steiner tree problem with a constant approximation ratio $c > 1$.*

3.2 The Minimum Range Assignment problem in ad-hoc networks

In this section we study the MINIMUM RANGE ASSIGNMENT problem in static ad-hoc networks with arbitrary structure, where the transmission distances can violate triangle inequality. We consider two versions of the MINIMUM RANGE ASSIGNMENT problem, where the communication graph has to fulfill either the h-strong connectivity condition (MIN-RANGE(h-SC)) or the h-broadcast condition (MIN-RANGE(h-B)). Both homogeneous and non-homogeneous cases are studied. By approximating arbitrary edge-weighted graphs by paths, we present probabilistic $O(\log n)$-approximation algorithms for MIN-RANGE(h-SC) and MIN-RANGE(h-B), which improves the previous best ratios $O(\log n \log\log n)$ and $O(n^2 \log n \log\log n)$, respectively [110]. The result for MIN-RANGE(h-B) matches the lower bound [103] for the case that triangle inequality for transmission distance holds (which is a special case of our model). Furthermore, we show that if the network fulfils certain property and the distance power gradient α is sufficiently small, the approximation ratio is improved to $O((\log\log n)^{\alpha})$. Finally we discuss applications of our algorithms in mobile ad-hoc networks.

3.2.1 Introduction

Nowadays *wireless communication* network plays an important role in the daily life due to the significant drop in the prices of equipments and the progress in new technology. In traditional wired communication networks signals are transmitted among nodes through fixed cables. However, in some critical environment, the wired backbone networks are impossible or too hard to establish. Thus the demand of wireless communication networks arises. In wireless communication networks there is no infrastructure backbone and for each device the radio signal transmission is conducted in a finite range around it. The locations of devices and the ranges can be adjusted dynamically in order to fulfil certain communication quality requirement and to extend the lifetime of the networks. In general the wireless devices are portable with only limited power resources (e.g., batteries). High quality of communication usually consume more energy and reduce the network lifetime, and vice verse [13].

Hence, a crucial issue of wireless communication networks is to minimize the energy consumption as well as keeping required communication quality.

Among the new generation's model of wireless communication networks, the *ad-hoc wireless network* based on *multi-hop* plays a very promising role [77]. Typical applications of the ad-hoc networks include emergency disaster, battlefield, and monitoring remote geographical region. The multi-hop ad-hoc networks are able to reduce the power consumption and to vary the power used for transmission such that the interference problem can be avoided. Since in the ad-hoc networks the power consumption is increasing in the transmission range (the coverage range) for the same device under the same environmental condition, it is important to study the range control to minimize the overall energy consumption without resulting in too bad communication quality. The RANGE ASSIGNMENT problem is a key subject of energy control in ad-hoc networks, which has been extensively studied in wireless network theory [2, 14, 20, 21, 24, 74].

The (static) RANGE ASSIGNMENT problem on d-dimensional Euclidean spaces ($d \geq 1$) is defined as follows. We are given a set S of stations (radio transmitter/receivers) on d-dimensional Euclidean spaces and the distances between all pairs are known according to their coordinates. The stations can communicate with each other by sending/receiving radio signals. We assume that each station is equipped with omnidirectional antennas such that its transmission range is characterized by the radius that its signals can reach. The message communication happens via *multi-hop transmission*, i.e., a message is delivered from the source to the destination through some intermediate stations and each station in this transmission chain other than the source station is in the coverage range of its predecessor. A range assignment for a set of station S is a function $r : S \to \mathbb{R}^+$, which indicates the radii that stations can cover. For a station $v \in S$ associated with a range $r(v)$ in a network-wise range assignment, its energy consumption (power consumption) is $cost(r(v)) = c(v)(r(v))^\alpha$, where $c(v)$ is a parameter depending on the individual device.The *distance-power gradient* α is a positive real number, usually in the interval $[1, 6]$ in practice. When $c(v)$ is a constant for all $v \in S$, we call the model *homogeneous*, otherwise it is *non-homogeneous* [2]. It is worth noting that the non-homogeneous model can be asymmetric, as the energy consumption for a device on

u to cover v may differ from the energy consumption of another device to conduct the same transmission. The overall energy consumption of a range assignment r is defined as $cost(r) = \sum_{v \in S} cost(r(v)) = \sum_{v \in S} c(v)(r(v))^{\alpha}$.

A range assignment r yields a directed communication graph $G_r = (S, E_r)$, such that for each pair of stations u and v there exists a directed edge $(u, v) \in E_r$ if and only if v is at an (Euclidean) distance at most $r(u)$ from u. For the purpose of a variety of communication requirements, the communication graph G_r must fulfil one of following two properties Π_h for any fixed h, $h \in \{1, \ldots, n-1\}$, where n is the number of stations:

- *h-strong connectivity*: from every station to any other station, G_r must contain a directed path of at most h hops (edges),
- *h-broadcast*: G_r must contain a directed source spanning tree rooted at a source station with depth at most h.

The goal of the Minimum Range Assignment problem (shortly, Min-Range) is to find a range assignment r for a given S such that G_r fulfils a given property Π_h and the overall energy consumption $cost(r)$ is minimized. We use notations Min-Range(h-SC) (respectively, Min-Range(h-B)) for the corresponding Minimum Range Assignment problems, when Π_h is the property of h-strong connectivity (respectively, h-broadcast). The detailed description of the problem can be also found in [103].

Known results. It is obvious that when $\alpha > 1$ the objective function is nonlinear. Therefore many traditional techniques such as linear programming can not be employed here, which makes the problem hard. Previously attention was mainly paid to the Minimum Range Assignment problems defined on one dimensional (Euclidean) spaces, which is equivalent to the case that a set S of stations are placed along a line (or a path). Polynomial time algorithms by dynamic programming were addressed for both homogeneous and non-homogeneous cases for Min-Range(h-B) on one dimensional (Euclidean) spaces in [74, 23, 2]. In the homogeneous case the Min-Range(h-SC) problem is polynomial time solvable for $h = 2$ (respectively, $h = n - 1$) within a running time $O(n^3)$ (respectively, $O(n^4)$) as presented in

[21] (respectively [74]). However, for any other h it is still open whether the MIN-RANGE(h-SC) problem on one dimensional spaces can be solved in polynomial time. Clementi et al. [21] proposed a 2-approximation algorithm for any $h \in \{2, \ldots, n-1\}$ for the homogeneous case with a complexity $O(hn^3)$. Furthermore, the algorithm can be extended to the non-homogeneous case.

There are only few results on d-dimensional Euclidean spaces for $d \geq 2$ (see [22, 103]). In case of $h = n - 1$, algorithms based on the minimum spanning tree technique can deliver a solution with a constant approximation ratio for the MIN-RANGE(h-B) problem [20], where the constant ratio depends on the dimension d and the distance-power gradient α. For the MIN-RANGE(h-SC) problem and $h = n-1$, a 2-approximation algorithm was addressed in [74]. Recently, Calinescu et al. [14] developed an $O(\log^{\alpha} n)$-approximation algorithm for the MIN-RANGE(h-B) problem on d dimensional Euclidean spaces. They also presented $(O(\log n), O(\log n))$ bicriteria approximation algorithms for both MIN-RANGE(h-B) and MIN-RANGE(h-SC) problems. For any fixed h, denote by OPT the optimal overall cost of the input instance. Then their algorithms can deliver a range assignment for the given instance such that the number of hops is bounded by $O(h \log n)$ and the total cost is at most $O(\log n)OPT$.

In [24] the MIN-RANGE(h-SC) problem was proved in $\mathcal{A}$v-$\mathcal{APX}$ for any fixed $h \geq 1$ and the problem is $\mathcal{APX}$-hard on d-dimensional Euclidean spaces for $d \geq 3$. In [110] the MINIMUM RANGE ASSIGNMENT problem on general metric spaces was studied. Approximation algorithms with ratios $O(\min\{\log n \log\log n, (\log n)^{\alpha}\})$ and $O(n^2 \min\{\log n \log\log n, (\log n)^{\alpha}\})$ were proposed for MIN-RANGE(h-B) and MIN-RANGE(h-SC), respectively. This was the first work to explore MINIMUM RANGE ASSIGNMENT problem on general spaces. In their model, the triangle inequality is still required for the transmission distance.

Our contributions. In this paper, we first propose a new model of the MINIMUM RANGE ASSIGNMENT problem. We show that our model is a generalization of previous models and is realistic. We notice that the transmission cost for the same device is not homogeneous on space, i.e., the costs from different locations to cover the same distance can be different, due to environmental factors. In this

case it is invalid to measure the cost by Euclidean distance. Thus we consider the problem with a station set S on a space with transmission distance (see Subsubsection 3.2.2.1) instead of the original Euclidean distance. In such an instance, the transmission distance can even violate the triangle inequality, and no previous study remains valid in this case. Our main ideas are as follows. For a given instance of the Minimum Range Assignment problem, in the first step we reduce it to an instance with only a simple network structure (e.g., a path in our study) and with bounded distortion of distance between all pairs of stations. Then in the second step we use some existing algorithms for the reduced instance. By this strategy we are able to obtain an algorithm with an approximation ratio bounded by the product of the ratios of both two steps. In this paper, based on the above idea, we first present a probabilistic algorithm to approximate any edge-weighted graph by a collection of paths, such that for any pair of nodes the expected distortion of shortest path distance is at most $O(\log n)$, where n is the number of nodes in the graph. The paths in the collection and the corresponding probability distribution are given by solving a packing problem defined in Chapter 2 and [46, 89, 109], and a solver of the Minimum Linear Arrangement problem [97] is employed as an oracle. With this algorithm we are able to approximate the general static ad-hoc networks to paths and run known algorithms in [2, 21, 23, 74] for the Minimum Range Assignment problem on one dimensional Euclidean spaces (lines, which correspond paths). Therefore this strategy leads to probabilistic $O(\log n)$-approximation algorithms for the Minimum Range Assignment problem (both Min-Range(h-B) and Min-Range(h-SC)) for general static ad-hoc networks. The ratio for the Min-Range(h-B) problem reaches the lower bound for networks that triangle inequality is valid for transmission distance [103]. It is worth noting that the case in [103] is only a special case of our model as here we allow violation of the triangle inequality. In addition, the large factor $O(n^2)$ in the approximation ratio of the algorithm for the Min-Range(h-SC) problem in [110] is removed. The ratios for both problems are the same, which also implies that the Min-Range(h-SC) problem is not harder than the Min-Range(h-B) problem. Furthermore, if the input graph of station set fulfils certain property, we show that the approximation ratio can be further reduced to $O((\log \log n)^{\alpha})$. Finally, we study the Minimum Range Assignment problem

in mobile ad-hoc networks, where the locations of the mobile stations can change and the topology of the network can vary. Based on the DSDV protocol [88] we are able to generalize our algorithms to the mobile model.

3.2.2 Preliminaries

We introduce our model and some notations in this section.

3.2.2.1 Our model

In most of previous works for the MINIMUM RANGE ASSIGNMENT problem either h is set as $n-1$ [74, 20], or the station set is on one dimensional (Euclidean) spaces [74, 21, 23, 2]. For studies of MINIMUM RANGE ASSIGNMENT problem on multidimensional spaces, the transmission cost is measured by the (Euclidean) geometric distance [74, 20, 14]. Even in [110], the triangle inequality must hold for the transmission cost.

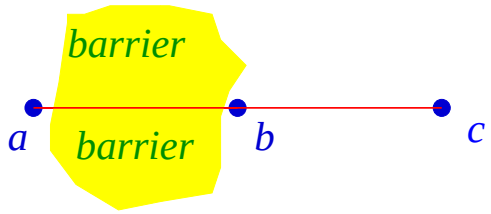


Figure 3.3: Example of difference between the transmission distance and the geometric distance.

The one dimensional model has already been widely studied as a good approximation of the real instances. However, demands on models on multidimensional spaces are gradually increasing, as they can be more precise tools to characterize the real ad-hoc networks. Furthermore, from the engineering point of view, the model with fixed $h = n - 1$ is not practical and hard to control by current protocols [17]. In that model, for any signal transmission, the number of hops used can be uncontrolled. In wireless communication, uncontrolled hops can lead to high probability of errors in coding/decoding, large latency, etc. Thus the quality of communication is significantly reduced. Finally, an important issue is the measurement of the transmission cost in the ad-hoc networks. In almost all of previous works the transmission cost is assumed to be characterized by the (Euclidean) geometric distance. We notice that due to environmental condition, that assumption

is not always true. For instance (Figure 3.3), three stations are along a line, and $d_E(a,b) = d_E(b,c)$, where d_E is the Euclidean geometric distance. However, there are some barriers (forests, buildings) between a and b, while there is nothing between b and c. In such an instance, it costs more energy to send signals from b to a than from b to c, though the geometric distances between above two pairs of stations are the same. Thus the geometric distance is not sufficient to measure the real energy cost in transmission though it is a good approximation. In [103, 110] the model of Minimum Range Assignment problem on metric spaces was suggested where the transmission cost is not represented by the geometric distance. We further notice that on metric spaces the triangle inequality is assumed valid, and this assumption could be violated in real wireless communication networks. For instance (See Figure 3.4), three stations are on a plane. There is a solid barrier (e.g. mountains, large buildings) between stations a and b, while there is no such solid barrier between station pairs a, c and b, c. In this example $d_E(a,b) \leq d_E(a,c)$ and $d_E(a,b) \leq d_E(b,c)$. However, it costs much more to launch a signal transmission between a and b due to the solid barrier. Therefore the energy cost for direct transmission between a and b can be greater than the sum of costs of transmissions between a, c and b, c. Thus the triangle inequality does not hold. When the transmission cost between a and b is unbounded (which does happen in real world), it is impossible to build an ad-hoc network for this instance with only $h = 1$ hop and bounded overall energy cost.

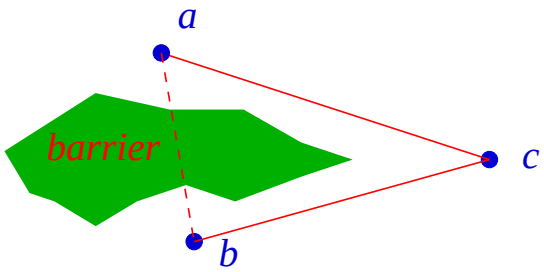


Figure 3.4: Example that the triangle inequality is violated.

In this paper, we propose a new model of the Minimum Range Assignment problem in ad-hoc communication networks. In this model, h can be any arbitrary integer number in $\{1, \ldots, n-1\}$. Furthermore, the stations are on arbitrary spaces, and the transmission cost between each pair can be arbitrary. We propose a concept *transmission distance*, which is a scalable quantity. Given a station set S and

a distance power gradient α, we can measure the *minimum* energy cost $cost(u, v)$ of *directly* sending signals from any station $u \in S$ to any other $v \in S \setminus \{u\}$ with a standard wireless device $c(u) = 1$. The transmission distance between the station pair (u, v) is defined as $d(u, v) = (cost(u, v))^{-\alpha}$. In our model, an instance of MINIMUM RANGE ASSIGNMENT problem $(G(S, E, l_G), \alpha, \Pi_h)$, we are given a complete edge weighted graph G, a distance power gradient $\alpha \geq 1$ and a required property Π_h of the communication graph (either h-strong connectivity or h-broadcast), for $h \in \{1, \ldots, n-1\}$. In the weighted graph G, the vertex set S is the station set and the weight $l(u, v)$ of any edge (u, v) is the transmission distance (which can violate the triangle inequality) between the two endpoints u and v. The edge weight can be infinity if the transmission cost between the corresponding two endpoints is unbounded. Same as previous models, the goal is to find a network-wide arrangement r such that the property Π_h holds in the resulting communication graph and the overall energy cost $cost(r)$ is minimized.

We also notice that this model of the MINIMUM RANGE ASSIGNMENT problem generalize previous models. If the edge weights fulfil the triangle inequality, then our model is reduced to the model on metric space. If the edge weights are Euclidean geometric distances, then it is reduced to classical model on Euclidean spaces. Other restricted models can be naturally reduced from our model.

3.2.2.2 Notations and definitions

We introduce some notations and definitions related to our algorithms in this subsection. Without loss of generality, we only consider connected graphs. Given a graph, we need to embed it in a simpler graph such that the distance between each pair of vertices are approximately preserved. This technique can be employed to solve some hard problems on arbitrary graphs, as an arbitrary graph may have a very complicated structure. We will propose the idea of *probabilistic approximation* of weighted graphs by a collection of simpler graphs (e.g. paths). This is a generalization of the concept of probabilistic approximation of metric spaces addressed in [6], because edge weights can violate triangle inequality.

Given two graphs $G_1 = (V, E_1, l_1)$ and $G_2 = (V, E_2, l_2)$ with the same node set V, G_1 *dominates* G_2 if and only if $d_{G_1}(u, v) \geq d_{G_2}(u, v)$ for all pair u, $v \in V$,

where $d_{G_i}(u, v)$ is the shortest path distance between node pair u and v in G_i. G_1 is also called a *non-contracting* embedding of G_2. The *distortion* is defined as $\max_{u,v \in V} d_{G_1(u,v)}/d_{G_2(u,v)}$.

Suppose that $\mathcal{H}$ is a collection of graphs that have the same node set V as another graph G. Assuming that each graph in $\mathcal{H}$ dominates G, $\mathcal{H}$ is defined to *ρ-probabilistically approximate* G if there is a probability distribution μ over $\mathcal{H}$ such that for each pair of nodes in V the expected distance between them in a graph $H \in \mathcal{H}$ chosen according to μ is at most ρ times the distance between the pair in G, i.e., $E[d_H(u, v)] \leq \rho d_G(u, v)$.

In this paper, we will develop an algorithm to $O(\log n)$-probabilistically approximate any graph by a collection of paths (See Section 3.2.3). Based on this algorithm, we are able to generalize the existing algorithms for the Minimum Range Assignment problems in ad-hoc networks on lines (paths) to arbitrary networks. In order to generate the collection of paths and the probability distribution over it, a packing problem has to be solved (See Chapter 2).

3.2.2.3 Algorithms for approximating metrics

In [6], Bartal proposed the concept of probabilistic approximation of metric spaces by a set of simpler metric spaces. A polynomial time algorithm to $O(\log^2 n)$-probabilistically approximate any metric space on $|V| = n$ nodes by a class of tree metrics was addressed in [6]. The approximation ratio was improved to $O(\log n \log \log n)$ in [7]. However, the numbers of the tree metric spaces are exponentially large in both algorithms. Charikar et al. [16] developed a polynomial time algorithm to construct a probability distribution on a set of $O(n \log n)$ trees metrics for any given metric space induced by a (weighted) graph G on n nodes, such that the expected *distortion* of each edge is not more than $O(\log n \log \log n)$. Recently, Fakcharoenphol et al. [33] improved the bound on distortion to $O(\log n)$ by using some special technique based on the triangle inequality. They also showed that this bound is tight. For the deterministic version of the problem, Matoušek [84] shows that any metric can be embedded into the real line with a distortion $O(n)$. This result is existentially tight as the n-cycle can not be embedded into a line with distortion $o(n)$ [50, 97]. For this problem, Dhamdhere et al. [28] considered a variant to minimize the average

distortion which is defined as the sum of distances over all pairs in the line divided by the sum of those distances in original metric space. For the general metric space they proposed a constant factor approximation algorithm and when the given metric is a tree, a quasi PTAS was addressed.

To generate the tree metrics and decide the probability distribution μ, a linear program with exponential number of variables has to be solved. Indeed this linear program is a packing problem and can be solved approximately by the approximation algorithm $\mathcal{L}$ in Section 2.3 or some other existing fast algorithms [16]. All the algorithms are based on iterative strategy. In each iteration a tree metric is generated and the probabilities on the tree metrics are assigned once, till the algorithms halt.

3.2.3 Approximate a graph by a collection of paths

We will study the problem of probabilistically embedding graphs in paths and develop a probabilistic approximation algorithm. For any given graph, our algorithm will deliver a collection of paths and a probability distribution over it such that the expected distortion is bounded by $O(\log n)$. This algorithm will be employed for approximately solving the Minimum Range Assignment problem in static ad-hoc networks in Section 3.2.4. We believe that it is of independent interests.

Given an edge-weighted graph $G(V, E, l)$, where $|V| = n$, $|E| = m$, and a weight function $l : E \to \mathbb{R}_0^+$ is defined on its edge set. The weight function l can violate the triangle inequality. Without loss of generality, we assume that the diameter of G is bounded by one. Otherwise a simple scaling method can be employed with a running time bounded by $O(n^2)$. Let $\mathcal{P} = \{P_1, \ldots, P_N\}$ be a collection of paths, each connecting all nodes in V. Each path in $\mathcal{P}$ dominates the graph G, i.e., $d_{P_i}(u, v) \geq d_G(u, v)$ for any pair u, $v \in V$ and $i \in \{1, \ldots, N\}$. Here the distance functions d_{P_i} and d_G are shortest path distance in the path P_i and the graph G, respectively. In addition, we assign to every path $P_i \in \mathcal{P}$ a real number $x_i \in [0, 1]$, which represents the probability distribution μ over the path collection $\mathcal{P}$, and the sum of x_i is 1. Denoting by λ the distortion of each edge and $l_G(e)$ the edge length of $e \in E$. The following linear program is to find the probability distribution μ that minimizes the expected edge distortion in $P \in \mathcal{P}$:

$$\begin{array}{ll} \min & \lambda \\ \text{s.t.} & \sum_{i=1}^{N} d_{P_i}(e)x_i \leq \lambda l_G(e), \quad \text{for every edge } e \in E; \\ & \sum_{i=1}^{N} x_i = 1; \\ & x_i \geq 0. \end{array} \tag{3.5}$$

Here the first set of constraints indicates that the expected distortion of every edge $e \in E$ in all paths in $\mathcal{P}$ is bounded by λ. The other constraints are directly from the definition of probability distribution. Notice that in (3.5) the number of variables (i.e., the number of paths in $\mathcal{P}$) can be exponentially large. Thus most traditional algorithms for linear programs are not applicable for it.

We then turn to approximation algorithms for (3.5). We notice that it can be formulated as a packing problem described in Chapter 2 and the packing constraints $f_e(x) = \sum_{i=1}^{N} d_{P_i}(e)x_i/l_G(e)$ are nonnegative linear functions, and the set $B = \{x = (x_1, \ldots, x_N)^T | \sum_{i=1}^{N} x_i = 1, x_i \geq 0\}$ is indeed a simplex. We will apply the approximation algorithms $\mathcal{L}$ in 2.3 to solve this packing problem.

In order to develop an algorithm for (3.5), we need to consider the block problem in advance, which is related to the dual problem and the structure of the set B. As showed in Chapter 2, given a price vector $y \in Y = \{(y_1, \ldots, y_m) | \sum_{e \in E} y_e = 1, y_e \geq 0\}$, the block problem is to find an $\hat{x} \in B$ such that $y^T f(\hat{x}) = \min_{x \in B} y^T f(x)$. With the formulation of the packing constraints in (3.5), the block problem can be simplified as follows:

$$\begin{aligned} \min_{x \in B} y^T f(x) &= \min_{x \in B} \sum_{e \in E} \left(y_e \cdot \sum_{i=1}^{N} \frac{d_{P_i}(e)}{l_G(e)} x_i \right) \\ &= \min_{x \in B} \sum_{i=1}^{N} \left(x_i \cdot \sum_{e \in E} y_e \frac{d_{P_i}(e)}{l_G(e)} \right) \\ &= \min_{P_i \in \mathcal{P}} \sum_{e \in E} \frac{y_e}{l_G(e)} d_{P_i}(e). \end{aligned}$$

The last equality holds because we can choose one path P_i which satisfies that $P_i = \arg\min_{P_k \in \mathcal{P}} \sum_{e \in E} y_e d_{P_k}(e)/l_G(e)$ (which means that P_i has the minimum value of

$\sum_{e\in E} y_e d_{P_i}(e)/l_G(e))$ and set its corresponding probability $x_i = 1$ (and the probabilities of other paths are set as 0) to achieve the optimum. Denote by $w(e) = y_e/l_G(e)$ the weight associated to edge e. Therefore the goal of the block problem is to find a path P connecting all nodes in G such that the value $\sum_{e\in E} w(e)d_P(e)$ is minimized with the given weight function w, for all edge $e \in E$.

The block problem actually is equivalent to the Minimum Linear Arrangement problem (MLA). The problem is defined as follows: Given a graph $G(V, E)$ and nonnegative edge weights $w(e)$ for all $e \in E$, where $|V| = n$ and $|E| = m$. The goal is to find a linear arrangement of the nodes $\sigma : V \to \{1, \ldots, n\}$ that minimizes the sum of the weighted edge lengths $|\sigma(u) - \sigma(v)|$, over all $(u, v) \in E$. If we define the overall cost as follows

$$c = \sum_{(u,v)\in E} w(u, v)|\sigma(u) - \sigma(v)|,$$

then the goal of the Minimum Linear Arrangement problem is to minimize the total cost c. Then we can place all vertices $u \in V$ on a path P (i.e., a one dimensional Euclidean space) and the coordinates are their arrangements $\sigma(u)$. It is obvious that the weight in the Minimum Linear Arrangement problem corresponds to the weight function in our block problem and the length $|\sigma(u) - \sigma(v)|$ corresponds to the distance in the path $d_P(u, v)$. Therefore we can directly apply the algorithms for the Minimum Linear Arrangement problem to solve our block problem to generate a path.

However, the Minimum Linear Arrangement problem is $\mathcal{NP}$-hard [40]. The best known algorithm for the Minimum Linear Arrangement problem is proposed by Rao and Richa [101] and the approximation ratio is $O(\log n)$. Using it we are able to construct an $O(\log n)$-approximation algorithm for the linear program (3.5) with the same ratio. In the algorithm iterative method is applied. An initial solution is set at the beginning. Then the algorithm runs the iterative procedure. In each iteration a pair of solutions to the linear program (3.5) and its dual problem is computed based on previous iterate. The duality gap is then decreasing. In one iteration there are following steps: First with a known solution, a price vector y related to the dual value is calculated to determine the direction of moving of the

iterate. Then an MLA solver is called to generate an approximate block solution according to y and error tolerance ε, i.e., a path delivered by the MLA solver with respect to the weight $w(e) = y_e/l_G(e)$ for any $e \in E$. Finally a new solution is obtained as a linear combination of the known solution and block solution with an appropriate step length. When certain stopping rules are satisfied (which indicates that the duality gap is already small enough to fulfil the required accuracy), the iterative procedure terminates and the solution returned by the last iteration is the desired approximate solution (see Chapter 2).

Algorithm $\mathcal{AG}(G(V, E, l_G), \mathcal{P}, x, \varepsilon)$:
initialization: $i = 1$, $\sigma = 1$, $w(e) = 1/m$, $P_1 = MLA(G(V, E), w)$, $\mathcal{P} = \{P_1\}$;
$x_1 = 1$, $f_e = d_{P_1}(e)/l_G(e)$, $\lambda' = \max_{e\in E} f_e$, $\kappa = (1+\sigma)/((1+\sigma/6)M)$;
while $\sigma > \varepsilon$ **do** /*scaling*/
finished=false;
while *not*(*finished*) **do** /*coordination*/
solve $\frac{\sigma}{6m}\sum_{e\in E}\frac{\theta}{\theta - f_e} = 1$ for θ;
$w(e) = \frac{\sigma}{6m}\frac{\theta}{\theta - f_e}$ for all $e \in E$;
$P_{i+1} = MLA(G(V, E), w)$;
$\mathcal{P} = \mathcal{P} \cup \{P_{i+1}\}$;
$\hat{f}_e = d_{P_{i+1}}(e)/l_G(e)$ for all $e \in E$;
$\nu = \frac{\sum_{e\in E} w(e)(f_e - \hat{f}_e)}{\sum_{e\in E} w(e)(f_e + \hat{f}_e)}$;
$\tau = \frac{\sigma\theta\nu}{12m(\sum_{e\in E} w(e)(f_e + \hat{f}_e))}$;
$x_k = (1 - \tau)x_k$ for $k = 1, \ldots, i$;
$x_{i+1} = \tau$;
$f_e = (1 - \tau)f_e + \tau\hat{f}_e$ for all $e \in E$;
$\lambda = \max_{e\in E} f_e$;
if $\nu \le \sigma/6$ **or** $\lambda \le \kappa\lambda'$ **then** *finished=true*;
$i = i + 1$;
enddo
$\sigma = \sigma/2$;
$\kappa = (1 + \sigma)/(1 + 2\sigma)$;
$\lambda' = \lambda$;
enddo

Table 3.1: Approximation Algorithm $\mathcal{AG}$ to probabilistically approximate arbitrary edge weighted graph.

Suppose that we have an approximate solver $MLA(G(V,E),w)$ for the MINIMUM LINEAR ARRANGEMENT problem for a given graph $G=(V,E)$ and a weight edge weight function w. For any given $\varepsilon \in (0,1)$ the approximation algorithm $\mathcal{AG}$ in Table 3.1 can deliver a collection $\mathcal{P}$ of paths and a probability distribution μ represented by the vector x, such that for each edge $e \in E$, the expected distortion over all paths in $\mathcal{P}$ is at most $c(1+\varepsilon)$, where c is the approximation ratio of the MLA solver.

We notice that in each iteration there is at most one new path generated. Thus the number of paths generated in the algorithm is bounded by the number of iteration, which is $O(m \log m)$ (see Chapter 2). This shows that in $\mathcal{P}$ there are at most $O(m \log m)$ paths associated with non-zero probability. It is obvious that all paths in $\mathcal{P}$ dominate the original graph. In addition, in each iteration of $\mathcal{L}$ in Section 2.3, a numerical overhead of $O(m \log\log m)$ is required. Therefore we have the following theorem for probabilistic approximating a graph by polynomial number of paths:

Theorem 3.2 *Given a graph $G=(V,E,l)$, where $|V|=n$, $|E|=m$, and an edge weight function $l: E \to \mathbb{R}_0^+$, there exists an algorithm that generates a collection $\mathcal{P}$ of $O(m \log m)$ paths and a probability distribution μ over the collection $\mathcal{P}$, such that for any edge $e \in E$, the expected distortion of e in $\mathcal{P}$ is bounded by $O(\log n)$. The running time of the algorithm is $O(m \log m(\beta + m \log\log m))$ time, where β is the running time of the* MINIMUM LINEAR ARRANGEMENT *solver.*

Here, the resulting set $\mathcal{P}$ is a collection of unit chains corresponding to the set $\{1,\ldots,n\}$. The algorithm for the packing problem in [16], which is a generalization of that in [89], can also been applied here for the case of a weak block solver. The number of iteration of their algorithm is bounded by $O(\rho \log m \log \rho)$, where $\rho = \max_{e\in E} \max_{x \in B} \sum_{i=1}^{N} x_i d_{P_i}(e)/l_G(e) = \max_{P_i \in \mathcal{P}} \max_{e \in E} d_{P_i}(e)/l_G(e)$ in this problem, which can be unbounded.

3.2.4 Approximation algorithms for the range assignment problem in static ad-hoc networks

We apply the algorithm to approximate an arbitrary graph by a collection of paths to develop an approximation algorithm for the MINIMUM RANGE ASSIGNMENT

Algorithm $\mathcal{RA}(G(S, E, l_G), \alpha, \Pi_h)$:

1. Construct a complete graph M_G for the given network G, such that for any pair u, $v \in S$, the weight of the edge (u, v) in M_G is $l_{M_G}(u, v) = (l_G(u, v))^\alpha$.
2. Run **Algorithm** $\mathcal{AG}(M_G(S, E_{M_G}, l_{M_G}), \mathcal{P}, x, \varepsilon)$ in Section 3.2.3 to generate a collection $\mathcal{P}$ of paths with a probability distribution μ over it; the generated paths can be represented in one-dimensional (Euclidean) lines such that neighbours are in (Euclidean) distance one.
3. Run algorithms for the Minimum Range Assignment problems (Min-Range(h-SC) and Min-Range(h-B)) for static ad-hoc networks on one dimensional (Euclidean) spaces for the reduced instances $(P(S_P, E_P, 1), 1, \Pi_h)$, for any $P \in \mathcal{P}$ chosen according to the probability distribution μ.

Table 3.2: Approximation Algorithm $\mathcal{RA}$ for the Minimum Range Assignment problem.

problem in static ad-hoc networks, for both Min-Range(h-B) and Min-Range(h-SC) even in case that transmission distances violate triangle inequality.

For any fixed h, let $(G(S, E, l_G), \alpha, \Pi_h)$ be an instance for the Minimum Range Assignment problem problem (Min-Range(h-SC) or Min-Range(h-B)) in static ad-hoc network with arbitrary structure, where S is the station set, l_G is the transmission distance in the complete graph G, α is the distance-power gradient, and Π_h is the property of the communication graph (h-strong connectivity or h-broadcast). Our approximation algorithm is in Table 3.2.

Hence, we obtain the following theorem for the approximation algorithm $\mathcal{RA}$:

Theorem 3.3 *There exists a probabilistic $O(\log n)$-approximation algorithm for* Min-Range(h-SC) *and* Min-Range(h-B) *in general static ad-hoc networks running in at most $O(n^2 \log n(\beta + n^2 \log\log n) + hn^4)$ time, where β is the running time of the* Minimum Linear Arrangement *solver.*

Proof: We first show the approximation ratio of the algorithm $\mathcal{RA}$. In the last step, a given instance $(P(S_P, E_P, 1), 1, \Pi_h)$ is defined on one dimensional (Euclidean) space. Therefore, the algorithms in [2, 23, 74] deliver the optimal solution for the Min-Range(h-B) problem and a 2-approximate solution for the Min-Range(h-SC) problem can be generated by the algorithm in [21]. Besides, in the second step, according to the analysis in Section 3.2.3, the expected distortion of any edge

in $\mathcal{P}$ is at most $O(\log n)$. Therefore combining the second and the third steps of Algorithm $\mathcal{RA}$, we are able to obtain an $O(\log n)$-approximate solution for an instance $(M_G, 1, \Pi_h)$ of the MINIMUM RANGE ASSIGNMENT problem for the network defined on M_G and $\alpha = 1$. We notice that after the first step, in the resulting complete graph M_G, the cost for station u to *directly* cover v is $cost_{M_G}(u, v) = c(u)l_{M_G}(u, v) = c(u)(d_G(u, v))^\alpha = cost_G(u, v)$. Therefore the solution of Algorithm $\mathcal{RA}$ has an overall cost with an expected factor of at most $O(\log n)$ times of the optimal overall cost of the original instance $(G(S, E, l_G), \alpha, \Pi_h)$ of the MINIMUM RANGE ASSIGNMENT problem.

To solve the packing problem in the second step, the running time is bounded by $O(n^2 \log n(\beta + n^2 \log \log n))$ time is needed, where β is the running time of the MLA solver. In the first step to construct the complete graph M_G runs in $O(n^2)$ time. And the running time of the dynamic programming for the range assignment problem on one dimensional spaces in the last step is $O(hn^3)$ time for the MIN-RANGE(h-SC) problem and $O(hn^4)$ time for the MIN-RANGE(h-B) problem, because the paths in $\mathcal{P}$ are in fact unit chains [21, 2]. The proof is complete. □

However, here we can not directly approximate the original graph G by paths. Otherwise the approximation ratio will be $O(\log^\alpha n)$ because the cost is $cost_G(u, v) = c(u)(d_G(u, v))^\alpha$.

Remark: It is worth noting that the approximation ratio of Algorithm $\mathcal{RA}$ for the MIN-RANGE(h-B) problem is $O(\log n)$, while the lower bound for the MIN-RANGE(h-B) problem on metric spaces, where triangle inequality holds, is also $O(\log n)$ [103]. In addition, in [110] the approximation ratio of the algorithms for the MIN-RANGE(h-B) problem is $O(\log n \log \log n)$ while for the MIN-RANGE(h-SC) problem the ratio is as large as $(n^2 \log n \log \log n)$. Here we have successfully removed the large factor $O(n^2)$ and for both problems the ratios are the same.

3.2.5 Improved approximation ratio for a special case

We show in this section that the approximation ratio $O(\log n)$ can be further improved if the distance-power gradient α and the weighted graph which represents

Algorithm $\mathcal{RA}'(G(S,E,l_G),\alpha,\Pi_h,\Omega)$:

1. Run **Algorithm** $\mathcal{AG}(G(S,E,l_G),\mathcal{P},x,\varepsilon)$ in Section 3.2.3 to generate a collection $\mathcal{P}$ of paths with a probability distribution μ over it;
2. Run algorithms for the Minimum Range Assignment problems (Min-Range(h-SC) and Min-Range(h-B)) for static ad-hoc networks on one dimensional (Euclidean) spaces for the reduced instances $(P(S_P,E_P,1),1,\Pi_h)$, for any $P\in\mathcal{P}$ chosen according to the probability distribution.

Table 3.3: Approximation Algorithm $\mathcal{RA}$ for the Minimum Range Assignment problem in a special case.

the ad-hoc network fulfils certain property.

Let H and G be graphs. A graph H is a *minor* of a graph G if H can be obtained from G by deleting and contracting some edges of G. Denote by $K_{r,r}$ the $r\times r$ complete bipartite graph. A typical example of a graph with no $K_{r,r}$-minors ($r\geq 3$) is a planar graph. We define a property Ω as follows: An instance $(G(S,E,l),\alpha,\Pi_h,\Omega)$ of the Minimum Range Assignment problem in general ad-hoc networks has the property Ω if and only if

A. the graph G does not contain $K_{r,r}$-minors for any $r\geq 3$;

B. $\alpha\leq O(\log\log n/\log\log\log n)$.

For any instance $(G(S,E,l_G),\alpha,\Pi_h,\Omega)$ of the Minimum Range Assignment problem we have the two-step algorithm in Table 3.3.

Then we have the following theorem:

Theorem 3.4 *There exists a probabilistic $O((\log\log n)^\alpha)$-approximation algorithm for* Min-Range(h-SC) *and* Min-Range(h-B) *in general static ad-hoc networks when property Ω holds.*

Proof: In the first step, we still need to apply the packing problem and its algorithm used in Section 3.2.3. And an MLA solver is called to solve the block problem. If the graph does not contain $K_{r,r}$-minors, $r\geq 3$, there exists an $O(\log\log n)$-approximation algorithm for the Minimum Linear Arrangement problem [101]. Therefore for each edge in G, the expected distortion in the paths is bounded by

$O(\log\log n)$. Notice that here we do not construct the graph M_G but directly approximate the original graph G by paths. With the similar arguments as in Section 3.2.4, the approximation ratio for the range assignment problem is $O((\log\log n)^\alpha)$. However we have the following bound on the approximation ratio when the property Ω holds:

$$\begin{aligned}(\log\log n)^\alpha &\leq (\log\log n)^{\log\log n/\log\log\log n}\\ &= (\log\log n)^{O(\log_{\log\log n}\log n)}\\ &= = O(\log n).\end{aligned}$$

In this way the approximation ratio claimed in the theorem is proved. □

Remark: We claim that indeed the instances with property Ω are not rare. It is obvious that for a fixed value of α (which is usually in the interval $[1, 6]$ in practice), a large station set S can result in the second assumption of the property Ω. In fact, an instance on a planar graph with $\alpha = 2$, a set of $n = 16$ stations are sufficient for the property Ω. We believe that many real applications belong to this category.

3.2.6 Mobile ad-hoc networks

In general, a mobile ad-hoc network consists of mobile nodes that are connected via wireless links. The nodes are free to move randomly. Because of the mobility of these nodes, the network topologies can change frequently and even invalidate existing routes. More details and challenges in mobile ad-hoc networks are referred to [17].

Routing protocols have attracted much attention in order that the mobile ad-hoc networks can work well. Proactive routing protocols and reactive on-demand routing protocols are two typically categories [8] in this field. The main characteristic of proactive protocols is the constant maintaining of a route by each node to all other network nodes. The route creation and maintenance are performed through both periodic and event-driven. Reactive routing protocols discover a route between two nodes only when it is needed.

DSDV [88] (Destination-Sequence Distance-Vector) is a proactive routing protocol. Each node maintains a routing table that contains routing information of all the reachable destination nodes, such as the number of hops and the next-hop to destination. To keep the routing table up to date, nodes in the network periodically broadcast routing table updates or use triggered route updates when the topology changes. Therefore in each period (time step) the structure information of the network is available for all nodes.

By the DSDV protocol, it needs to compute a temporary ad-hoc networks based on the routing table. Thus, DSDV is the base to study a static ad-hoc networks and apply the results in static networks to the mobile one. Actually, the research of static ad-hoc networks is significant for the proactive protocols. Here we can also generalize our algorithms to the model of mobile ad-hoc networks (dynamical model) according to the DSDV protocol. Assuming that the time for radio signals to reach any node in the network is no more than the time step for updating the routing table, our algorithm $\mathcal{RA}$ or $\mathcal{RA}'$ lead to a $O(\min\{\log n, (\log\log n)^{\alpha}\})$-approximation algorithm for any instance of the Minimum Range Assignment problems in general mobile ad-hoc networks.

Chapter 4

Scheduling Malleable Tasks with Precedence Constraints

In this chapter we study the problem of Scheduling Malleable Tasks with Precedence Constraints. The best previous approximation algorithm (that works in two phases) by Lepére et al [81] has a ratio $3 + \sqrt{5} \approx 5.236$. In the first phase a discrete time-cost tradeoff problem is solved approximately, and in the second phase a variant of the list scheduling algorithm is used. In phase one a rounding parameter $\rho = 1/2$ and in phase two an allotment parameter $\mu = (3m - \sqrt{5m^2 - 4m})/2$ are employed, respectively. We study the influence of the rounding parameter ρ to the second phase. With setting $\rho \neq 1/2$ we are able to obtain a better approximation algorithm with a ratio of $(3 + \sqrt{5})/2 + \sqrt{2(\sqrt{5} + 1)} \approx 5.162$. Furthermore, we study the linear programming relaxation and rounding technique in the first phase more carefully. As a result we develop an improved approximation algorithm with a ratio of $100/43 + 100(\sqrt{4349} - 7)/2451 \approx 4.730598$. In addition, in a new realistic model of the malleable tasks, instead of solving the discrete time-cost tradeoff problem, we solve a piecewise linear program and use a new rounding technique. Thus we obtain an improved approximation algorithm with a ratio of $100/63 + 100(\sqrt{6469} + 13)/5481 \approx 3.291919$.

4.1 Introduction

In recent development of information technology, traditional super-computers have been gradually replaced by systems with large number of standard units. All units

have certain similar structure with processing ability [26]. To manage these resources efficient algorithms are needed. Unfortunately classical scheduling algorithms usually are not able to play this role, mostly due to the large influence of communications between units. There have been many models for this problem [25, 31, 52, 64, 91, 106]. Among them scheduling *malleable tasks* is an important and practical one, which was proposed in [106]. In this model, the processing time of a malleable task depends on the number of processors allotted to it. The influence of communications between processors allotted to the same task, synchronization and scheduling overhead is included in the processing time. The communication between malleable tasks is usually tiny and neglected.

In this chapter, we study the problem of Scheduling Malleable Tasks with Precedence Constraints. We assume that the malleable tasks are linked by *precedence constraints*, which are determined in advance by the data flow between tasks. Let $G = (V, E)$ be a directed graph, where $V = \{1, \ldots, n\}$ represents the set of malleable tasks, and $E \subseteq V \times V$ represents the set of precedence constraints among the tasks. If there is an arc $(i, j) \in E$, then task J_j can not be processed before the completion of processing of task J_i. The task J_i is called a *predecessor* of J_j, while J_j a *successor* of J_i. We denote by $\Gamma^-(j)$ and by $\Gamma^+(j)$ the set of the predecessors and the successors of J_j, respectively. In addition, the n precedence constrained malleable tasks can be processed on m given identical processors. Each task J_j can be processed on any integer number $l \in \{1, \ldots, m\}$ of processors, and the corresponding integer processing time is $p_j(l)$. The goal of the problem is to find a feasible schedule minimizing the makespan $C_{\max}$ (maximum completion time).

According to the usual behaviour of parallel tasks in practice, Blayo et al. [11] showed that the following *monotonous penalty assumptions* are realistic:

Assumption 4.1 *The processing time $p_j(l)$ of a malleable task J_j is non-increasing in the number l of the processors allotted to it, i.e.,*

$$p_j(l) \leq p_j(l'), \quad \text{for } l \geq l'; \tag{4.1}$$

Assumption 4.2 *The work $W_j(l) = w_j(p_j(l)) = lp_j(l)$ of a malleable task J is non-decreasing in the number l of the processors allotted to it, i.e.,*

$$W_j(l) \leq W_j(l'), \quad \textit{for } l \leq l'. \tag{4.2}$$

Assumption 4.1 indicates that more processing power is available with more processors allotted such that the malleable task can run faster. Furthermore, Assumption 4.2 implies that the increase of processors allotted leads to increasing amount of communication, synchronization and scheduling overhead.

Prasanna et al. [91, 92, 93] proposed another model of the malleable tasks. In their model, for each malleable task, the processing time is non-decreasing in the number of processors allotted. In addition, a *speedup* function $s_j(l)$ for a malleable task J_j that is defined as the processing time $p_j(1)$ on one processor divided by the processing time $p_j(l)$ on l processors is concave in l. Their model has already been applied to the very massively parallel MIT Alewife machine [1, 90]. However, their model allows non-integral numbers of processors. We proposed a discrete model based on two natural assumptions for processing time of malleable tasks. The first assumption is exactly Assumption 4.1 and the second assumption is as follows:

Assumption 4.3 *The speedup function $s_j(l) = p_j(1)/p_j(l)$ of a malleable task J_j is concave in the number l of the processors allotted to it, i.e., for any $0 \leq l'' \leq l \leq l' \leq m$,*

$$\frac{p_j(1)}{p_j(l)} = s_j(l) \geq \frac{1}{l'-l''}[(l-l'')s_j(l') - (l-l')s_j(l'')] = \frac{p_j(1)}{l'-l''}\left[\frac{l-l''}{p_j(l')} - \frac{l-l'}{p_j(l'')}\right]. \tag{4.3}$$

Here we assume that $p_j(0) = \infty$ as any task J_j can not be executed if there is no processor available. Assumption 4.3 also implies that the increase of processors allotted leads to increasing amount of communication, synchronization and scheduling overhead, such that the speedup effect can not be linear. A typical example is that the processing time $p(l) = p(1)l^{-d_j}$, where l is the number of processors and $0 < d_j < 1$ (similar to the continuous case in [91, 92, 93]). We can show that under Assumption 4.3 the work function fulfils Assumption 4.2 and is convex in processing time (see Section 4.5).

In a schedule each task J_j has two associated values: the starting time τ_j and the number of processors l_j allotted to task J_j. A task J_j is called active during the

time interval from its starting time τ_j to its completion time $C_j = \tau_j + p_j(l_j)$. A schedule is feasible if at any time t, the number of active processors does not exceed the total number of processors

$$\sum_{j:t\in[\tau_j,C_j]} l_j \leq m$$

and if the precedence constraints

$$\tau_i + p_i(l_i) \leq \tau_j,$$

are fulfilled for all $i \in \Gamma^-(j)$.

Related Works: The problem of scheduling independent malleable tasks (without precedence constraints) is strongly $\mathcal{NP}$-hard even for only 5 processors [29]. The previous best known algorithm for the problem of scheduling independent malleable tasks has an approximation ratio 2 [39, 83]. This was improved to $\sqrt{3}+\varepsilon$ by Mounié et al. [86], and further to $3/2+\varepsilon$ [87]. For the case of fixed m, Jansen and Porkolab proposed a PTAS [57]. If $p(l) \leq 1$, for arbitrary m an AFPTAS was addressed by Jansen [53].

Du and Leung [29] showed that Scheduling Malleable Tasks with Precedence Constraints is strongly $\mathcal{NP}$-hard for $m = 3$. Furthermore, there is no polynomial time algorithm with approximation ratio less than $4/3$, unless $\mathcal{P}$=$\mathcal{NP}$[79]. If the precedence graph is a tree, a $(4+\varepsilon)$-approximation algorithm was developed in [80]. The idea of the two-phase algorithms was proposed initially in [80] and further used in [81] to obtain the best previous known approximation algorithm for general precedence constraints with a ratio $3 + \sqrt{5} \approx 5.236$. In [81] the ratio was improved to $(3 + \sqrt{5})/2 \approx 2.618$ when the precedence graph is a tree. More details on the problem of scheduling independent or precedence constrained malleable tasks can be found in [31].

Our Contribution: We develop improved approximation algorithms for the problem of Scheduling Malleable Tasks with Precedence Constraints. Our first and second algorithms are for the model under Assumption 4.1 and 4.1, while the third algorithm is for the model under Assumption 4.1 and 4.3.

Our first algorithm is based on the two-phase approximation algorithm in [80, 81] for the model under Assumption 4.1 and 4.2. In the first phase, an *allotment problem* for malleable tasks is solved. The goal is to find an allotment $\alpha : V \to \{1, \ldots, m\}$ indicating the numbers of processors to execute the tasks such that the maximum of the overall critical path length and the average work (the total work divided by m) is minimized. The problem can be formulated as a bicriteria version of the time-cost tradeoff problem, which can be solved by the approximation algorithm in [104]. In the second phase a variant of the list scheduling algorithm is employed and a feasible schedule is generated. In the first and the second phase of the algorithm in [81], a rounding parameter $\rho = 1/2$ and an allotment parameter μ are used, respectively. We first borrow the idea of the algorithm in [104] to develop a linear program for the allotment problem and avoid the binary search procedure. Besides, we do not fix the rounding parameter $\rho = 1/2$ (as suggested in [81]) but introduce it as an unspecific parameter to the second phase. As a result we obtain a min-max nonlinear integer program where the objective value is an upper bound of the approximation ratio. Solving it we obtain $\rho = 1/[1+\sqrt{m/(m-\mu)}]$ and $\mu = (3m - \sqrt{5m^2 - 4m})/2$, which yield an approximation ratio of $(3+\sqrt{5})/2 + \sqrt{2(\sqrt{5}+1)} \approx 5.162$.

Furthermore, we we carefully study the rounding technique. We notice that a certain amount of work is not involved in the rounding procedure. By counting this term carefully, we obtain some new bounds depending on ρ for the total work of the rounded solution. Together with the rounding parameter μ employed in the second phase, we develop another min-max nonlinear program, whose optimal objective value is an upper bound on the approximation ratio by choosing appropriate values of ρ and μ. Next we analyze the nonlinear program to obtain an optimum values for the parameters ρ and μ. Using $\rho = 0.43$ and $\mu = (93m - \sqrt{4349m^2 - 4300m})/100$ we obtain an improved approximation algorithm with a ratio at most $100/43 + 100(\sqrt{4349} - 7)/2451 \approx 4.730598$. In addition, we show that asymptotically the best ratio is 4.730577 when $m \to \infty$. This indicates that our choice is very close to the best possibility.

Finally, we study our new model under Assumption 4.1 and 4.3. We show that in this model, the work function is non-decreasing in the number of processors and

is convex in the processing time. The first property is indeed the Assumption 4.2 on work function in the old model in [81]. Then we develop a new approximation algorithm for this new model. Our algorithm is also a two-phase algorithm. In the first phase we we do not apply the strategy of reducing the allotment problem to the discrete time-cost tradeoff problem. We just construct a piecewise linear work function according to the discrete values of works and processing times. With respect to the precedence constraints we are able to develop a piecewise linear program. Furthermore, since the work function is convex in the processing time, we are able to formulate the piecewise linear program as a linear program. We include also some additional constraints to avoid the binary search. Next we apply a new rounding technique for the (fractional) optimal solution to the linear program. The rounded solution yields a feasible solution of the allotment problem with a good approximation ratio depending on our rounding parameter $\rho \in [0, 1]$. In the second phase the variant of the list scheduling algorithm is employed to generate a new allotment and to schedule all tasks according to the precedence constraints. By studying the structure of the resulting schedule, we show that the approximation ratio is bounded by the optimal objective value of a min-max nonlinear program. Exploiting the solution to the min-max nonlinear program, we prove that the approximation ratio of our algorithm is not more than $100/63 + 100(\sqrt{6469} + 13)/5481 \approx 3.291919$. This ratio is much better than all previous results. We also study the asymptotic behaviour of the solution to the min-max nonlinear program and show that the asymptotic best ratio is 3.291913.

4.1.1 Discrete time-cost tradeoff problem

In [104] the discrete *time-cost tradeoff problem* is studied. An instance of the discrete time-cost tradeoff problem is a *project* given by a finite set J of *activities* with a *partial order* $(J, \prec)$ on the set of activities. All activities have to be executed in accordance with the precedence constraints given by the partial order. Each activity $J_j \in J$ has a set of feasible *durations* $\{d_{j_1}, \ldots, d_{j_{k(j)}}\}$ sorted in a non-decreasing order, and has a non-increasing non-negative *cost* function $c_j : \mathbb{R}_+ \to \mathbb{R}_+ \cup \infty$, where $c_j(x_j)$ is the amount paid to run J_j with duration x_j.

There are three related optimization problems in this class to minimize either

time or cost for the project by fixing the other parameter, or to find approximate values of both. The *budget problem* is for a given budget $B \geq 0$ to find a solution x satisfying $c(x) \leq B$, where $c(x)$ is the total cost of the solution, such that the project length (the makespan of the corresponding scheduling problem with unbounded number of processors) is minimized. The second problem is the *deadline problem*: for a given project duration $L \geq 0$, to find a solution x satisfying $C_{\max}(x) \leq L$, where $C_{\max}(x)$ is the makespan of the solution x, such that the total cost is minimized. The bicriteria problem is, given a budget B and a deadline L, to find a solution x such that $t(x) \leq \kappa L$ and $c(x) \leq \lambda B$ for given parameters $\kappa, \lambda \geq 1$.

Skutella [104] presented approximation algorithms for above problems, in particular, an algorithm for the bicriteria problem such that $c(x) < B/(1-\rho)$ and $t(x) \leq L/\rho$ for a fixed $\rho \in (0,1)$. The algorithm is outlined as follows:

The budget problem can be described as the following integer linear program:

$$\begin{array}{lll} \min & L & \\ \text{s.t.} & C_j \leq L, & \text{for all } j; \\ & C_i + x_j \leq C_j, & \text{for all } i \in \Gamma^-(j) \text{ and all } j; \\ & c(x) = \sum_{j=1}^{n} c_j(x_j) \leq B; & \\ & x_j \in \{d_{j_1}, \ldots, d_{j_{k(j)}}\}, & \text{for all } j. \end{array} \tag{4.4}$$

Here C_j is the completion time of activity J_j, $c_j(x_j)$ the cost of the duration x_j and d_{j_p} the p-th feasible duration of activity J_j. The first set of constraints shows that completion times of any tasks are bounded by the project length. The second set is related to the precedence constraints. In addition, the third set of constraint means that the total cost should be bounded by the budget B.

To solve it, first a "reduced" cost function $\hat{c}$ is set such that for any duration d_{j_l}, $\hat{c}_j(d_{j_l}) = c_j(d_{j_l}) - c_j(d_{j_{k(j)}})$. Since $d_{j_{k(j)}}$ is the maximum duration for activity J_j, $c_j(d_{j_{k(j)}})$ is the minimum over all durations for activity J_j and the "reduced" cost $\hat{c}$ is also positive. The amount of $P = \sum_{j=1}^{n} c_j(d_{j_{k(j)}})$ is called "fixed" cost. Therefore we just turn to consider the following integer linear program equivalent to (4.4):

$$
\begin{array}{lll}
\min & L & \\
\text{s.t.} & C_j \leq L, & \text{for all } j; \\
& C_i + x_j \leq C_j, & \text{for all } i \in \Gamma^-(j) \text{ and all } j; \\
& \hat{c}(x) + P = \sum_{j=1}^{n} \hat{c}_j(x_j) + P \leq B; & \\
& x_j \in \{d_{j_1}, \ldots, d_{j_{k(j)}}\}, & \text{for all } j.
\end{array} \tag{4.5}
$$

In the second step, the "reduced" instance is transformed to a two-duration instance such that each given "virtual" activity has only at most two feasible durations. For any activity J_j, we introduce the first "virtual" activity J_{j_1} as follows: J_{j_1} has only two fixed feasible durations $s_j(1) = t_j(1) = d_{j_1}$, and the corresponding "virtual" costs are $\bar{c}_j(s_j(1)) = \bar{c}_j(t_j(1)) = 0$. Then for each $1 < i \leq k(j)$ a "virtual" activity J_{j_i} is introduced. Each "virtual" activity J_{j_i} has a duration x_{j_i} chosen from only two feasible "virtual" durations $s_j(i) = 0$ and $t_j(i) = d_{j_i}$, and the corresponding "virtual" costs are $\bar{c}_{j_i}(s_j(i)) = \hat{c}_j(d_{j_{i-1}}) - \hat{c}_j(d_{j_i})$ and $\bar{c}_{j_i}(t_j(i)) = 0$. It is easy to verify that for each activity J_j, the sum of corresponding "virtual" costs equals to the "reduced" cost. Thus the activity J_j can be modelled as $k(j)$ number of parallel "virtual" activities, which are represented by parallel edges in the edge diagram. Then there is a canonical mapping of feasible durations x_j for activity J_j to tuples of feasible durations $x_{j_1}, \ldots, x_{j_{k(j)}}$ for "virtual" activities $J_{j_1}, \ldots, J_{j_{k(j)}}$ such that the duration of the activity J_j is the maximum over all durations of the corresponding "virtual" activities and the cost of J_j is the sum of the costs of the "virtual" activities, i.e., $x_j = \max\{x_{j_1}, \ldots, x_{j_{k(j)}}\}$ and $\hat{c}_j(x_j) = \sum_{i=1}^{k(j)} \bar{c}_{j_i}(x_{j_i})$. Moreover, this mapping is bijective if we restrict ourselves without loss of generality to tuples of durations $x_{j_1}, \ldots, x_{j_{k(j)}}$ satisfying $x_{j_i} = t_j(i)$ if $t_j(i) \leq \max\{x_{j_1}, \ldots, x_{j_{k(j)}}\}$. In this way we have obtained a two-duration instance such that each activity has at most two feasible durations, and the solution of this instance can be transformed to a solution to the "reduced" instance (4.5).

Finally, we consider the linear relaxation of the two-duration instance, where for each "virtual"activity J_{j_i}, the "virtual" cost function is linear and non-increasing within the interval $[s_j(i), t_j(i)]$ as follows:

$$\bar{c}_{j_i}(y) = \begin{cases} \infty, & \text{if } y < s_j(i); \\ \dfrac{t_j(i)-y}{t_j(i)-s_j(i)}\bar{c}_{j_i}(s_j(i)) + \dfrac{y-s_j(i)}{t_j(i)-s_j(i)}\bar{c}_{j_i}(t_j(i)), & \text{if } s_j(i) \le y \le t_j(i); \\ \bar{c}_{j_i}(t_j(i)) = 0, & \text{if } y \ge t_j(i). \end{cases} \tag{4.6}$$

The linear relaxation can be solved by algorithms in [70, 37]. Therefore in this way we are able to find a fractional solution of the linear relaxation of the two-duration instance. The corresponding linear relaxation of (4.5) is as follows:

$$\begin{array}{lll} \min & L & \\ \text{s.t.} & C_j \le L, & \text{for all } j; \\ & C_i + x_j \le C_j, & \text{for all } j \text{ s.t. } \Gamma^-(j) \ne \emptyset \text{ and all } i \in \Gamma^-(j); \\ & x_j \le C_j, & \text{for all } j \text{ s.t. } \Gamma^-(j) = \emptyset; \\ & x_j \le d_{j_{k(j)}}, & \text{for all } j; \\ & x_{j_i} \le x_j, & \text{for all } j \text{ and } i = 1, \ldots, k(j); \\ & 0 \le x_{j_i}, & \text{for all } j \text{ and } i = 2, \ldots, k(j); \\ & x_{j_i} \le d_{j_i}, & \text{for all } j \text{ and } i = 2, \ldots, k(j); \\ & x_{j_1} = d_{j_1}, & \text{for all } j; \\ & \bar{c}_{j_i}(x_{j_i}) = [c_j(d_{j_{i-1}}) - c_j(d_{j_i})]\dfrac{d_{j_i} - x_{j_i}}{d_{j_i}}, & \text{for all } j \text{ and } i = 2, \ldots, k(j); \\ & \bar{c}_{j_1}(x_{j_1}) = 0, & \text{for all } j; \\ & \hat{c}_j(x_j) = \sum_{i=1}^{k(j)} \bar{c}_{j_i}(x_{j_i}), & \text{for all } j; \\ & \sum_{j=1}^{n} \hat{c}_j(x_j) + P \le B. & \end{array} \tag{4.7}$$

To obtain a feasible (integer) solution of (4.5), we need to take some rounding technique. For a given $\rho \in (0,1)$, if for a "virtual" activity J_{j_i} the fractional solution $x_{j_i} \in [0, \rho d_{j_{k(j)}})$, we round it to $x_{j_i} = 0$. Otherwise if $x_{j_i} \in [\rho d_{j_{k(j)}}, d_{j_{k(j)}}]$ we round it to $x_{j_i} = d_{j_{k(j)}}$. Then the following proposition holds [104]:

Proposition 4.1 *For a given deadline L (or budget $B - P$) and a fixed rounding parameter $\rho \in (0,1)$, there exists an algorithm that computes a realization x for (4.5) such that $\hat{c}(x) \le (B-P)/(1-\rho)$ and $C_{\max} \le L/\rho$, where $B-P$ is the optimal*

cost for the linear relaxation of (4.5) with a deadline L (respectively T is the optimal project length for the linear relaxation of (4.5) with a budget $B - P$).

Proof: We consider two cases. In the first case the solution $x_{j_i} \in [0, \rho d_{j_{k(j)}})$ and it is rounded to $\bar{x}_{j_i} = 0$. In this case the duration does not increase but the cost increases. We substitute $s_j(i) = 0$, $t_j(i) = d_{j_{k(j)}}$ and $c_{j_i}(t_j(i)) = 0$ to (4.6), which leads to $\bar{c}_{j_i}(x_{j_i}) = (d_{j_{k(j)}} - x_{j_i})c_{j_i}(0)/d_{j_{k(j)}}$. Since $x_{j_i} \leq \rho d_{j_{k(j)}}$, we immediately have that $c_{j_i}(\bar{x}_{j_i})/\bar{c}_{j_i}(x_{j_i}) = c_{j_i}(0)/\bar{c}_{j_i}(x_{j_i}) \leq 1/(1-\rho)$. In the second case the solution $x_{j_i} \in [\rho d_{j_{k(j)}}, d_{j_{k(j)}}]$ and it is rounded to $\bar{x}_{j_i} = d_{j_{k(j)}}$. The cost does not increase while the duration increases in this case. It is obvious that $\bar{x}_{j_i}/x_{j_i} = d_{j_{k(j)}}/x_{j_i} \leq 1/\rho$. With the approach to convert a solution of the two-duration instance to a solution of the "reduced" instance (4.5), we conclude that the rounded solution can increase the overall project length by a factor at most $1/\rho$ and increase the overall cost without "fixed" cost by a factor at most $1/(1-\rho)$. The proposition is proved. □

4.2 Structure of approximation algorithms

Our algorithms are based on the two-phase approximation algorithm in [81]. In their algorithm, in the first phase a $(2, 2)$-approximate solution to the bicriteria version of the discrete time-cost tradeoff problem with a guessed budget B and $\rho = 1/2$ was computed. Intuitively the choice $\rho = 1/2$ is a good strategy, since it leads to a $(2, 2)$-approximate solution for both objective functions (the critical path length and average work of the solution). But our analysis shows that it is better not to keep this balance. In fact, we keep ρ as an unspecified parameter in the second phase in order to enlarge the set of feasible solution and to find a better solution. Furthermore, the discrete time-cost tradeoff problem is equivalent to the scheduling problem with an unbounded number of processors and a bounded total work. However, in the solution of the algorithm in [104] the case of small critical path length with large total work can happen as there is no bound on the number of processors available. Thus a binary search procedure is conducted in [81] to find a fractional solution such that the maximum of the overall critical path length and the average work is minimized.

We present three algorithms. Algorithm I and II are for the model under Assumption 4.1 and 4.2, while Algorithm III is for the model under Assumption 4.3. All of our three algorithms are of similar structure. The difference lies on the approaches applied in the first phase and the values of two parameters used in our algorithms. Here we outline our algorithms as follows:

Given any instance of the problem of Scheduling Malleable Tasks with Precedence Constraints, we compute a real number $\rho \in (0, 1)$ and an integer $\mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}$. Here ρ is the *rounding parameter* which is used for the rounding stage in the first phase and μ is the *allotment parameter* which is the upper bound on the number of processors allotted to each task in the final schedule. They are used in the two phases of our algorithms, respectively. The choices of the values of above parameters either depend on the given number m of processors or are fixed in the algorithms. See Section 4.3, 4.4 and 4.5 for details.

In the first phase of our algorithms, we solve an allotment problem for the malleable tasks with precedence constraints. For any given malleable task J_j, we need to decide a number l'_j of processors allotted to it such that the maximum of overall critical path length and average work is minimized. To reach the goal, in our Algorithm I and II we borrow the idea of approximation algorithms for discrete time-cost tradeoff problem in [104]. In order to avoid the case of very small critical path length and very large total work, we include two additional constraints to the linear programming relaxation for the two-duration instances (4.7). In this case the binary search procedure is avoided. Then the rounding parameter ρ is used to obtain a feasible solution from a fractional solution to (4.4). Different from our Algorithm I and II, in our Algorithm III for our new model, we do not use the idea of solving the discrete time-cost tradeoff problem. Instead, we directly solve a piecewise linear program with a convex constraint function. Due to properties of the malleable tasks in this model, we are able to transform the piecewise linear program to a linear program. Then we round the fractional solution to the linear program according to a new rounding technique with a different rounding parameter ρ. Finally we obtain a feasible allotment α'.

In the second phase, with the computed allotment α' and the pre-computed allotment parameter μ, the algorithm generates a new allotment α and runs a variant

LIST (J, m, α', μ)
initialization: allot $l_j = \min\{l'_j, \mu\}$ processors to task J_j, for $j \in \{1, \ldots, n\}$;
$SCHEDULED = \emptyset$;
if $SCHEDULED \neq J$ **then**
  $READY = \{J_j | \Gamma^-(j) \subseteq SCHEDULED\}$;
  compute the earliest possible starting time under α for all tasks in $READY$;
  schedule the task $J_j \in READY$ with the smallest earliest starting time;
  $SCHEDULED = SCHEDULED \cup \{J_j\}$;
end

Table 4.1: Algorithm **LIST**

of the list scheduling algorithm in Table 4.1 (as proposed in [44, 81]).

4.3 Approximation algorithm I

In the first phase of our Algorithm I, as indicated in Section 4.2, we use the idea of the approximation algorithm for the discrete time-cost tradeoff problem in [104] to obtain a feasible solution to the allotment problem. For a given instance with n malleable tasks, each task J_j is associated with a processing time function $p_j(l)$ and work function $W_j(l) = lp_j(l)$ for $l \in \{1, \ldots, m\}$ number of processors allotted. We also denote by $w(\cdot)$ the work function in processing time, i.e., $w_j(p_j(l)) = W_j(l)$. Thus for task J_j we define its corresponding cost as its work function in processing time, i.e., $c_j(d_{j_l}) = w_j(p_j(m+1-l)) = W_j(m+1-l) = (m+1-l)p_j(m+1-l)$, for $l = 1, \ldots, m$. Since $c_j(d_{j_m}) = W_j(1)$ is the minimum work for task J_j, it is obvious that the "fixed" cost $P = \sum_{j=1}^{n} W_j(1) = \sum_{j=1}^{n} p_j(1)$, and the corresponding "reduced" work function is $\hat{w}_j(d_{j_l}) = \hat{c}_j(d_{j_l}) = W_j(m+1-l) - W_j(1)$ for $l = 1, \ldots, m-1$. In the way described in Subsection 4.1.1 or [104], we can formulate the two-duration instance, and also its linear relaxation. Denote by $\bar{w}(\cdot) = \bar{c}(\cdot)$. For a "virtual" activity J_{j_i} in the two-duration instance, its durations are $s_j(i) = 0$, $t_j(i) = d_{j_i} = p_j(m+1-i)$, and the corresponding "virtual" works are $\bar{w}_{j_i}(s_j(i)) = \bar{c}_{j_i}(s_j(i)) = \hat{c}_j(d_{j_{i-1}}) - \hat{c}_j(d_{j_i}) = \hat{w}_j(d_{j_{i-1}}) - \hat{w}_j(d_{j_i}) = \hat{w}_j(p_j(m-i+2)) - \hat{w}_j(p_j(m-i+1)) = w_j(p_j(m-i+2)) - w_j(p_j(m-i+1)) = W_j(m-i+2) - W_j(m-i+1)$,

$\bar{w}_{j_i}(t_j(i)) = \bar{c}_{j_i}(t_j(i)) = 0$. By substituting them to (4.6), the continuous "virtual" work function in the fractional processing time of the task in the two-duration instance is as follows:

$$\bar{w}_{j_i}(x_{j_i}) = \begin{cases} \dfrac{p_j(m-i+1) - x_{j_i}}{p_j(m-i+1)}[W_j(m-i+2) \\ -W_j(m-i+1)], & \text{for all } j \text{ and } i = 2, \ldots, m; \\ 0, & \text{for all } j \text{ and } i = 1. \end{cases} \tag{4.8}$$

Furthermore, we apply two additional constraints to bound the total work in order to avoid the binary search procedure in [81]. In any schedule, any critical path length should be bounded by the makespan. In addition, the makespan is an upper bound on the average work (total work divided by m). Denote by x_j the fractional duration of task J_j in the allotment problem. The corresponding linear program is as follows:

$$
\begin{array}{lll}
\min & C & \\
\text{s.t.} & C_j \le L, & \text{for all } j; \\
& C_i + x_j \le C_j, & \text{for all } j \text{ s.t. } \Gamma^-(j) \neq \emptyset \text{ and all } i \in \Gamma^-(j); \\
& x_j \le C_j, & \text{for all } j \text{ s.t. } \Gamma^-(j) = \emptyset; \\
& x_j \le p_j(1), & \text{for all } j; \\
& x_{j_i} \le x_j, & \text{for all } j \text{ and } i = 1, \ldots, m; \\
& 0 \le x_{j_i}, & \text{for all } j \text{ and } i = 2, \ldots, m; \\
& x_{j_i} \le p_j(m-i+1), & \text{for all } j \text{ and } i = 2, \ldots, m; \\
& x_{j_1} = p_j(m), & \text{for all } j; \\
& \bar{w}_{j_i}(x_{j_i}) = [(m-i+2)p_j(m-i+2) - (m & \\
& -i+1)p_j(m-i+1)]\dfrac{p_j(m-i+1) - x_{j_i}}{p_j(m-i+1)}, & \text{for all } j \text{ and } i = 2, \ldots, m; \\
& \bar{w}_{j_1}(x_{j_1}) = 0, & \text{for all } j; \\
& \hat{w}_j(x_j) = \sum_{i=1}^{m} \bar{w}_{j_i}(x_{j_i}), & \text{for all } j; \\
& \sum_{j=1}^{n} \hat{w}_j(x_j) + P \le W; & \\
& P = \sum_{j=1}^{n} p_j(1); & \\
& L \le C; & \\
& W/m \le C. &
\end{array}
\tag{4.9}
$$

Solving (4.9) we are able to obtain a (fractional) optimal solution x_j^* for each task J_j. Then we apply the rounding technique in [104] for the fractional solution to (4.9). With the rounded solution of the processing time in $\{p_j(m), \ldots, p_j(1)\}$ we are able to identify an l'_j such that $p_j(l'_j)$ equals to the rounded solution. Then we develop an allotment α' for all jobs where each job J_j is allotted a number l'_j processors.

In our Algorithm I, for a given processor number m, the allotment parameter μ is determined as follows:

$$
\mu = \mu^* = \left\lceil \frac{3m - \sqrt{5m^2 - 4m}}{2} \right\rceil \text{ or } \left\lfloor \frac{3m - \sqrt{5m^2 - 4m}}{2} \right\rfloor. \tag{4.10}
$$

We compute and keep both above integers. With the computed allotment parameter μ, the rounding parameter ρ can be determined as follows:

$$\rho = \rho^* = \frac{1}{1 + \sqrt{\frac{m}{m - \mu^*}}}. \tag{4.11}$$

In practice, we should use both two values of μ to compute the two corresponding values of ρ and then approximation ratio r (which is defined later). Then we compare the two calculated r's and choose the smaller one, which is the overall approximation ratio. Then we fix the corresponding values of μ and ρ as the parameters used in the initialization step of our Algorithm I.

We are going to analyze the performance of Algorithm I. Denote by L, W, $C_{\max}$ and L', W', $C'_{\max}$ the critical path lengths, the total works and the makespans of the final schedule delivered by our algorithm and the schedule corresponding to the allotment α' generated in the first phase, respectively. Furthermore, we denote by $C^*_{\max}$ the optimal objective value of (4.9), and L^*, W^* the (fractional) optimal critical path length and the (fractional) optimal total work in (4.9). It is worth noting that here $W^* = P + \sum_{j=1}^{n} \hat{w}_j(x^*_j)$ and $W' = \sum_{j=1}^{n} l'_j p_j(l'_j)$. According to Proposition 4.1, in the rounding strategy in [104], the critical path length and the "reduced" cost increase by at most $1/\rho$ and $1/(1-\rho)$, respectively, i.e., $L' \leq L^*/\rho$ and $(W' - P) \leq (W^* - P)/(1-\rho)$. Denote by OPT the overall optimal makespan (over all feasible schedules with integral number of processors allotted to all tasks). It is obvious that

$$\max\{L^*, W^*/m\} \leq C^*_{\max} \leq OPT. \tag{4.12}$$

We have the following bound on the total work W:

Lemma 4.1 $W \leq \dfrac{m}{1-\rho} C^*_{\max}$.

Proof: In the first phase, according to Proposition 4.1 in [104], we have $W' - P \leq (W^* - P)/(1-\rho)$. Thus

$$W' \leq \frac{W^*}{1-\rho} - \frac{\rho P}{1-\rho} \leq \frac{W^*}{1-\rho}.$$

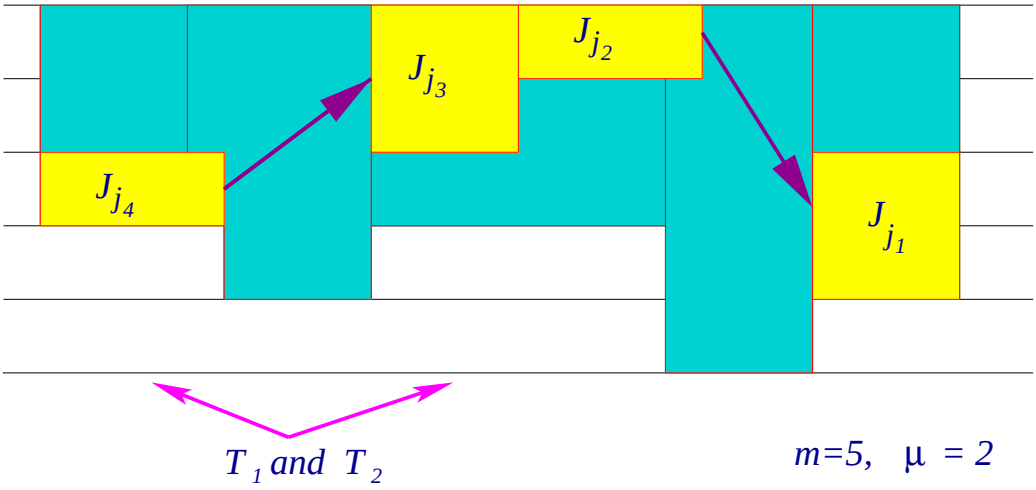


Figure 4.1: An example of the "heavy" path.

In the second phase, for each task J_j, the number of processors allotted to it $l_j \leq l'_j$. According to Assumption 4.2, the work is non-increasing, i.e., $W_j(l'_j) \geq W_j(l_j)$. Together with (4.12), we immediately obtain

$$W \leq W' \leq \frac{m}{1-\rho} C^*_{\max}.$$

□

The time interval $[0, C_{\max}]$ consists of three types of time slots. In the first type of time slots, at most $\mu - 1$ processors are busy. In the second type of time slots, at least μ while at most $m - \mu$ processors are busy. In the third type at least $m - \mu + 1$ processors are busy. Denote the sets of the three types time slots by T_1, T_2 and T_3, and $|T_i|$ the overall lengths for $i \in \{1, 2, 3\}$. In the case that $\mu = (m+1)/2$ for m odd, $T_2 = \emptyset$. In other cases all three types of time slots may exist. Same as in [81] we have following two lemmas:

Lemma 4.2 $|T_1| + \frac{\mu}{m}|T_2| \leq L'$.

Proof: The main idea is to construct a "heavy" directed path $\mathcal{P}$ in the transitive closure of the graph $G = (V, E)$. The last task in the path $\mathcal{P}$ is any multiprocessor task J_{j_1} that completes at time $C_{\max}$ (the makespan of the final schedule). After we have defined the last $i \geq 1$ tasks $J_{j_i} \to J_{j_{i-1}} \to \cdots \to J_{j_2} \to J_{j_1}$ on the path $\mathcal{P}$, we can determine the next task $J_{j_{i+1}}$ as follows: Consider the latest time slot t in

$T_1 \cup T_2$ that is before the starting time of task J_{j_i} in the final schedule. Let V' be the set of task J_{j_i} and its predecessor tasks that start after time t in the schedule. Since during time slot t at most $m - \mu$ processors are busy, and since at most μ processors are allotted to any task in V', all the tasks in V' can not be ready for execution during the time slot t. Therefore for every task in V' some predecessor is being executed during the time slot t. Then we select any predecessor of task J_{j_i} that is running during slot t as the next task $J_{j_{i+1}}$ on the path $\mathcal{P}$. This search procedure stops when $\mathcal{P}$ contains a task that starts before any time slot in $T_1 \cup T_2$. An example of the "heavy" path is illustrated in Figure 4.1.

Consider a task J_j on the resulting path $\mathcal{P}$. If in the allotment α of the second phase the task J_j is allotted $l_j < \mu$ processors, then α' and α both allot the same number of processors to J_j. In this case the processing times of J_j in α' and α are identical. In the final schedule such a task is processed during any time slot in $T_1 \cup T_2$. If in α exact $l_j = \mu$ number of processors are allotted to task J_j, then in α' there may be l'_j processors allotted to J_j for $\mu \leq l'_j \leq m$. With Assumption 4.2, the work $\mu \cdot p_j(\mu)$ in α is not more than the work $l'_j p_j(l'_j)$ in α'. Therefore $p_j(l'_j)$ is at least $\mu/l'_j \geq \mu/m$ times the processing time $p_j(\mu)$ in allotment α. In the final schedule such a task can be processed during any time slot in T_2 but not in T_1.

With the construction of the direct path $\mathcal{P}$, it covers all time slots in $T_1 \cup T_2$ in the final schedule. In the schedule after the first phase, the tasks processed in T_1 in the final schedule contribute a total length of at least $|T_1|$ to $L'(\mathcal{P})$. In addition, the tasks processed in T_2 contribute a total length of at least $|T_2|\mu/m$ to $L'(\mathcal{P})$. Since $L'(\mathcal{P})$ is not more than the length L' of the critical path in α', we have obtained the claimed inequality. □

Lemma 4.3 $(m - \mu + 1)C_{\max} \leq W + (m - \mu)|T_1| + (m - 2\mu + 1)|T_2|$.

Proof: According to the definitions, all time slots in T_1, T_2 and T_3 in the final schedule cover the whole interval $[0, C_{\max}]$. Therefore

$$C_{\max} = |T_1| + |T_2| + |T_3|. \tag{4.13}$$

In addition, as during the time slots of the first (respectively the second and the third) type at least one (respectively μ and $m - \mu + 1$) processors are busy, a lower

bound on the total work in the final schedule is:

$$W \geq |T_1| + \mu|T_2| + (m - \mu + 1)|T_3|. \tag{4.14}$$

Multiplying (4.13) by $m-\mu+1$ and subtracting (4.14) from it the proof is completed. □

Define the normalized overall length of i-th type of time slots by $x_i = |T_i|/C^*_{\max}$ for $i = 1, 2, 3$. The following lemma holds:

Lemma 4.4 *The Algorithm I has an approximation ratio*

$$r \leq \frac{\frac{m}{1-\rho} + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1}.$$

Proof: From Lemma 4.3 and Lemma 4.1 we have

$$(m-\mu+1)C_{\max} \leq \frac{m}{1-\rho}C^*_{\max} + (m-\mu)|T_1| + (m-2\mu+1)|T_2|.$$

Recall the definition of approximation ratio and (4.12),

$$r = \sup \frac{C_{\max}}{OPT} \leq \sup \frac{C_{\max}}{C^*_{\max}}.$$

From definitions of x_i, we can immediately obtain the claimed bound on r since $m - \mu + 1 > 0$. □

Lemma 4.5 *The normalized time slot lengths are bounded by* $x_1 + \frac{\mu}{m}x_2 \leq \frac{1}{\rho}$.

Proof: According to Proposition 4.1 and (4.12), $L' \leq L^*/\rho \leq C^*_{\max}/\rho$. Together with Lemma 4.2 we have

$$|T_1| + \frac{\mu}{m}|T_2| \leq \frac{C^*_{\max}}{\rho}.$$

Dividing both sides by $C^*_{\max}$ yields the lemma together with the definitions of x_i. □

Lemma 4.6 *The approximation ratio r of Algorithm I is bounded by the optimal value of the following min-max nonlinear program:*

$$\begin{array}{rl} \min_{\mu,\rho} \max_{x_1,x_2} & \dfrac{m/(1-\rho)+(m-\mu)x_1+(m-2\mu+1)x_2}{m-\mu+1} \\ s.t. & x_1+\dfrac{\mu}{m}x_2 \le \dfrac{1}{\rho}; \\ & x_1, x_2 \ge 0; \\ & \rho \in (0,1); \\ & \mu \in \left\{1,\ldots,\left\lfloor \dfrac{m+1}{2} \right\rfloor\right\}. \end{array} \tag{4.15}$$

Proof: The positive variables x_1 and x_2 can be any feasible real numbers bounded by Lemma 4.5 because in the final schedule the total lengths of time slots in T_1 and T_2 can be any possible values constrained by Lemma 4.2. Then in Lemma 4.4 the approximation ratio r should be chosen as the maximum over all feasible x_1 and x_2. On the other hand, we can select appropriate μ and ρ to minimize the ratio r. Hence, by combining them together with the other constraints for the variables according to their definitions, the approximation ratio is the objective value of (4.15). □

We observe that here (4.15) is linear in x_1 and x_2 but nonlinear in μ and ρ. Besides, μ is an integer. So (4.15) is a mixed nonlinear integer program. Now we shall solve (4.15) to obtain the approximation ratio r depending on the number of machines m. Recall that the notation C^d means the class of all d-th order continuously differentiable functions. We first need the following lemma:

Lemma 4.7 *For two functions $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ defined on $[a, b]$ and $f(x)$, $g(x) \in C^1$, if one of the following two properties holds:*

Ω_1*: $f'(x) \cdot g'(x) < 0$ for all $x \in [a, b]$;*

Ω_2*: $f'(x) = 0$ and $g'(x) \neq 0$ for all $x \in [a, b]$,*

and the equation $f(x) = g(x)$ has a solution in interval $[a, b]$, then the root x_0 is unique and it minimizes the function $h(x) = \max\{f(x), g(x)\}$.

Proof: First we study the case of property Ω_1. Without loss of generality we assume that $f'(x) < 0$ and $g'(x) > 0$ holds for all $x \in [a, b]$. Their signs can not change in

the interval $[a, b]$. Otherwise $f(x) = 0$ occurs at certain point $x = y \in [a, b]$. Then $f(x)$ is strictly decreasing and $g(x)$ is strictly increasing. If there are two distinct roots x_1 and x_2 in $[a, b]$ to equation $f(x) = g(x)$, we have $f(x_1) - g(x_1) = 0$ and $f(x_2) - g(x_2) = 0$. Assuming $x_1 < x_2$, there must be an $x' \in (x_1, x_2)$ such that $f'(x') - g'(x') = 0$, i.e., $f'(x') = g'(x')$. This is a contradiction to $f'(x) < 0$ and $g'(x) > 0$. Thus the root is unique. Denote by x_0 the root, i.e., $f(x_0) = g(x_0)$. We are going to prove that $h(x_0) = \min_x \max\{f(x), g(x)\}$. For any $x \in [a, x_0)$, it holds that $f(x) > f(x_0) = g(x_0) > g(x)$. Thus $h(x) = \max\{f(x), g(x)\} = f(x) > f(x_0) = h(x_0)$ for all $x \in [a, x_0)$. Similarly we can prove for any $x \in (x_0, b]$, $h(x) = g(x) > g(x_0) = h(x_0)$. Therefore $h(x)$ is minimized at x_0. An example is illustrated in Figure 4.2.

Then we study the case of property Ω_2. Because $g(x) \in C^1$ and $g'(x) \neq 0$, $g'(x)$ is either positive or negative. Without loss of generality we assume that $g'(x) > 0$. Again, if there are two distinct roots x_1 and x_2 in $[a, b]$ to equation $f(x) = g(x)$, we have $f(x_1) - g(x_1) = 0$ and $f(x_2) - g(x_2) = 0$. Assuming $x_1 < x_2$, there must be an $x' \in (x_1, x_2)$ such that $f'(x') - g'(x') = 0$, i.e., $g'(x') = f'(x') = 0$. This is a contradiction to $g'(x) > 0$. Denote by x_0 the unique root, i.e., $f(x_0) = g(x_0)$. For any $x \in [a, x_0)$, it holds that $f(x) = f(x_0) = g(x_0) > g(x)$ as $f(x)$ is a constant. Thus $h(x) = \max\{f(x), g(x)\} = f(x) = f(x_0) = h(x_0)$. Similarly we can prove for any $x \in (x_0, b]$, $h(x) = g(x) > g(x_0) = h(x_0)$. Therefore $h(x)$ is minimized at x_0 and the minimum value is $f(x)$. Here $h(x_0)$ is not the unique minimum. An example is illustrated in Figure 4.3. □

Furthermore, we need the following propositions:

Proposition 4.2 *The roots of a quadratic equation* $ax^2 + bx + c = 0$ *(*$a \neq 0$*) are*

$$x_{1,2} = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}.$$

Proposition 4.3 *Suppose that* x_1 *is the largest real root to the equation* $f(x) = \sum_{i=1}^{n} a_i x^i = 0$ *(*$a_n \neq 0$*). If* $a_n > 0$*, then* $f(x) > 0$ *for* $x \in (x_1, \infty)$*. If* $a_n < 0$*, then* $f(x) < 0$ *for* $x \in (x_1, \infty)$*.*

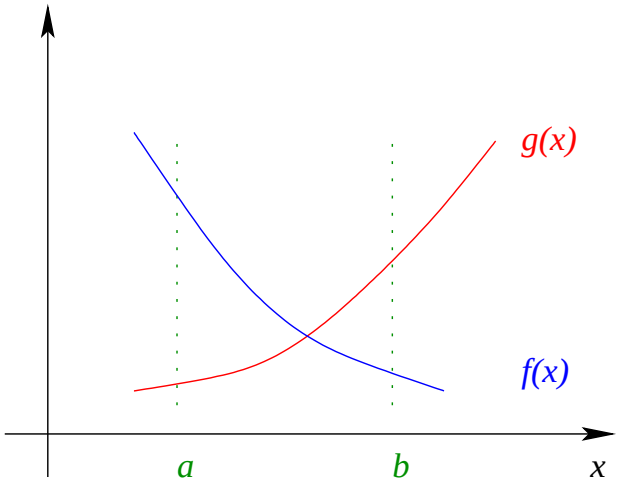


Figure 4.2: An example of functions with property Ω_1 in Lemma 4.7.

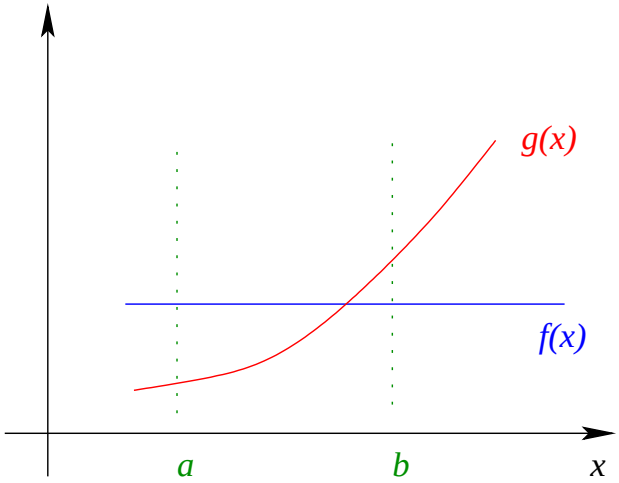


Figure 4.3: An example of functions with property Ω_2 in Lemma 4.7.

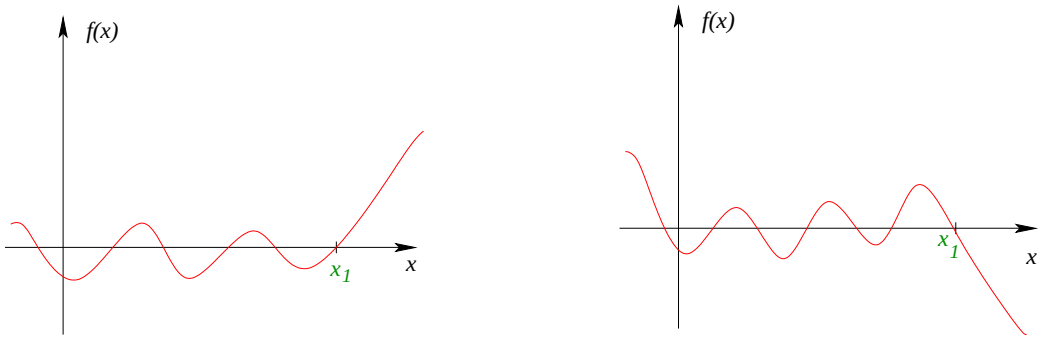
Examples of the polynomials with the properties in Proposition 4.3 are illustrated in Figure 4.4. Now we are able to prove the following theorem:

Theorem 4.1 *For each $m \in \mathbb{N}$ and $m \geq 3$, the linear relaxation of the min-max nonlinear program (4.15) has an optimum objective value*

$$OPT \leq 1 + \frac{\sqrt{5m^2 - 4m} + m}{2m} + \frac{\sqrt{5m - 4} + \sqrt{m}}{m - 1}\sqrt{\frac{\sqrt{5m^2 - 4m} - m - 2}{2}};$$

corresponding to μ^* and ρ^* defined in (4.10) and (4.11), respectively.

Proof: First we observe that the constraints on x_1 and x_2 in (4.15) form a triangle, and the extreme points are $E_1 : (x_1, x_2) = (1/\rho, 0)$, $E_2 : (x_1, x_2) = (0, m/(\rho\mu))$ and



$a_n > 0$ $a_n < 0$

Figure 4.4: Examples of polynomials in Proposition 4.3.

$E_3 : (x_1, x_2) = (0, 0)$. Since the min-max nonlinear program (4.15) is linear in x_1 and x_2, for a fixed pair of μ and ρ, the maximum value exists at one of the extreme points. It is obvious that at extreme point E_3 the maximum objective value can not be attained. Therefore we consider only E_1 and E_2. Denote by $A(\mu, \rho)$ and $B(\mu, \rho)$ the objective values at the two extreme points E_1 and E_2, respectively. Then they can be calculated as

$$\begin{aligned} A(\mu,\rho) &= \frac{m-\mu(1-\rho)}{\rho(1-\rho)(m-\mu+1)} = \frac{1}{\rho} + \frac{m\rho-(1-\rho)}{\rho(1-\rho)(m-\mu+1)}; \\ B(\mu,\rho) &= \frac{m(m-2\mu+1-m\rho-\rho+3\rho\mu)}{\rho(1-\rho)(m-\mu+1)\mu}. \end{aligned}$$

If $\rho > 1/(m+1)$, then $m\rho - (1-\rho) > 0$. Therefore $\dfrac{m\rho-(1-\rho)}{\rho(1-\rho)(m-\mu+1)}$ is also positive and $A(\mu, \rho)$ is strictly increasing in μ. In addition, consider the first order partial derivative of $B(\mu, \rho)$ with respect to μ:

$$\begin{aligned} B'(\mu,\rho)_\mu &= \frac{m}{\rho(1-\rho)\mu^2(m-\mu+1)^2}[(-2+3\rho)(m-\mu+1)\mu \\ &\quad -(m-2\mu+1-m\rho-\rho+3\rho\mu)(m-2\mu+1)] \\ &= \frac{-(2-3\rho)\mu^2+2(m+1)(1-\rho)\mu-(m+1)^2(1-\rho)}{\mu^2(m-\mu+1)^2} \cdot \frac{m}{\rho(1-\rho)}. \end{aligned}$$

If $\rho = 2/3$, then $2 - 3\rho = 0$ and the quadratic term of the numerator of $B'(\mu, \rho)_\mu$ is

0. Solving equation $B'(\mu,\rho)_\mu = 0$ we have the only root $\mu = (m+1)/2$. It means for any $\mu \in (1,(m+1)/2]$ the numerator is non-positive according to Proposition 4.3, i.e., $B'(\mu,\rho)_\mu \leq 0$. We now consider the case that $\rho \neq 2/3$. The quadratic equation $B'(\mu,\rho)_\mu = 0$ for the variable μ has following roots according to Proposition 4.2:

$$\begin{aligned}\mu &= \frac{-2(m+1)(1-\rho) \pm \sqrt{4(m+1)^2(1-\rho)^2 - 4(2-3\rho)(m+1)^2(1-\rho)}}{-2(2-3\rho)} \\ &= \frac{(m+1)(1-\rho)}{2-3\rho}\left(1 \mp \sqrt{\frac{2\rho-1}{1-\rho}}\right).\end{aligned}$$

If $\rho < 1/2$, then $\frac{2\rho-1}{1-\rho} < 0$. Then there is no real root to the equation $B'(\mu,\rho)_\mu = 0$ due to properties of quadratic equations. Thus in this case $B'(\mu,\rho)_\mu < 0$, i.e., $B(\mu,\rho)$ is strictly decreasing in μ.

Now we study the following three cases according to the value of ρ.

CASE 1: $\rho \in [1/(m+1), 1/2)$. Since we need to choose the minimum value of the maximum between $A(\mu,\rho)$ and $B(\mu,\rho)$ for all feasible μ, according to Lemma 4.7, only if $A(\mu,\rho) = B(\mu,\rho)$ can the minimum value be achieved. Equation $A(\mu,\rho) = B(\mu,\rho)$ is as follows:

$$\frac{m-\mu(1-\rho)}{\rho(1-\rho)(m-\mu+1)} = \frac{m(m-2\mu+1-m\rho-\rho+3\rho\mu)}{\rho(1-\rho)(m-\mu+1)\mu}.$$

After simplification (with elimination of a positive common factor $1-\rho$) the equation is:

$$\mu^2 - 3m\mu + m(m+1) = 0.$$

Solving it with Proposition 4.2 yields

$$\mu = \frac{3m \mp \sqrt{9m^2 - 4m(m+1)}}{2} = \frac{3m \mp \sqrt{5m^2-4m}}{2}.$$

Since $\frac{3m+\sqrt{5m^2-4m}}{2} \geq \frac{3m}{2} > \frac{m+1}{2}$, which violates the constraint that $\mu \leq \frac{m+1}{2}$, the only feasible solution to the equation $A(\mu,\rho) = B(\mu,\rho)$ is

$$\mu^* = \frac{3m - \sqrt{5m^2 - 4m}}{2}. \tag{4.16}$$

Here the solution μ^* is not related to ρ. Because μ^* must be an integer, we have (4.10).

To find the optimum ρ^*, substitute μ^* to $B(\mu, \rho)$ to obtain function $B(\rho)$ of ρ. Since μ^* in (4.16) does not depend on ρ, we can calculate the partial derivative of $B(\rho)$ with respect to ρ as follows:

$$\begin{aligned} B'(\rho)_\rho &= \frac{m}{(m-\mu^*+1)\mu^*} \cdot \frac{1}{\rho^2(1-\rho)^2}[(-m-1+3\mu^*)\rho(1-\rho) \\ &\quad -(m-2\mu^*+1-m\rho-\rho+3\mu^*\rho)(1-2\rho)] \\ &= \frac{m}{\mu^*}\frac{-(m-3\mu^*+1)\rho^2+2(m-2\mu^*+1)\rho-(m-2\mu^*+1)}{(m-\mu^*+1)\rho^2(1-\rho)^2}. \end{aligned}$$

When $m+1-3\mu^* = 0$, there is only one root $\rho^* = 1/2$ to the equation $B'(\rho)_\rho = 0$. Otherwise solving the quadratic equation $B'(\rho)_\rho = 0$, one can get

$$\begin{aligned} \rho &= \frac{-2(m-2\mu^*+1) \mp \sqrt{4(m-2\mu^*+1)^2 - 4(m-3\mu^*+1)(m-2\mu^*+1)}}{-2(m-3\mu^*+1)} \\ &= \frac{m+1-2\mu^* \mp \sqrt{\mu^*(m+1-2\mu^*)}}{m+1-3\mu^*}. \end{aligned}$$

Because $\dfrac{m+1-2\mu^*+\sqrt{\mu^*(m+1-2\mu^*)}}{m+1-3\mu^*} \geq \dfrac{m+1-2\mu^*}{m+1-3\mu^*} \geq 1$, this solution violates the constraint that $\rho < \dfrac{1}{2}$. Thus we have

$$\rho^* = \frac{m+1-2\mu^* - \sqrt{\mu^*(m+1-2\mu^*)}}{m+1-3\mu^*}, \tag{4.17}$$

as in (4.11).

In order to show that $\rho^* \leq 1/2$ with our assumption, we need that

$$\rho^* = \frac{m+1-2\mu^*}{m+1-2\mu^*+\sqrt{\mu^*(m+1-2\mu^*)}} \leq \frac{1}{2},$$

which is equivalent to

$$m+1-2\mu^* \leq \sqrt{\mu^*(m+1-2\mu^*)}.$$

Taking the square of both sides we have $m+1 \leq 3\mu^*$. Since $\mu^* = (3m-\sqrt{5m^2-4m})/2$, substituting it to the inequality yields

$$7m-2 \geq 3\sqrt{5m^2-4m}.$$

Taking the square of both side again we obtain that $4(m+1)^2 \geq 0$, which is always true. Therefore we have shown that the ρ^* in (4.11) is less than $1/2$. Now we are going to prove that $1/(m+1) < \rho^*$. Similar as before, this inequality is equivalent to

$$\frac{1}{m+1} < \frac{1}{1+\sqrt{\dfrac{\mu^*}{m+1-2\mu^*}}} = \rho^*.$$

With simplification it becomes $m^2(m+1-2\mu^*) > \mu^*$, i.e., $m^2(m+1) > (2m^2+1)\mu^*$. So it is required that

$$\mu^* = \frac{3m-\sqrt{5m^2-4m}}{2} < \frac{m^2(m+1)}{2m^2+1}.$$

It can then be simplified as

$$4m^3-2m^2+3m < (2m^2+1)\sqrt{5m^2-4m}.$$

Take the square of both sides and simplify it. Then we have the following inequality:

$$m^5-2m^3-m^2-m-1 > 0.$$

It is true because we observe that it always holds for $m \geq 2$ as

$$m^5-2m^3-m^2-m-1 = m(m^2-1)^2-(m+1)^2 = (m+1)^2[m(m-1)^2-1],$$

and the right hand side is always positive for $m \geq 2$. Therefore we conclude that ρ^* in (4.17) lies in the assumed domain $[1/(m+1), 1/2)$.

Another approach to obtain the optimum ρ^* is to substitute μ^* to $A(\mu, \rho)$ to obtain a function $A(\rho)$ of ρ and with similar analysis as above. Again, as μ^* is independent of ρ, the first order partial derivative of $A(\rho)$ with respect to ρ is:

$$\begin{aligned} A'(\rho)_\rho &= \frac{\mu^*\rho(1-\rho) - (m - \mu^*(1-\rho))(1-2\rho)}{\rho^2(1-\rho)^2(m-\mu^*+1)} \\ &= \frac{\mu^*\rho^2 + 2(m-\mu^*)\rho - (m-\mu^*)}{\rho^2(1-\rho)^2(m-\mu^*+1)}. \end{aligned}$$

Solving the quadratic equation $A'(\rho)_\rho = 0$ we obtain the following roots by Proposition 4.2:

$$\begin{aligned} \rho &= \frac{-2(m-\mu^*) \mp \sqrt{4(m-\mu^*)^2 + 4\mu^*(m-\mu^*)}}{2\mu^*} \\ &= \frac{-(m-\mu^*) \mp \sqrt{m(m-\mu^*)}}{\mu^*}. \end{aligned}$$

Because $\dfrac{-(m-\mu^*) - \sqrt{m(m-\mu^*)}}{\mu^*} < 0$, this root violate the constraint that $\rho > 0$. Thus we have

$$\begin{aligned} \rho^* &= \frac{-(m-\mu^*) + \sqrt{m(m-\mu^*)}}{\mu^*} \\ &= \frac{m-\mu^*}{m-\mu^* + \sqrt{m(m-\mu^*)}} \\ &= \frac{1}{1 + \sqrt{\dfrac{m}{m-\mu^*}}}. \end{aligned} \tag{4.18}$$

Since $\mu^* > 0$, we have $\sqrt{\dfrac{m}{m-\mu^*}} > 1$. Therefore $\rho^* = \dfrac{1}{1+\sqrt{m/(m-\mu^*)}} \leq \dfrac{1}{2}$ in (4.18). In addition, because $m \geq \mu^* + 1$ and $m > 1$, we have $m(m-\mu^*) > 1$, i.e. $1/(m-\mu^*) < m$. Multiplying both sides by m and taking square root of both

sides leads to $\sqrt{m/(m-\mu^*)} < m$. So $\dfrac{1}{m+1} < \dfrac{1}{1+\sqrt{m/(m-\mu^*)}} = \rho^*$. The ρ^* in (4.18) is in the assumed domain $\rho \in (1/(m+1), 1/2)$. In fact one can prove that ρ^* in (4.17) and (4.18) are equivalent with the μ^* in (4.16).

However, the optimum μ^* in (4.16) can be fractional while it is required that μ is an integer. Thus we can take the integer μ^* by rounded it to the integer below or above the fractional one in (4.16). Then we obtain (4.10). Now we need to consider the upper bound of the optimal objective value of (4.15) with rounded μ^*. It is worth noting that the optimum ρ^* is a function of μ^*. Thus we can regard $A(\mu^*, \rho^*)$ and $B(\mu^*, \rho^*)$ as functions of μ^*, i.e., $A(\mu^*, \rho^*(\mu^*))$ and $B(\mu^*, \rho^*(\mu^*))$, or simply $A(\mu)$ and $B(\mu)$. Now we need to examine the behaviour of functions $A(\mu)$ and $B(\mu)$.

The function $A(\mu)$ is as follows:

$$\begin{aligned}
A(\mu) = A(\mu, \rho^*(\mu)) &= \frac{m - \mu\dfrac{\sqrt{m/(m-\mu)}}{1+\sqrt{m/(m-\mu)}}}{\dfrac{\sqrt{m/(m-\mu)}}{(1+\sqrt{m/(m-\mu)})^2}(m-\mu+1)} \\
&= \frac{(1+\sqrt{m/(m-\mu)})(m+\sqrt{m(m-\mu)})}{\sqrt{m/(m-\mu)}(m-\mu+1)} \\
&= \frac{(\sqrt{m}+\sqrt{m-\mu})^2}{m-\mu+1} \\
&= 1 + \frac{m-1}{m-\mu+1} + \frac{2\sqrt{m(m-\mu)}}{m-\mu+1}.
\end{aligned}$$

It is obvious that $(m-1)/(m-\mu+1)$ is increasing in μ. We now check the following derivative with respect to μ:

$$\begin{aligned}
\left(\frac{\sqrt{m-\mu}}{m-\mu+1}\right)'_{\mu} &= \frac{-\dfrac{m-\mu+1}{2(\sqrt{m-\mu})} + \sqrt{m-\mu}}{(m-\mu+1)^2} \\
&= \frac{\sqrt{m-\mu} - \dfrac{1}{\sqrt{m-\mu}}}{2(m-\mu+1)^2}.
\end{aligned}$$

Since $\mu \leq (m+1)/2$, we have $\sqrt{m-\mu} \geq \sqrt{(m-1)/2}$. Therefore

$$\sqrt{m-\mu} - \frac{1}{\sqrt{m-\mu}} \geq \sqrt{\frac{m-1}{2}} - \sqrt{\frac{2}{m-1}}.$$

If $m \geq 3$, then $(m+1)(m-3) \geq 0$. So $(m-1)^2 \geq 4$, i.e., $(m-1)/2 \geq 2/(m-1)$. Therefore $\sqrt{(m-1)/2} \geq \sqrt{2/(m-1)}$ and $[\sqrt{m-\mu}/(m-\mu+1)]'_\mu \geq 0$. In this way we conclude that $A(\mu)$ is increasing in μ.

We substitute (4.17) to $B(\mu^*, \rho^*(\mu^*))$ to obtain $B(\mu)$ as follows:

$$\begin{aligned} B(\mu) &= B(\mu, \rho^*(\mu)) = \frac{m\sqrt{\mu(m-2\mu+1)}}{\frac{\sqrt{\mu/(m-2\mu+1)}}{(1+\sqrt{\mu/(m-2\mu+1)})^2}(m-\mu+1)\mu} \\ &= \frac{[(m-\mu+1)/(m-2\mu+1) + 2\sqrt{\mu/(m-2\mu+1)}]m\sqrt{\mu(m-2\mu+1)}}{\sqrt{\mu/(m-2\mu+1)}\mu(m-\mu+1)} \\ &= \frac{m}{\mu} + \frac{2m}{\sqrt{\mu(m-2\mu+1)}}. \end{aligned}$$

Then we examine its first order derivative with respect to μ, i.e.,

$$\begin{aligned} B'(\mu)_\mu &= \left(\frac{m}{\mu} + \frac{2m}{\sqrt{\mu(m-2\mu+1)}}\right)'_\mu \\ &= -\frac{m}{\mu^2} - \frac{m(m-4\mu+1)}{((m-2\mu+1)\mu)^{3/2}}. \end{aligned}$$

Since the denominators of both terms are positive, the equation $B'(\mu)_\mu = 0$ is equivalent to

$$(m-2\mu+1)^{3/2} = -(m-4\mu+1)\sqrt{\mu}.$$

Taking square of both sides with simplification one has

$$(m+1)^3 - 7\mu(m+1)^2 + 20\mu^2(m+1) - 24\mu^3 = (m+1-3\mu)[(m+1)^2 - 4(m+1)\mu + 8\mu^2] = 0.$$

It is obvious that second factor $(m+1)^2-4(m+1)\mu+8\mu^2 = (m+1-2\mu)^2+4\mu^2 > 0$ for any feasible m and μ. Therefor the only real root to the equation $B'(\mu)_\mu = 0$ is $\mu = (m+1)/3$. However, we can show that the special case that $\mu = (m+1)/3$ can not happen for any $m \geq 19$ as follows. The quadratic equation $m^2 - 19m + 16 = 0$ has two roots

$$m_1 = (19 + \sqrt{297})/2 \approx 18.1168, \quad m_2 = (19 - \sqrt{297})/2 \approx 0.8832.$$

Thus for any $m \geq 19$, $m^2 - 19m + 16 > 0$, i.e., $49m^2 - 112m + 64 > 45m^2 - 36m$. Since both sides are positive for $m \geq 2$, we can take the square roots of them and obtain $7m - 8 > 3\sqrt{5m^2 - 4m}$, i.e., $9m - 3\sqrt{5m^2 - 4m} > 2m + 8$. So we have

$$\frac{3m - \sqrt{5m^2 - 4m}}{2} - 1 > \frac{m+1}{3}.$$

Notice here μ is in fact the optimal over all integers in $\{1, \ldots, \lfloor (m+1)/2 \rfloor\}$. Since μ is an integer chosen according to (4.10), $\mu \geq (3m - \sqrt{5m^2 - 4m})/2 - 1$. Therefore for any $m \geq 19$, $\mu > (m+1)/3$ and the equation $B'(\mu)_\mu = 0$ has no root. We will also show later for all $3 \leq m \leq 18$, $\mu > (m+1)/3$ by listing the values. Thus for any $m \geq 3$, it holds that $B'(\mu)_\mu < 0$, i.e., $B(\mu)$ is decreasing in μ.

We now consider the optimal value of (4.15) with the integer value of μ^* in (4.10) and the corresponding ρ^* in (4.11). We still denote by $\rho^*(\mu^*)$ the function of μ^* in (4.11). In addition, we denote by μ^* the fractional value of optimal μ in (4.16) and its rounded values as $\lfloor \mu^* \rfloor$ and $\lceil \mu^* \rceil$. According to the arguments above, we know that for $m \geq 3$, $A(\mu^*, \rho^*(\mu^*))$ is increasing in μ^* and $B(\mu^*, \rho^*(\mu^*))$ is decreasing in μ^*. Thus $A(\lceil \mu^* \rceil, \rho^*(\lceil \mu^* \rceil)) \geq B(\lceil \mu^* \rceil, \rho^*(\lceil \mu^* \rceil))$ and we have:

$$\begin{aligned}
OPT &= \min\{\max\{A(\lfloor \mu^* \rfloor, \rho^*(\lfloor \mu^* \rfloor)), B(\lfloor \mu^* \rfloor, \rho^*(\lfloor \mu^* \rfloor))\}, \\
&\quad \max\{A(\lceil \mu^* \rceil, \rho^*(\lceil \mu^* \rceil)), B(\lceil \mu^* \rceil, \rho^*(\lceil \mu^* \rceil))\}\} \\
&\leq \max\{A(\lceil \mu^* \rceil, \rho^*(\lceil \mu^* \rceil)), B(\lceil \mu^* \rceil, \rho^*(\lceil \mu^* \rceil))\} \\
&= A(\lceil \mu^* \rceil, \rho^*(\lceil \mu^* \rceil)) \\
&\leq A(\mu^* + 1, \rho^*(\mu^* + 1))
\end{aligned}$$

$$
\begin{aligned}
&= \frac{m-(\mu^*-1)\dfrac{\sqrt{m/(m-\mu^*-1)}}{1+\sqrt{m/(m-\mu^*-1)}}}{\dfrac{\sqrt{m/(m-\mu^*-1)}}{(1+\sqrt{m/(m-\mu^*-1)})^2}(m-\mu^*)} \\
&= \frac{(1+\sqrt{m/(m-\mu^*-1)})(m+(m-\mu^*-1)\sqrt{m/(m-\mu^*-1)})}{\sqrt{m/(m-\mu^*-1)}(m-\mu^*)} \\
&= \frac{(\sqrt{m}+\sqrt{m-\mu^*-1})(m+\sqrt{m(m-\mu^*-1)})}{\sqrt{m}(m-\mu^*)} \\
&= \frac{(\sqrt{m}+\sqrt{m-\mu^*-1})^2}{m-\mu^*} \\
&= 1+\frac{m-1}{m-\mu^*}+\frac{2\sqrt{m(m-\mu^*-1)}}{m-\mu^*} \\
&= 1+\frac{\sqrt{5m^2-4m}+m}{2m}+\frac{\sqrt{5m-4}+\sqrt{m}}{m-1}\sqrt{\frac{\sqrt{5m^2-4m}-m-2}{2}}.
\end{aligned}
$$

This gives the bound claimed in the theorem and the analysis for **CASE 1** is completed.

CASE 2: $\rho \in (0, 1/(m+1)]$. In this case $m\rho-(1-\rho) < 0$. Therefore $\dfrac{m\rho-(1-\rho)}{\rho(1-\rho)(m-\mu+1)}$ is negative and $A(\mu,\rho)$ is decreasing in μ. Since $B(\mu,\rho)$ is decreasing in μ for $\rho < 1/2$, we have that both $A(\mu,\rho)$ and $B(\mu,\rho)$ are decreasing in this interval. Thus we can fix $\mu^* = (m+1)/2$ to reach the minimum of objective function. Substitute μ^* to $A(\mu,\rho)$ and $B(\mu,\rho)$ we have:

$$
\begin{aligned}
A(\rho) &= \frac{m-\dfrac{m+1}{2}(1-\rho)}{\rho(1-\rho)(m+1-\dfrac{m+1}{2})} = \frac{m-1+\rho(m+1)}{\rho(1-\rho)(m+1)}; \\
B(\rho) &= \frac{m(m-(m+1)+1-m\rho-\rho+3\rho\cdot\dfrac{m+1}{2})}{\rho(1-\rho)(m-\dfrac{m+1}{2}+1)\dfrac{m+1}{2}} = \frac{2m}{(1-\rho)(m+1)}.
\end{aligned}
$$

Since $\rho < 1$, $m-1+\rho(m+1) > 2m\rho$. Thus $A(\rho) > B(\rho)$ when $\mu = (m+1)/2$ (See

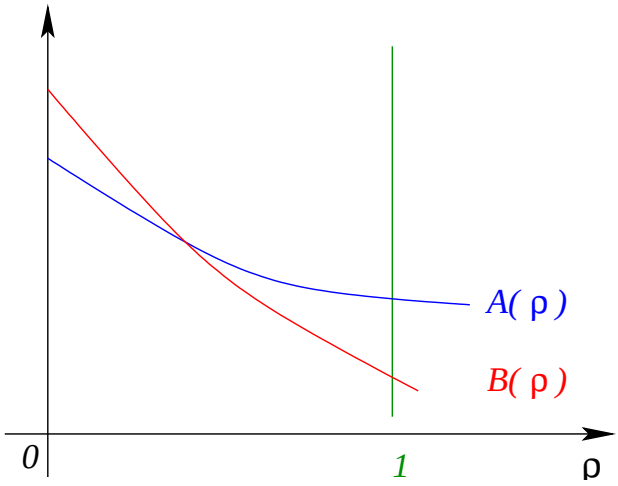


Figure 4.5: Functions $A(\rho)$ and $B(\rho)$ in **CASE 2**.

Figure 4.5). We are going to minimize $A(\rho)$ for $\rho \in (0, 1/(m+1)]$. The first order partial derivative of $A(\rho)$ with respect to ρ is

$$\begin{aligned} A'(\rho)_\rho &= \left[\frac{m-1}{m+1}\frac{1}{\rho} + \frac{2m}{m+1}\frac{1}{1-\rho}\right]'_\rho \\ &= -\frac{m-1}{m+1}\frac{1}{\rho^2} + \frac{2m}{m+1}\frac{1}{(1-\rho)^2} \\ &= \frac{2m\rho^2 - (m-1)(1-\rho)^2}{(m+1)\rho^2(1-\rho)^2} \\ &= \frac{(m+1)\rho^2 + 2(m-1)\rho - (m-1)}{(m+1)\rho^2(1-\rho)^2}. \end{aligned}$$

It is clear that the denominator is positive. We now examine the equation $A'(\rho)_\rho = 0$. It is equivalent to

$$(m+1)\rho^2 + 2(m-1)\rho - (m-1) = 0.$$

According to Proposition 4.2, the roots are as follows:

$$\rho = \frac{-2(m-1) \mp \sqrt{4(m-1)^2 + 4(m^2-1)}}{2(m+1)} = \frac{1-m \mp \sqrt{2m(m+1)}}{m+1}.$$

It is obvious that for $m \geq 1$, $\dfrac{1-m-\sqrt{2m(m+1)}}{m+1} < 0$ is infeasible. Since $2m^2 + 2m > m^2$, we have $\sqrt{2m^2+2m} > m$, i.e., $1 - m + \sqrt{2m^2+2m} > 1$. So the root

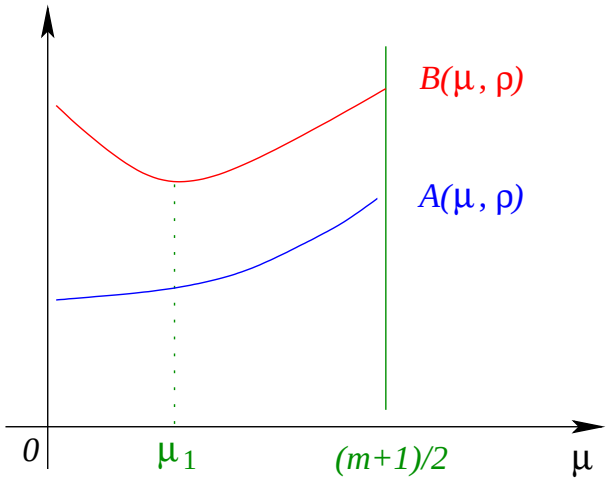


Figure 4.6: Functions $A(\mu, \rho)$ and $B(\mu, \rho)$ in **CASE 3.2.1**.

$\frac{1-m+\sqrt{2m(m+1)}}{m+1} > \frac{1}{m+1}$ is also infeasible. Thus when $\rho \in (0, 1/(m+1)$, $A'(\rho)_\rho < 0$, i.e., $A(\rho)$ is decreasing in ρ. Then we fix $\rho^* = 1/(m+1)$ to obtain the optimal objective value:

$$OPT = A(\rho)|_{\rho=1/(m+1)} = \frac{m-1+1}{m/(m+1)} = m+1.$$

However, we show now that the optimal objective value in this case is not better than the bound on the optimal value for the case $\rho \in [1/(m+1), 1/2)$ claimed in the theorem for $m \geq 3$. Denote by $BOUND$ the bound in the theorem (also for the case $\rho \in [1/(m+1), 1/2)$). It is obvious that

$$BOUND \leq 1 + \frac{\sqrt{5}+1}{2} + \frac{\sqrt{5}+1}{m-1} m \sqrt{\frac{\sqrt{5}-1}{2}}.$$

Therefore if the right hand side of the above inequality is not greater than $m+1$ our claim holds. It is equivalent to

$$m^2 - \left[1 + \frac{\sqrt{5}+1}{2} + (\sqrt{5}+1)\sqrt{\frac{\sqrt{5}-1}{2}}\right] m + \frac{\sqrt{5}+1}{2} \approx m^2 - 5.162m + 1.618 \geq 0.$$

It has two real roots $m_3 \approx 4.8268$ and $m_4 = 0.3352$, which shows when $m \geq 5 > m_3$, the inequality $BOUND \leq m+1$ holds. It can be calculated that if $m = 3$,

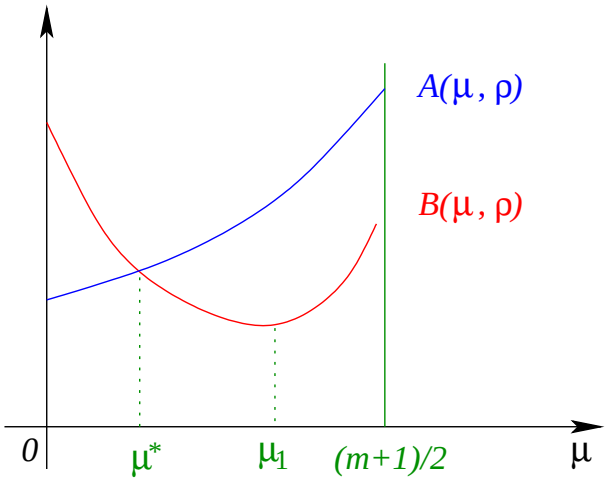


Figure 4.7: Functions $A(\mu, \rho)$ and $B(\mu, \rho)$ in one case of **CASE 3.2.2**.

$BOUND = 3.9976 \leq m+1 = 4$, and if $m = 4$, $BOUND = 4.5 \leq m+1 = 5$. Hence we conclude that for any $m \geq 3$, the optimal value of (4.15) can not be attained for $\rho \leq 1/(m+1)$. This completes the analysis for **CASE 2**.

CASE 3: $\rho \geq 1/2$. Here we have $(2\rho - 1)/(1 - \rho) > 0$. Therefore there exist roots for the quadratic equation $B'(\mu, \rho)_\mu = 0$. Remember that in this case $A(\mu, \rho)$ is increasing in μ. Since $\rho \neq 2/3$, we need to consider the following two cases:

- **CASE 3.1**: $\rho \in [1/2, 2/3)$;
- **CASE 3.2**: $\rho \in (2/3, 1)$.

In **CASE 3.1**, $3\rho < 2$, i.e. $2\rho - 1 < 1 - \rho$. Therefore $\sqrt{(2\rho-1)/(1-\rho)} < 1$. Then the following two roots by Proposition 4.2:

$$\mu_1 = \frac{(m+1)(1-\rho)}{2-3\rho}\left(1 - \sqrt{\frac{2\rho-1}{1-\rho}}\right),$$

$$\mu_2 = \frac{(m+1)(1-\rho)}{2-3\rho}\left(1 + \sqrt{\frac{2\rho-1}{1-\rho}}\right)$$

are positive as $2-3\rho > 0$. In addition, since $(3\rho-2)^2 > 0$, we have $\rho^2 > -4+12\rho-8\rho^2 = 4(1-\rho)(2\rho-1)$. As both sides are positive, we take the square root and it becomes $\rho > 2\sqrt{(1-\rho)(2\rho-1)}$. Then $2-2\rho-2(1-\rho)\sqrt{(2\rho-1)/(1-\rho)} > 2-3\rho$.

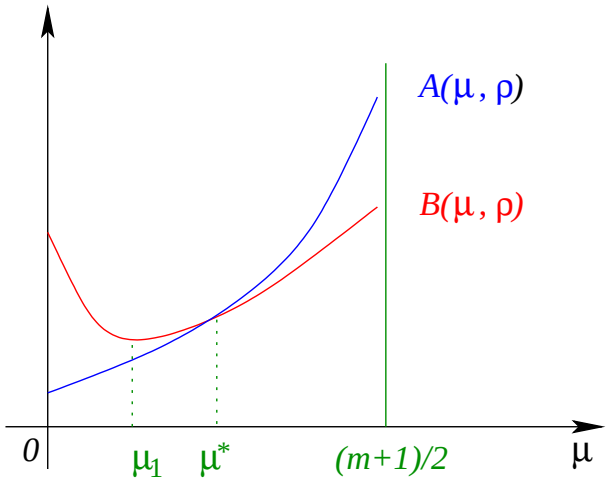


Figure 4.8: Functions $A(\mu, \rho)$ and $B(\mu, \rho)$ in one case of **CASE 3.2.2**.

Thus $\mu_2 \geq \mu_1 > (m+1)/2$. Both roots violate the constraint $\mu \leq (m+1)/2$. Furthermore, $B'(\mu, \rho)_\mu$ is negative. So $B(\mu, \rho)$ is decreasing in μ. Then we can solve the equation $A(\mu, \rho) = B(\mu, \rho)$ to obtain the same optimum μ^* as in (4.16) and also the same optimum ρ^* in (4.17) or (4.18). However, the optimum $\rho^* < 1/2$, which violates the assumption of **CASE 3.1**. So in this case the bound on the approximation ratio is not better than that of the case that $\rho \in [1/(m+1), 1/2)$.

In **CASE 3.2**, with the similar analysis in **CASE 3.1**, $\sqrt{(2\rho-1)/(1-\rho)} > 1$. Since $2 - 3\rho < 0$, we have $\mu_2 < 0$ and $\mu_1 > 0$. Because $(3\rho-2)^2 > 0$, $\rho^2 > -4+12\rho-8\rho^2 = 4(1-\rho)(2\rho-1)$. As both sides are positive, we take the square root and it becomes $\rho > 2\sqrt{(1-\rho)(2\rho-1)}$. Then $2\sqrt{(1-\rho)(2\rho-1)} - 2 + 2\rho < 3\rho - 2$, which indicates $\mu_1 < (m+1)/2$ and thus feasible. So $B'(\mu, \rho)_\mu < 0$ when $\mu \in [1, \mu_1]$ and $B'(\mu, \rho)_\mu > 0$ when $\mu \in (\mu_1, (m+1)/2]$. It means that $B(\mu, \rho)$ is decreasing in μ for $\mu \in [1, \mu_1]$ while it is increasing in μ for $\mu \in (\mu_1, (m+1)/2]$. Since $A(\mu, \rho)$ is increasing in μ for all μ when $\rho \geq 1/(m+1)$, there are four possibilities for the relations between $A(\mu, \rho)$ and $B(\mu, \rho)$:

- **CASE 3.2.1**: $A(\mu, \rho) < B(\mu, \rho)$ for all feasible μ;
- **CASE 3.2.2**: Equation $A(\mu, \rho) = B(\mu, \rho)$ has one root for all feasible μ;
- **CASE 3.2.3**: Equation $A(\mu, \rho) = B(\mu, \rho)$ has two roots for all feasible μ;

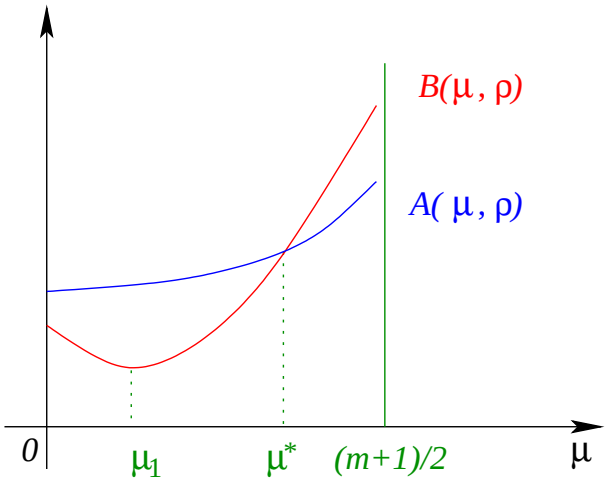


Figure 4.9: Functions $A(\mu, \rho)$ and $B(\mu, \rho)$ in one case of **CASE 3.2.2**.

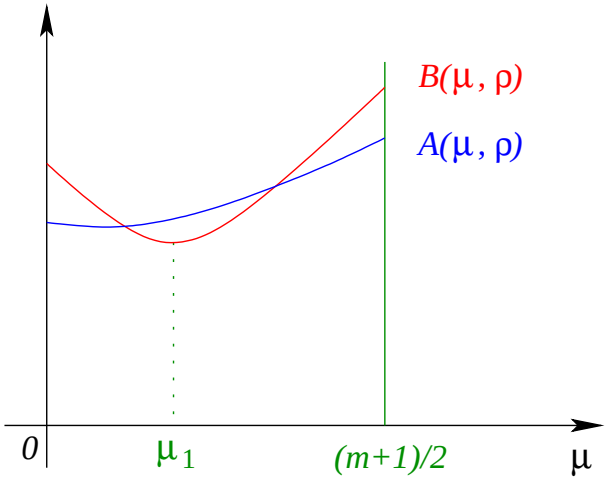


Figure 4.10: Functions $A(\mu, \rho)$ and $B(\mu, \rho)$ in **CASE 3.2.3**.

- **CASE 3.2.4**: $A(\mu, \rho) > B(\mu, \rho)$ for all feasible μ.

In **CASE 3.2.1**, equation $A(\mu, \rho) = B(\mu, \rho)$ has no root and when $\mu = (m+1)/2$, $A(\mu, \rho) < B(\mu, \rho)$ (See Figure 4.6). In **CASE 3.2.2**, when $\mu = 1$, $A(\mu, \rho) < B(\mu, \rho)$ and when $\mu = (m+1)/2$, $A(\mu, \rho) > B(\mu, \rho)$, or vice versa (See Figure 4.7, 4.8 and 4.9). In **CASE 3.2.3**, when $\mu = 1$ and $\mu = (m+1)/2$, $A(\mu, \rho) < B(\mu, \rho)$, and equation $A(\mu, \rho) = B(\mu, \rho)$ has two roots for $\mu \in [1, (m+1)/2]$ (See Figure 4.10). In **CASE 3.2.4**, equation $A(\mu, \rho) = B(\mu, \rho)$ has no root and when $\mu = 1$, $A(\mu, \rho) > B(\mu, \rho)$ (See Figure 4.11). Now we examine the existence of roots to the equation $A(\mu, \rho) = B(\mu, \rho)$. It is

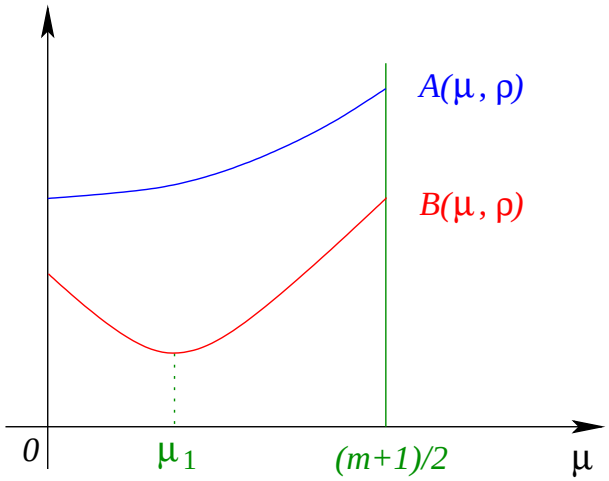


Figure 4.11: Functions $A(\mu, \rho)$ and $B(\mu, \rho)$ in **CASE 3.2.4**.

$$\mu^2 - 3m\mu + m(m+1) = 0.$$

Same as the previous analysis, there exists only one feasible root of equation $A(\mu, \rho) = B(\mu, \rho)$

$$\mu^* = \frac{3m - \sqrt{5m^2 - 4m}}{2},$$

and the other root is not feasible (not in the interval $[0, (m+1)/2]$). So neither **CASE 3.2.1**, **CASE 3.2.3** nor **CASE 3.2.4** is feasible. We now consider **CASE 3.2.2**. In Figure 4.7, the optimal value of (4.15) exists at the point $\mu = \mu^*$. However, in Figure 4.8 and 4.9, the optimal value exists at the point $\mu = \mu_1$. Here we just need to search for an upper bound of the optimal value. Therefore in this case we still count the objective value at the point $\mu = \mu^*$ as the upper bound. Thus we have finished the analysis for **CASE 3.2**.

Then the analysis for **CASE 3** is complete and the theorem is proved. □

It can be observed that our fractional μ^* in (4.16) is exactly the same as that in [81]. As μ^* must be chosen as an integer, our optimal value of μ^* only differs from that in [81] by at most one.

For our algorithm for Scheduling Malleable Tasks with Precedence Constraints, the approximation ratio obeys the following corollary:

Corollary 4.1 *For each $m \in \mathbb{N}$ and $m \geq 2$, Algorithm I has an approximation ratio*

$$r \leq \begin{cases} 4, & \text{if } m = 2; \\ 1 + \dfrac{\sqrt{5m^2 - 4m} + m}{2m} & \\ \quad + \dfrac{\sqrt{5m-4} + \sqrt{m}}{m-1}\sqrt{\dfrac{\sqrt{5m^2-4m} - m - 2}{2}}, & \text{otherwise.} \end{cases} \tag{4.19}$$

Proof: The conclusion comes directly from Lemma 4.4 and Theorem 4.1. If $m = 2$, then μ^* must be set as 1 and the condition that $\mu > (m+1)/3$ does not hold. In this case the objective values of (4.15) on the two extreme points are

$$\begin{aligned} A(1,\rho) &= \frac{1+\rho}{2\rho(1-\rho)}, \\ B(1,\rho) &= \frac{1}{\rho(1-\rho)}. \end{aligned}$$

Because $\rho < 1$, it holds that $(1+\rho)/2 < 1$. Thus $A(1,\rho) < B(1,\rho)$ for any $\rho \in (0,1)$. Therefore the optimal value is bounded by $B(1,\rho)$. We can choose an optimal ρ^* to minimize $B(1,\rho)$. Because $\rho(1-\rho) = 1/4 - (\rho - 1/2)^2 \leq 1/4$, only if $\rho = 1/2$ can $B(1,\rho)$ attain the minimum. Therefore when $m = 2$ we set $\rho^* = 1/2$ and $\mu^* = 1$, and the approximation ratio $r \leq B(1,1/2) = 4$. □

The values of ratio r for m from 2 to 33 are listed in Table 4.2, which is to compare with the result in [81] listed in Table 4.3. To compute the values, we need to substitute the values of μ^* in (4.10) to (4.11) to obtain the corresponding values of ρ^*. Then we substitute the two groups of μ^* and ρ^* to the formulae $A(\mu,\rho)$ and $B(\mu,\rho)$. For each group of parameters, we find the maximum between $A(\mu,\rho)$ and $B(\mu,\rho)$. Then the minimum of these two maximums is the approximation ratio r. This gives an improvement of almost all m (except for $m = 2$). With sophisticated analysis of the formula (4.19), the following corollary holds:

Corollary 4.2 *For all $m \in \mathbb{N}$ and $m \geq 2$, the approximation ratio*

$$r \leq \frac{3+\sqrt{5}}{2} + \sqrt{2(\sqrt{5}+1)}.$$

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.500	4.0000	18	8	0.427	4.9848
3	2	0.366	3.7321	19	8	0.432	4.9095
4	2	0.414	3.8856	20	8	0.436	4.9333
5	3	0.387	4.4415	21	9	0.431	4.9807
6	3	0.366	4.4151	22	9	0.435	4.9159
7	3	0.431	4.6263	23	10	0.429	5.0417
8	4	0.414	4.6627	24	10	0.433	4.9774
9	4	0.427	4.5694	25	10	0.436	4.9206
10	5	0.464	4.9744	26	11	0.432	5.0311
11	5	0.425	4.7497	27	11	0.435	4.9747
12	5	0.433	4.7240	28	11	0.438	5.0279
13	6	0.423	4.8848	29	12	0.434	5.0226
14	6	0.431	4.7962	30	12	0.436	4.9724
15	6	0.436	4.9495	31	13	0.432	5.0655
16	7	0.429	4.9000	32	13	0.435	5.0158
17	7	0.434	4.8252	33	13	0.437	5.0040

Table 4.2: Bounds on approximation ratios for Algorithm I.

m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$
2	1	4.0000	10	4	5.0000	18	8	5.0908	26	10	5.1250
3	2	4.0000	11	5	4.8570	19	8	5.0000	27	11	5.0588
4	2	4.0000	12	5	4.8000	20	8	5.0000	28	11	5.0908
5	3	4.6667	13	6	5.0000	21	9	5.0768	29	12	5.1111
6	3	4.5000	14	6	4.8889	22	9	5.0000	30	12	5.0526
7	3	4.6667	15	6	5.0000	23	9	5.1111	31	13	5.1578
8	4	4.8000	16	7	5.0000	24	10	5.0667	32	13	5.1000
9	4	4.6667	17	7	4.9091	25	10	5.0000	33	13	5.0768

Table 4.3: Bounds on approximation ratios for the algorithm in [81].

Furthermore, when $m \to \infty$, the upper bound in Corollary 4.1 tends to

$$\frac{3+\sqrt{5}}{2} + \sqrt{2(\sqrt{5}+1)} \approx 5.162.$$

Proof: We still denote by $BOUND$ the bound claimed in Theorem 4.1, i.e.,

$$BOUND = 1 + \frac{\sqrt{5m^2-4m}+m}{2m} + \frac{\sqrt{5m-4}+\sqrt{m}}{m-1}\sqrt{\frac{\sqrt{5m^2-4m}-m-2}{2}}.$$

It is obvious that

$$\frac{\sqrt{5m^2-4m}+m}{2m} \leq \frac{\sqrt{5}+1}{2}.$$

Now we examine the last term in $BOUND$. Suppose

$$\frac{\sqrt{5m-4}+\sqrt{m}}{m-1}\sqrt{\frac{\sqrt{5m^2-4m}-m-2}{2}} \leq (\sqrt{5}+1)\sqrt{\frac{\sqrt{5}-1}{2}}. \tag{4.20}$$

Since both sides are positive, we take square of them and obtain the following inequality:

$$(3m-2+\sqrt{5m^2-4m})(\sqrt{5m^2-4m}-m-2) \leq 2(\sqrt{5}+1)(m-1)^2,$$

i.e.,

$$(m-2)\sqrt{5m^2-4m} \leq \sqrt{5}m^2 - 2(\sqrt{5}-1)m + (\sqrt{5}-1).$$

Again, we take the square of both sides as they are positive. After simplification we obtain:

$$(2+2\sqrt{5})m^3 - (1+5\sqrt{5})m^2 + 4(\sqrt{5}-1)m + 3 - \sqrt{5} \geq 0.$$

If $m \geq \dfrac{1+5\sqrt{5}}{2+2\sqrt{5}} \approx 1.8820$, then $(2+2\sqrt{5})m^3 - (1+5\sqrt{5})m^2 \geq 0$. Furthermore, $4(\sqrt{5}-1)m+3-\sqrt{5}$ are always positive. Thus (4.20) holds. We then conclude that

$$r \leq BOUND \leq 1 + \frac{\sqrt{5}+1}{2} + (\sqrt{5}+1)\sqrt{\frac{\sqrt{5}-1}{2}} \;=\; \frac{3+\sqrt{5}}{2} + \sqrt{2(\sqrt{5}+1)}.$$

In addition, it is easy to verify when $m \to \infty$, the upper bound in Corollary 4.1

$$\begin{aligned}
& \frac{m + \sqrt{5m^2 - 4m} + 2\sqrt{2m(\sqrt{5m^2 - 4m} - m)}}{\sqrt{5m^2 - 4m} - m + 2} \\
= & \frac{1 + \sqrt{5 - 4/m} + 2\sqrt{2(\sqrt{5 - 4/m} - 1)}}{\sqrt{5 - 4/m} - 1 + 2/m} \\
\to & \frac{1 + \sqrt{5} + 2\sqrt{2(\sqrt{5} - 1)}}{\sqrt{5} - 1} \\
= & \frac{3 + \sqrt{5}}{2} + \sqrt{2(\sqrt{5} + 1)} \\
\approx & 5.162.
\end{aligned}$$

□

4.4 Approximation algorithm II

In the first phase of Algorithm I, to solve the allotment problem the algorithm for the budget problem in [104] is applied. We observe that in the first step of the algorithm the "fixed" cost P is removed such that in the remaining step of the algorithm for each task J_j the "reduced" cost $\hat{c}_j(d_{j_k(j)}) = 0$. The "reduced" cost for each duration d_k is defined as $\hat{c}_j(d_k) = c_j(d_k) - c_j(d_{j_{k(j)}})$, where $d_{j_{k(j)}}$ is the maximal feasible duration of task J_j. It is worth noting that in the algorithm in [104] the rounding technique is only conducted for the linear relaxation of the two-duration instance with the "virtual" cost. Therefore the amount of the "fixed" cost does not change during the rounding procedure.

In this section, we carefully study the increase of the work in the algorithm in [104] by dividing the total work corresponding the integer solution to the discrete time-cost tradeoff problem (4.4) into rounded and non-rounded parts. Because the non-rounded part does not increase, we are able to develop an improved approximation algorithm. In our Algorithm II, the structure is essentially same as Algorithm I. However, the values of the rounding parameter ρ and the allotment parameter μ are different from those in Section 4.3 for Algorithm I (see Subsection 4.4.2).

First we have the following bound on the sum of the "fixed" cost:

Lemma 4.8 $|T_1| + |T_2| + |T_3| \leq P$.

Proof: Let us consider a "naive" schedule with a makespan exactly $P = \sum_{j=1}^{n} p_j(1)$. The "naive" schedule is constructed in the way that all tasks are scheduled on the same processor. In this "naive" schedule for each task the processing time is $p_j(1)$ and obviously its makespan is equal to or larger than the makespan of any other feasible schedule. Thus the lemma is proved. □

Furthermore, we can show the following relation between the total work W in the final solution and the fractional work W^* corresponding the optimal solution to (4.9):

Lemma 4.9 $W \leq \dfrac{1}{1-\rho} W^* - \dfrac{\rho}{1-\rho} P$.

Proof: For each task J_j, the number l of processors allotted in α is not more than the number l' of processors allotted in α'. According to Assumption 4.2, its work does not increase. Therefore $W \leq W'$. The total work W^* of the optimal solution to (4.9) is the sum of the "fixed" cost and the remaining unrounded part, i.e., $W^* = P + \sum_{j=1}^{n} \hat{w}_j(x_j^*)$. According to Proposition 4.1, $(W' - P) \leq (W^* - P)/(1-\rho)$. Therefore we have that

$$W \leq W' \leq \frac{W^* - P}{1-\rho} + P = \frac{1}{1-\rho} W^* - \frac{\rho}{1-\rho} P.$$

□

We define a piecewise work function $w_j(x)$ in fractional processing time x_j for task J_j based on (4.8) as follows: For $x_j \in [p_j(m), p_j(1)]$, if $l = \min\{k | k \in \{1, \ldots, m\}, p_j(k) = x_j\}$, then $w_j(x_j) = l p_j(l)$. If $x_j \in (p_j(l+1), p_j(l))$ and $l = 1, \ldots, m-1$, then

$$w_j(x_j) = \begin{cases} l p_j(l), & \text{if } x_j = p_j(l), l = 1, \ldots, m; \\ \dfrac{p_j(l) p_j(l+1)}{p_j(l) - p_j(l+1)} & \\ -\dfrac{(l+1) p_j(l+1) - l p_j(l)}{p_j(l) - p_j(l+1)} x_j, & \text{if } x_j \in (p_j(l+1), p_j(l)) \text{ and } l = 1, \ldots, m-1. \end{cases} \tag{4.21}$$

In allotments α and α', a task J_j is allotted l_j and l'_j processors, and their processing times are $p_j(l_j)$ and $p_j(l'_j)$, respectively. In the optimal (fractional) solution to (4.9), each task J_j has a fractional processing time x_j^*. We define the fractional number of processors allotted as follows:

$$l_j^* = w_j(x_j^*)/x_j^*. \tag{4.22}$$

According to this definition, l_j^* has the following property:

Lemma 4.10 *For any malleable task J_j, if $p_j(l+1) \leq x_j^* \leq p_j(l)$ for some $l \in \{1, \ldots, m-1\}$, then $l \leq l_j^* \leq l+1$.*

Proof: Suppose that $p_j(l+1) \leq x_j^* \leq p_j(l)$, according to (4.21), we can calculate the value of l_j^* as follows:

$$\begin{aligned} l_j^* = \frac{w(x_j^*)}{x_j^*} &= \frac{(l+1)p_j(l+1) - lp_j(l)}{p_j(l+1) - p_j(l)} - \frac{p_j(l)p_j(l+1)}{p_j(l+1) - p_j(l)} \frac{1}{x_j^*} \\ &= l + \frac{p_j(l+1)}{p_j(l) - p_j(l+1)} \left[\frac{p_j(l)}{x_j^*} - 1 \right]. \end{aligned} \tag{4.23}$$

Since $p_j(l) \geq x_j^*$, we have $p_j(l)/x_j^* \geq 1$ and $l_j^* \geq l$. From $p_j(l+1) \leq x_j^*$, by multiplying both sides by $p_j(l)$ and subtracting $x_j^* p_j(l+1)$ from both sides, we obtain that $[p_j(l) - x_j^*]p_j(l+1) \leq x_j^*[p_j(l) - p_j(l+1)]$, i.e.,

$$\frac{p_j(l+1)}{p_j(l) - p_j(l+1)} \left[\frac{p_j(l)}{x_j^*} - 1 \right] \leq 1.$$

Thus $l_j^* \leq l+1$ and the lemma is proved. □

Lemma 4.10 shows that l_j^* is well defined. We notice that l_j^* is just a notation and we only have knowledge that the real fractional number of processors corresponding to x_j^* should be in the interval $[l, l+1]$ if $p_j(l+1) \leq x_j^* \leq p_j(l)$. The notation l_j^* here fulfils this property and is convenient for our following analysis.

Same as in [81] or Section 4.3, in the final schedule, the time interval $[0, C_{\max}]$ consists of three types of time slots. In the first type of time slots, at most $\mu - 1$ processors are busy. In the second type of time slots, at least μ while at most $m - \mu$

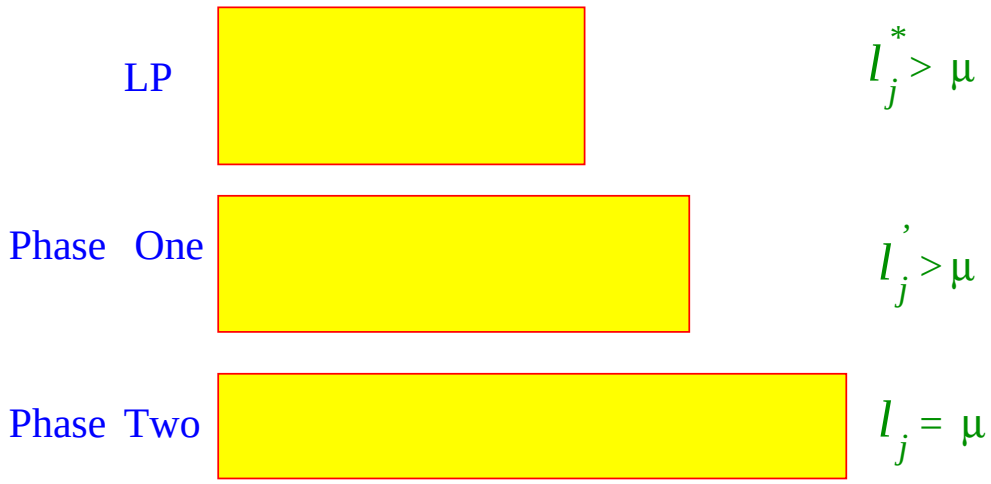


Figure 4.12: Change of processing time in the first subcase in Lemma 4.11.

processors are busy. In the third type at least $m-\mu+1$ processors are busy. Denote the sets of the three types time slots by T_1, T_2 and T_3, and $|T_i|$ the overall lengths for $i \in \{1,2,3\}$. In the case that $\mu = (m+1)/2$ for m odd, $T_2 = \emptyset$. In other cases all three types of time slots may exist. Then we have the following bound on $|T_1|$ and $|T_2|$:

Lemma 4.11 $\rho|T_1| + \min\{\mu/m, \rho\}|T_2| \leq C^*_{\max}$.

Proof: We also construct a "heavy" path $\mathcal{P}$ in the same way as in Lemma 4.2. Now we examine the stretch of processing time for all jobs in $\mathcal{P}$ in the rounding procedure of the first phase and in the new allotment α of the second phase.

For any job J_j in $T_1 \cap \mathcal{P}$, the processing time of the fractional solution to (4.9) increases by at most a factor $1/\rho$. The processing time does not change in the second phase as in α' the job J_j is only allotted a number $l'_j \leq \mu$ of processors such that it can be in the time slot of T_1. Therefore for such kind of jobs we have $p_j(l_j) = p_j(l'_j) \leq x^*_j/\rho$.

For any job J_j in $T_2 \cap \mathcal{P}$, there are two cases. In the first case, in α' a job J_j is allotted $l'_j \leq \mu$ processors. This is same as the case before and we also have $p_j(l_j) \leq x^*_j/\rho$. In the second case, in α' a job J_j is allotted $l'_j > \mu$ processors, and $l_j = \mu$. Then there are two subcases according to the solution to the linear program. In the first subcase, in the fractional solution to (4.9) there are $l^*_j \geq \mu$ processors

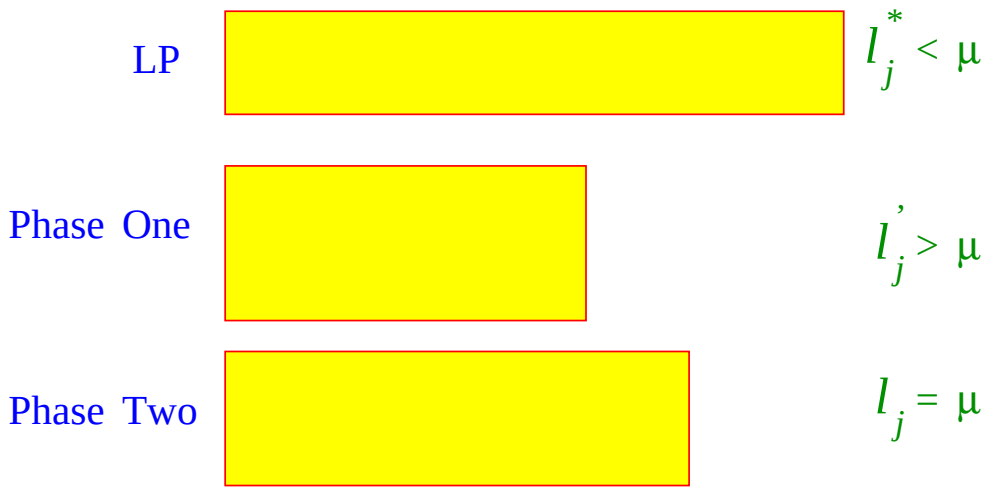


Figure 4.13: Change of processing time in the second subcase in Lemma 4.11.

allotted. Since μ is an integer, we have $l_j^* \geq \lfloor l_j^* \rfloor \geq \mu \geq l_j$. Then $l_j p_j(l_j) = W_j(l_j) \leq W_j(\mu) \leq W_j(\lfloor l_j^* \rfloor) \leq w_j(x_j^*) = l_j^* x_j^* \leq W_j(\lceil l_j^* \rceil)$ due to Assumption 4.2 and (4.22). Because $l_j^* \leq m$, and $w_j(x_j^*) = x_j^* l_j^* \geq p_j(l_j) l_j = W_j(l_j)$, it holds that $p_j(l_j) \leq x_j^* l_j^* / l_j \leq x_j^* m/\mu$. (See Figure 4.12). In the second subcase, in the fractional solution there are $l_j^* < \mu$ processors allotted to J_j. Then in the rounding procedure in the first phase the processing time must be rounded down from x_j^* to $p_j(l_j')$ as only in this way the assumption that $l_j' > \mu$ of this case can be satisfied. Then in the second phase, J_j is allotted μ processors and from Assumption 4.2 $p_j(l_j) l_j \leq p_j(l_j') l_j'$. Since there are at most m processors allotted to J_j in α', we have $p_j(l_j) \leq p_j(l_j') l_j'/l_j \leq p_j(l_j') m/\mu \leq x_j^* m/\mu$. Therefore for any job J_j in $T_2 \cap \mathcal{P}$, $p_j(l_j) \leq x_j^* \max\{1/\rho, m/\mu\}$ (See Figure 4.13).

Same as the argument in Lemma 4.2, the path $\mathcal{P}$ covers all time slots in the final schedule. In addition, in the schedule resulted from the fractional solution to the linear program, the jobs processed in T_1 in the final schedule contribute a total length of at least $\rho|T_1|$ to $L^*(\mathcal{P})$. In addition, the tasks processed in T_2 contribute a total length of at least $|T_2| \min\{\rho, \mu/m\}$ to $L^*(\mathcal{P})$. Since $L^*(\mathcal{P})$ is not more than the length $C_{\max}^*$ of the critical path in α^*, we have obtained the claimed inequality. □

We here use the same normalization technique and notations in Section 4.3, and denote by $x_p = P/C_{\max}^*$. Then the following lemma holds:

Lemma 4.12 *The optimal approximation ratio r of our Algorithm II is bounded by the optimal objective value of the following min-max nonlinear program:*

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2,x_3,x_p} \quad & x_1 + x_2 + x_3 \\ s.t. \quad & x_1 + x_2 + x_3 \leq x_p; \\ & x_1 + \mu x_2 + (m - \mu + 1)x_3 \leq \frac{1}{1-\rho} m - \frac{\rho}{1-\rho} x_p; \\ & \rho x_1 + \min\{\mu/m, \rho\} x_2 \leq 1; \\ & x_1, x_2, x_3, x_p \geq 0; \\ & \rho \in (0, 1); \\ & \mu \in \left\{1, \ldots, \left\lfloor \frac{m+1}{2} \right\rfloor \right\}. \end{aligned} \tag{4.24}$$

Proof: First, by multiplying both sides of the inequality in the Lemma 4.8 by the factor $1/C^*_{\max}$, it holds that $x_1 + x_2 + x_3 \leq x_p$. Furthermore, since in time slots T_1, T_2 and T_3, there are at least 1, μ and $m - \mu + 1$ processors are busy, respectively, the total work $W \geq |T_1| + \mu|T_2| + (m - \mu + 1)|T_3|$. According to (4.12), we have that $W^* \leq mC^*_{\max}$. Together with the definitions of the normalized notations, Lemma 4.9 leads to $x_1 + \mu x_2 + (m - \mu + 1)x_3 \leq \frac{1}{1-\rho} m - \frac{\rho}{1-\rho} x_p$. Then the Lemma 4.11 leads to the third constraint $\rho x_1 + \min\{\mu/m, \rho\} x_2 \leq 1$. All other constraints are based on the definitions of ρ and μ. Since the total makespan $C_{\max} = |T_1| + |T_2| + |T_3|$ according to (4.13), the approximation ratio

$$r = \sup \frac{C_{\max}}{OPT} \leq \sup \frac{C_{\max}}{C^*_{\max}} = \max_{x_1,x_2,x_3} x_1 + x_2 + x_3,$$

which is our objective function and our goal is to minimize the approximation ratio over all feasible μ and ρ. Thus the lemma is proved. □

We then can simplify the min-max nonlinear program (4.24) as follows:

Lemma 4.13 *The min-max nonlinear program (4.24) is equivalent to the following nonlinear program:*

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \quad & \frac{x_1(m-\mu)(1-\rho)+x_2(1-\rho)(m-2\mu+1)+m}{(m-\mu)(1-\rho)+1} \\ s.t. \quad & \rho x_1 + \min\left\{\rho, \frac{\mu}{m}\right\} x_2 \le 1; \\ & x_1 + x_2(\mu(1-\rho)+\rho) \le m; \\ & x_1, x_2 \ge 0; \\ & \rho \in (0,1); \\ & \mu \in \left\{1, \ldots, \left\lfloor \frac{m+1}{2} \right\rfloor\right\}. \end{aligned} \tag{4.25}$$

Proof: Substituting the first constraint $x_1 + x_2 + x_3 \le x_p$ in (4.24) to the second constraint with simplification, we have

$$x_1 + x_2(\mu(1-\rho)+\rho) + x_3((m-\mu)(1-\rho)+1) \le m. \tag{4.26}$$

We then have eliminated the additional variable x_p. Since (4.24) is linear in x_i, to find the optimum we can set

$$x_3 = [m - x_1 - x_2(\mu(1-\rho)+\rho)]/[(m-\mu)(1-\rho)+1] \ge 0$$

because for a fixed pair of x_1 and x_2 we want to maximize x_3, so do for the objective function. Then constraint (4.26) can be removed and the objective function is replaced by

$$x_1 + x_2 + x_3 = \frac{x_1(m-\mu)(1-\rho)+x_2(1-\rho)(m-2\mu+1)+m}{(m-\mu)(1-\rho)+1}.$$

However, an additive constraint

$$x_1 + x_2(\mu(1-\rho)+\rho) \le m$$

is needed to guarantee that $x_3 \ge 0$, i.e., $|T_1| + |T_2| \le C_{\max}$. By these means the new min-max nonlinear program (4.25) can be obtained. □

In the following we will solve the min-max nonlinear program (4.25) to get the approximation ratio of Algorithm II.

4.4.1 Analysis of the min-max nonlinear program (4.25)

In order to solve (4.25), we need to consider two cases that either $\rho < \mu/m$ or $\rho \geq \mu/m$ to simplify the first constraint. Furthermore, there are also relations between the first and the second constraints. Therefore we need to solve (4.25) by case study.

4.4.1.1 Solve (4.25) for the case $\rho \leq \mu/m$

In this case, we need to solve the following min-max nonlinear program:

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \quad & \frac{x_1(m-\mu)(1-\rho) + x_2(1-\rho)(m-2\mu+1) + m}{(m-\mu)(1-\rho)+1} \\ \text{s.t.} \quad & \rho x_1 + \rho x_2 \leq 1; \\ & x_1 + x_2(\mu(1-\rho)+\rho) \leq m; \\ & x_1, x_2 \geq 0; \\ & \rho \in (0, \mu/m); \\ & \mu \in \left\{1, \ldots, \left\lfloor \frac{m+1}{2} \right\rfloor \right\}. \end{aligned} \tag{4.27}$$

Define by C_1 the first constraint $\rho x_1 + \rho x_2 \leq 1$, by C_2 the second constraint $x_1 + x_2(\mu(1-\rho)+\rho) \leq m$, and by C_3 the third constraint $x_1, x_2 \geq 0$. Same as the analysis in Section 4.3, for a fixed pair ρ and μ, we need to search for the maximum objective value over all x_1 and x_2 constrained by C_1, C_2 and C_3. From C_1 and C_3 the three extreme points are $E_1 : (x_1, x_2) = (1/\rho, 0)$, $E_2 : (x_1, x_2) = (0, 1/\rho)$ and $E_5 : (x_1, x_2) = (0, 0)$. From C_2 and C_3 the three extreme points are $E_3 : (x_1, x_2) = (m, 0)$, $E_4 : (0, m/(\mu(1-\rho)+\rho))$ and $E_5 : (x_1, x_2) = (0, 0)$. Therefore we need to study several cases with different relations between ρ and μ.

Solving equation $\dfrac{1}{\rho} = \dfrac{m}{\mu(1-\rho)+\rho}$ for ρ yields a unique solution $\rho = \dfrac{\mu}{m+\mu-1}$. Since $\mu \geq 1$, $(m-1)\mu \geq m-1$, i.e., $m+\mu-1 \leq m\mu$. Thus $\dfrac{1}{m} \leq \dfrac{\mu}{m+\mu-1}$. So there are following three cases:

- **CASE 1**: $\rho \in \left(\dfrac{\mu}{m+\mu-1}, \dfrac{\mu}{m}\right]$;

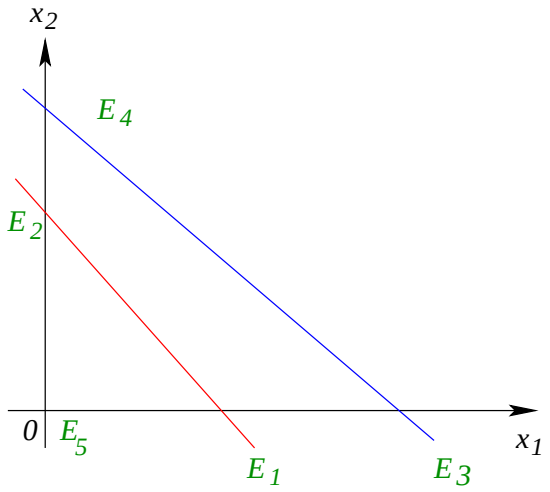


Figure 4.14: Polytope of **CASE 1**.

- **CASE 2**: $\rho \in \left(\frac{1}{m}, \frac{\mu}{m+\mu-1}\right]$;
- **CASE 3**: $\rho \in \left(0, \frac{1}{m}\right]$.

We now consider **CASE 1**. In this case, $\rho > 1/m$ so $1/\rho < m$. In addition, $\rho > \mu/(m+\mu-1)$ so $1/\rho < m/(\mu(1-\rho)+\rho)$. Therefore the polytope defined by E_1, E_2 and E_5 is inside the polytope defined by E_3, E_4 and E_5 as in Figure 4.14. So the constraint C_2 can be removed. It is obvious that the maximum value can not be attained on the extreme point E_5. Similar to Section 4.3, we denote by $A(\mu,\rho)$ and $B(\mu,\rho)$ the objective values at the two extreme points E_1 and E_2, respectively. By simple calculation we have:

$$\begin{aligned} A(\mu,\rho) &= \frac{(m-\mu)(1-\rho)+\rho m}{\rho(m-\mu)(1-\rho)+\rho}; \\ B(\mu,\rho) &= \frac{(m-2\mu+1)(1-\rho)+\rho m}{\rho(m-\mu)(1-\rho)+\rho}. \end{aligned}$$

Since $\mu \geq 1$, $m-\mu \geq m-2\mu+1$. Then we have $A(\mu,\rho) \geq B(\mu,\rho)$. The problem is now to minimize $A(\mu,\rho)$ over all feasible μ and ρ as follows:

$$\begin{aligned} \min_{\mu,\rho} \quad & A(\mu,\rho) = \frac{(m-\mu)(1-\rho)+\rho m}{\rho(m-\mu)(1-\rho)+\rho} \\ \text{s.t.} \quad & \rho \in \left(\frac{\mu}{m+\mu-1}, \frac{\mu}{m}\right); \\ & \mu \in \left\{1, \ldots, \left\lfloor \frac{m+1}{2} \right\rfloor\right\}. \end{aligned}$$

The first order partial derivative of $A(\mu,\rho)$ with respect to μ is:

$$\begin{aligned} A'(\mu,\rho)_\mu &= \frac{-(1-\rho)[(m-\mu)(1-\rho)+1]+(1-\rho)[(m-\mu)(1-\rho)+\rho m]}{\rho[(m-\mu)(1-\rho)+1]^2} \\ &= \frac{(1-\rho)(\rho m-1)}{\rho[(m-\mu)(1-\rho)+1]^2}. \end{aligned}$$

The only root to equation $A'(\mu,\rho)_\mu = 0$ is $\rho = 1/m$ as $\rho \leq \mu/m < 1$. However, it is infeasible in **CASE 1** as it is required that $\rho > 1/m$. Therefore we have $A'(\mu,\rho)_\mu > 0$, i.e., $A(\mu,\rho)$ is increasing in μ. As we are going to minimize $A(\mu,\rho)$, we can set $\mu^* = 1$. Then the value of $A(\rho)$ is as follows:

$$A(\rho) = \frac{(m-1)(1-\rho)+\rho m}{\rho(m-1)(1-\rho)+\rho}.$$

We notice that in **CASE 1**, $\mu/(m+\mu-1) = 1/m$, and also $\mu/m = 1/m$. Therefore it is fixed that $\rho^* = 1/m$. Then the optimal approximation ratio is

$$r = A(\rho)|_{\rho=1/m} = \frac{(m-1)(1-1/m)+1}{(m-1)(1-1/m)/m+1/m} = m.$$

It shows that the bound on the approximation ratio r would be m if we choose the parameters proposed above in **CASE 1**. However, in our algorithm we will not choose the parameters in this way as other choice gives a better bound on the ratio.

Now we turn to **CASE 3**. Since $\rho \leq 1/m \leq \mu/(m+\mu-1)$, $1/\rho \geq m/(\mu(1-\rho)+\rho)$. In addition, $1/\rho \geq m$. These show that the polytope defined by E_1, E_2 and E_5 covers the polytope defined by E_3, E_4 and E_5 as in Figure 4.15. Hence the constraint C_1 can be removed. We can skip the extreme point E_5 as it does not corresponds to the

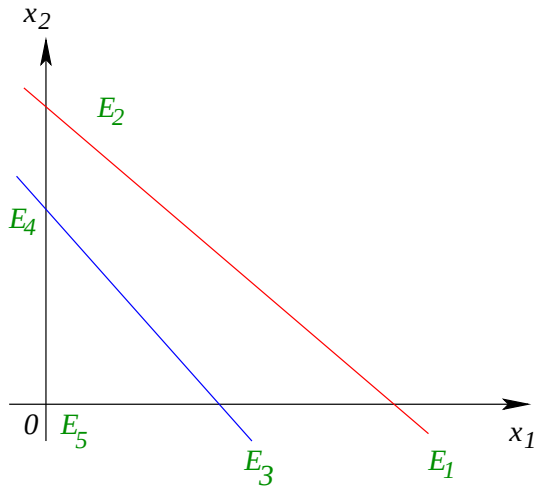


Figure 4.15: Polytope of **CASE 3**.

maximum objective value. We denote by $A(\mu,\rho)$ and $B(\mu,\rho)$ the objective values at the two extreme points E_3 and E_4, respectively. Therefore:

$$\begin{aligned} A(\mu,\rho) &= \frac{m(m-\mu)(1-\rho)+m}{(m-\mu)(1-\rho)+1} = m; \\ B(\mu,\rho) &= \frac{\dfrac{m(1-\rho)(m-2\mu+1)}{\mu(1-\rho)+\rho}+m}{(m-\mu)(1-\rho)+1}. \end{aligned}$$

Since $\mu \geq 1$, we have

$$\mu(1-\rho)+\rho \geq 1-\rho+\rho = 1,$$

and

$$m-2\mu+1 \leq m-\mu \leq (m-\mu)[\mu(1-\rho)+\rho].$$

Therefore

$$\frac{(1-\rho)(m-2\mu+1)}{\mu(1-\rho)+\rho}+1 \leq (m-\mu)(1-\rho)+1.$$

Then we are able to conclude that

$$B(\mu,\rho) = \frac{\dfrac{m(1-\rho)(m-2\mu+1)}{\mu(1-\rho)+\rho}+m}{(m-\mu)(1-\rho)+1}$$

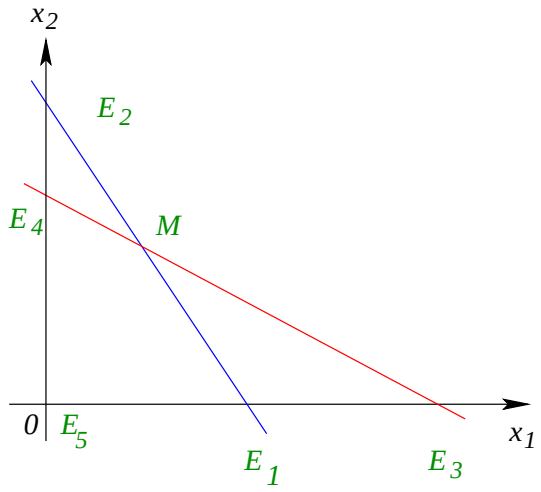


Figure 4.16: Polytope of **CASE 2**.

$$\leq \frac{m[(m-\mu)(1-\rho)+1]}{(m-\mu)(1-\rho)+1} = m = A(\mu,\rho).$$

Therefore in **CASE 3** for any feasible ρ and $\mu > 1$ the objective value (the upper bound on approximation ratio r) is always m. Furthermore, we need to consider the case $\mu = 1$. In this case, it holds that $B(\mu,\rho) = m = A(\mu,\rho)$. Therefore the approximation ratio of **CASE 3** is bounded by m.

We then study **CASE 2**. In this case $\frac{1}{m} \leq \rho \leq \frac{\mu}{m+\mu-1}$. So $1/\rho \leq m$ and $\frac{1}{\rho} \geq \frac{m}{\mu(1-\rho)+\rho}$. Therefore the straight lines (E_1, E_2) and (E_3, E_4) intersect at a point $M : (x_{1M}, x_{2M})$ (See Figure 4.16). Since the slopes of the two straight lines (E_1, E_2) and (E_3, E_4) are -1 and $\frac{-1}{\mu(1-\rho)+\rho}$, the coordinates x_{1M} and x_{2M} of the intersection point M can be calculated by the following system of equations:

$$\begin{cases} x_{2M} - \frac{1}{\rho} = -1 \cdot x_{1M}; \\ x_{2M} - \frac{m}{\mu(1-\rho)+\rho} = -\frac{1}{\mu(1-\rho)+\rho} \cdot x_{1M}. \end{cases}$$

If $\mu > 1$, then the solutions to the above system is

$$x_{1M} = \frac{(1-\rho)\mu - \rho(m-1)}{\rho(1-\rho)(\mu-1)},$$

$$x_{2M} = \frac{\rho m - 1}{\rho(1-\rho)(\mu-1)}.$$

We now examine the feasibility of M. It is clear that $\rho(1-\rho)(\mu-1) > 0$ for $\mu > 1$. Since $1/m \leq \rho$, $\rho m \geq 1$. So $x_{2M} \geq 0$. Because $\rho \leq \frac{\mu}{m+\mu-1}$, we have $(1-\rho)\mu - \rho(m-1) = \mu - \rho(m+\mu-1) \geq 0$. Thus $x_{1M} \geq 0$ and M is feasible. Therefore the polytope defined by the constraints has four extreme points E_2, E_3, E_5 and M. Same as before, we can skip the analysis for E_5. Denote by $A(\mu,\rho)$, $B(\mu,\rho)$ and $D(\mu,\rho)$ the objective values at E_2, E_3 and M, respectively. We have

$$\begin{aligned}
A(\mu,\rho) &= \frac{(m-\mu)(1-\rho)+\rho m}{\rho(m-\mu)(1-\rho)+\rho};\\
B(\mu,\rho) &= \frac{\frac{m(1-\rho)(m-2\mu+1)}{\mu(1-\rho)+\rho}+m}{(m-\mu)(1-\rho)+1};\\
D(\mu,\rho) &= \frac{1}{(m-\mu)(1-\rho)+1}\left\{\frac{(1-\rho)\mu-\rho(m-1)}{\rho(1-\rho)(\mu-1)}(m-\mu)(1-\rho)\right.\\
&\quad \left.+\frac{\rho m-1}{\rho(1-\rho)(\mu-1)}(1-\rho)(m-2\mu+1)+m\right\}\\
&= \frac{m\mu-\mu^2+\rho\mu^2+\rho m-\rho\mu-\rho m\mu-m+2\mu-1}{\rho(\mu-1)[(m-\mu)(1-\rho)+1]}\\
&= \frac{m-\rho m+1-\mu+\rho\mu}{\rho[(m-\mu)(1-\rho)+1]} = \frac{1}{\rho}.
\end{aligned}$$

Since in this case $\rho m \geq 1$, $\frac{(m-\mu)(1-\rho)+\rho m}{(m-\mu)(1-\rho)+1} \geq 1$, and then $A(\mu,\rho) \geq 1/\rho = D(\mu,\rho)$. Therefore we do not need to consider the extreme point M any more. As showed in the analysis of **CASE 1**, $A(\mu,\rho)$ is increasing in μ, we can set $\mu^* = 1$ to attain the minimum of $A(\mu,\rho)$, and in this case $\rho^* = 1/m$. Therefore when $\mu^* = 1$ and $\rho^* = 1/m$, $A(\mu,\rho)$ is minimized and the minimum value is m. This shows that in this case $A(\mu,\rho) \geq m$. In addition, according to the analysis of **CASE 3**, $B(\mu,\rho) \leq m$ holds for all feasible μ and ρ. Because $A(\mu,\rho) \geq m$ and $B(\mu,\rho) \leq m$, in this case $A(\mu,\rho) \geq B(\mu,\rho)$ holds and the minimum value of $A(\mu,\rho)$ is the bound on the approximation ratio, which means $r \leq m$.

In addition, we still need to study the case that $\mu = 1$ in **CASE 2**. Now since $\mu/(m+\mu-1) = 1/m$, $\rho = 1/m$ is fixed. Thus E_1 coincides E_3 while E_2 coincides E_4, i.e., the straight lines (E_1, E_2) and (E_3, E_4) coincide. Therefore at any point on the straight lines the value of the objective function is always m. Hence the approximation ratio is still upper bounded by m.

Combining all three cases, we have the following lemma as the optimal objective value of (4.27) is the approximation ratio of Algorithm II:

Lemma 4.14 *In the case that $\rho \leq \mu/m$, Algorithm II has an approximation ratio $r \leq m$.*

4.4.1.2 Solve (4.25) for the case $\rho > \mu/m$

In this case, we will solve the following min-max nonlinear program:

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \quad & \frac{x_1(m-\mu)(1-\rho) + x_2(1-\rho)(m-2\mu+1) + m}{(m-\mu)(1-\rho)+1} \\ \text{s.t.} \quad & \rho x_1 + \mu x_2/m \leq 1; \\ & x_1 + x_2(\mu(1-\rho)+\rho) \leq m; \\ & x_1, x_2 \geq 0; \\ & \rho \in (\mu/m, 1); \\ & \mu \in \left\{1, \ldots, \left\lfloor \frac{m+1}{2} \right\rfloor \right\}. \end{aligned} \tag{4.28}$$

Denote by C_1 the first constraint $\rho x_1 + \mu x_2/m \leq 1$, by C_2 the second constraint $x_1 + x_2(\mu(1-\rho)+\rho) \leq m$, and by C_3 the third constraint $x_1, x_2 \geq 0$ in (4.28). With a fixed pair ρ and μ we will search for the maximum objective value over all x_1 and x_2 constrained by C_1, C_2 and C_3. From C_1 and C_3 the three extreme points are $E_1 : (x_1, x_2) = (1/\rho, 0)$, $E_2 : (x_1, x_2) = (0, m/\mu)$ and $E_5 : (x_1, x_2) = (0, 0)$. From C_2 and C_3 the three extreme points are $E_3 : (x_1, x_2) = (m, 0)$, $E_4 : (x_1, x_2) = (0, m/(\mu(1-\rho)+\rho))$ and $E_5 : (x_1, x_2) = (0, 0)$.

Since $\rho > 0$ and $\mu \geq 1$, $\rho \leq \mu\rho$, i.e., $\mu(1-\rho)+\rho \leq \mu$. So $m/\mu \leq m/(\mu(1-\rho)+\rho)$. Therefore E_2 is in the polytope defined by E_3, E_4 and E_5. Furthermore, since in this case $\rho \geq \mu/m \geq 1/m$, so $1/\rho \leq m$. Thus the extreme point E_1 is also in the polytope defined by E_3, E_4 and E_5. This shows that the polytope defined by E_3, E_4

and E_5 completely covers the polytope defined by E_1, E_2 and E_5 (See Figure 4.14). Hence the constraint C_2 can be removed, and we just need to consider the following min-max nonlinear program:

$$\begin{array}{rl} \min_{\mu,\rho} \max_{x_1,x_2} & \dfrac{x_1(m-\mu)(1-\rho)+x_2(1-\rho)(m-2\mu+1)+m}{(m-\mu)(1-\rho)+1} \\ \text{s.t.} & \rho x_1+\mu x_2/m \le 1; \\ & x_1, x_2 \ge 0; \\ & \rho \in (\mu/m, 1); \\ & \mu \in \left\{1, \ldots, \left\lfloor \dfrac{m+1}{2} \right\rfloor \right\}. \end{array} \tag{4.29}$$

The next step is to solve (4.29). Similar as before, we can skip the extreme point E_5 and denote by $A(\mu,\rho)$ and $B(\mu,\rho)$ the objective values at extreme points E_1 and E_2, respectively. Simple calculation yields:

$$\begin{aligned} A(\mu,\rho) &= \frac{(m-\mu)(1-\rho)+\rho m}{\rho(1-\rho)(m-\mu)+\rho} = \frac{1}{\rho} + \frac{\rho m - 1}{\rho(1-\rho)(m-\mu)+\rho}; \\ B(\mu,\rho) &= \frac{(1-\rho)m(m-2\mu+1)+m\mu}{\mu(1-\rho)(m-\mu)+\mu}. \end{aligned}$$

As $\rho m > \mu \ge 1$, it is obvious that $A(\mu,\rho)$ is increasing in μ. Now we consider $B(\mu,\rho)$. Its partial derivative with respect to μ is:

$$\begin{aligned} B'(\mu,\rho)_\mu &= \frac{1}{[\mu(1-\rho)(m-\mu)+\mu]^2}\{[-2(1-\rho)m+m][\mu(1-\rho)(m-\mu)+\mu] \\ &\quad -[(1-\rho)m(m-2\mu+1)+m\mu][(1-\rho)(m-2\mu)+1]\} \\ &= \frac{m(1-\rho)}{[(1-\rho)\mu(m-\mu)+\mu]^2}\{(2\rho-1)\mu^2 \\ &\quad +2(1-\rho)(m+1)\mu-(m+1)[(1-\rho)m+1]\}. \end{aligned}$$

If $\rho = 1/2$, then

$$B'(\mu,\rho)_\mu|_{\rho=1/2} = \frac{2m}{[\mu(m-\mu)+2\mu]^2}\left[(m+1)\mu-(m+1)\left(\frac{m}{2}+1\right)\right]$$

$$= \frac{m(m+1)(2\mu - m - 2)}{[\mu(m-\mu)+2\mu]^2}.$$

Because $\mu \leq (m+1)/2 < m/2+1$, $B'(\mu,\rho)_\mu < 0$, and $B(\mu,\rho)$ is decreasing in μ. Otherwise ($\rho \neq 1/2$), we need to check the existence of roots to equation $B'(\mu,\rho)_\mu = 0$. Since $\rho < 1 < m/(m-1)$ for $m \geq 2$, $\rho(m-1) < m$, or $\rho(m+1) > 2\rho m - m$, which yields $\rho^2(m+1)^2 - \rho(2\rho-1)m(m+1) > 0$. So there exist two roots to $B'(\mu,\rho)_\mu = 0$. Solving equation $B'(\mu,\rho)_\mu = 0$, we get the following solutions by Proposition 4.2:

$$\begin{aligned}
\mu_{1,2} &= \frac{1}{2(2\rho-1)}\left[-2(1-\rho)(m+1)\right. \\
&\quad \left. \mp\sqrt{4(1-\rho)^2(m+1)^2 - 4(2\rho-1)(m+1)[(1-\rho)m+1]}\right] \\
&= \frac{(1-\rho)(m+1) \mp \sqrt{\rho^2(m+1)^2 + \rho(1-2\rho)m(m+1)}}{1-2\rho}
\end{aligned}$$

We have now two cases as follows:

- **CASE 1**: $\rho \in (\mu/m, 1/2)$;
- **CASE 2**: $\rho \in (1/2, 1)$.

We now consider **CASE 1**. In this case $1-2\rho > 0$, so $\mu_1 < \mu_2$. It is clear that

$$(1-2\rho)m + 1 + 2\rho > 0,$$

i.e.,

$$(1+2\rho)(m+1) > 4\rho m.$$

Multiply both sides by the positive factor $(1-2\rho)(m+1)$,

$$(1-4\rho^2)(m+1)^2 > 4\rho(1-2\rho)m(m+1).$$

Thus it holds that

$$(m+1)^2 > 4\rho^2(m+1)^2 + 4\rho(1-2\rho)m(m+1).$$

Taking the square root of both sides yields

$$m+1 > 2\sqrt{\rho^2(m+1)^2 + \rho(1-2\rho)m(m+1)}.$$

Then we have

$$(2-2\rho)(m+1) - 2\sqrt{\rho^2(m+1)^2 + \rho(1-2\rho)m(m+1)} > (1-2\rho)(m+1).$$

Since $1-2\rho > 0$,

$$\mu_1 = \frac{(1-\rho)(m+1) - \sqrt{\rho^2(m+1)^2 + \rho(1-2\rho)m(m+1)}}{1-2\rho} > \frac{m+1}{2}.$$

In addition, $\mu_2 > \mu_1 > (m+1)/2$, too. The values of $\mu_{1,2}$ violate the constraint that $\mu \leq (m+1)/2$. Therefore in the feasible domain of μ there is no root to equation $B'(\mu,\rho)_\mu = 0$. As $2\rho - 1 < 0$, according to Proposition 4.3, $B'(\mu,\rho)_\mu < 0$ for $\mu \leq (m+1)/2$. Hence we conclude that $B(\mu,\rho)$ is decreasing in μ in **CASE 1**.

In **CASE 2**, $1-2\rho < 0$, so $\mu_2 < 0$ is infeasible. We consider the other root

$$\mu_1 = \frac{\sqrt{\rho^2(m+1)^2 + \rho(1-2\rho)m(m+1)} - (1-\rho)(m+1)}{2\rho - 1}$$

Since $\rho \in (0,1)$,

$$(1-\rho)m + \rho \geq 0.$$

Multiplying both sides with a negative factor $1-2\rho$ results in the following inequality:

$$(1-3\rho+2\rho^2)m \leq 2\rho^2 - \rho,$$

i.e.,

$$\rho m - 2\rho^2 m + \rho \geq m - 2\rho m + 2\rho - 2\rho^2.$$

Multiply both sides by m and add $\rho^2 m^2 + \rho^2$ to both sides:

$$\rho^2(m+1)^2 + \rho(1-2\rho)m(m+1) \geq (1-\rho)^2 m^2 + 2(1-\rho)\rho m + \rho^2 = [(1-\rho)m + \rho]^2.$$

Take the square root of both sides:

$$\sqrt{\rho^2(m+1)^2+\rho(1-2\rho)m(m+1)} \geq (1-\rho)m+\rho.$$

Subtracting both side by $(1-\rho)(m+1)$ and dividing both sides by the positive factor $2\rho-1$ we obtain:

$$\mu_1 = \frac{\sqrt{\rho^2(m+1)^2+\rho(1-2\rho)m(m+1)}-(1-\rho)(m+1)}{2\rho-1} \geq 1.$$

In order to check whether μ_1 satisfies the constraint that $\mu \leq (m+1)/2$, we need to consider the following two subcases:

- **CASE 2.1**: $\rho \in (1/2, 1/2+1/(m-1))$;
- **CASE 2.2**: $\rho \in [1/2+1/(m-1), 1)$.

We first consider **CASE 2.1**. From the assumption $\rho < 1/2+1/(m-1)$ we have $2(m-1)\rho < m+1$, or $2(m+1)\rho+m+1 > 4m\rho$. Multiplying both sides by a positive factor $(2\rho-1)(m+1)$ yields

$$(4\rho^2-1)(m+1)^2 > 4\rho(2\rho-1)m(m+1),$$

i.e.,

$$4[\rho^2(m+1)^2+\rho(1-2\rho)m(m+1)] > (m+1)^2.$$

Take the square root of both sides:

$$2\sqrt{\rho^2(m+1)^2+\rho(1-2\rho)m(m+1)} > m+1 = (2\rho-1)(m+1)+2(1-\rho)(m+1).$$

Subtracting both sides by $2(1-\rho)(m+1)$ and dividing both sides by a positive factor $2(2\rho-1)$ leads to:

$$\mu_1 = \frac{\sqrt{\rho^2(m+1)^2+\rho(1-2\rho)m(m+1)}-(1-\rho)(m+1)}{2\rho-1} > \frac{m+1}{2}.$$

Therefore in **CASE 2.1** two roots to the equation $B'(\mu,\rho)_\mu = 0$ are $\mu_1 > (m+1)/2$ and $\mu_2 < 0$. In addition, the coefficient of the term of μ^2 in $B'(\mu,\rho)_\mu = 0$ is

$2\rho - 1 > 0$. So for any feasible $1 \leq \mu \leq (m+1)/2$, $B'(\mu,\rho)_\mu < 0$ according to Proposition 4.3, and $B(\mu,\rho)$ is decreasing in μ.

In **CASE 2.2**, with the similar argument as the analysis in **CASE 2.1**, $\mu_1 \leq (m+1)/2$ is feasible. Therefore in this case $B(\mu,\rho)$ is decreasing in μ when $\mu \in [1,\mu_1]$ and is increasing in μ when $\mu \in (\mu_1,(m+1)/2]$ according to Proposition 4.3. Since $A(\mu,\rho)$ is increasing in μ, there are four possibilities of the relations between $A(\mu,\rho)$ and $B(\mu,\rho)$:

- **CASE 2.2.1**: $A(\mu,\rho) < B(\mu,\rho)$ for all feasible μ;
- **CASE 2.2.2**: Equation $A(\mu,\rho) = B(\mu,\rho)$ has one root for all feasible μ;
- **CASE 2.2.3**: Equation $A(\mu,\rho) = B(\mu,\rho)$ has two roots for all feasible μ;
- **CASE 2.2.4**: $A(\mu,\rho) > B(\mu,\rho)$ for all feasible μ.

In **CASE 2.2.1**, equation $A(\mu,\rho) = B(\mu,\rho)$ has no root and for $\mu = (m+1)/2$, $A(\mu,\rho) < B(\mu,\rho)$ (See Figure 4.6). In **CASE 2.2.2**, for $\mu = 1$, $A(\mu,\rho) < B(\mu,\rho)$ and for $\mu = (m+1)/2$, $A(\mu,\rho) > B(\mu,\rho)$, or vice versa (See Figure 4.7, 4.8 and 4.9). In **CASE 2.2.3**, for $\mu = 1$ and $\mu = (m+1)/2$, $A(\mu,\rho) < B(\mu,\rho)$, and equation $A(\mu,\rho) = B(\mu,\rho)$ has two roots (See Figure 4.10). In **CASE 2.2.4**, equation $A(\mu,\rho) = B(\mu,\rho)$ has no root and for $\mu = 1$, $A(\mu,\rho) > B(\mu,\rho)$ (See Figure 4.11). Now we examine the existence of roots to the equation $A(\mu,\rho) = B(\mu,\rho)$. It is

$$\mu^2 - (1+2\rho)m\mu + m^2\rho + m\rho = 0.$$

Since in our assumption $m\rho > \mu \geq 1$, we have $4\rho(m\rho - 1) + m > 0$, i.e., $(1 + 4\rho^2)m^2 - 4\rho m > 0$. Solving equation $A(\mu,\rho) = B(\mu,\rho)$ by Proposition 4.2 we have

$$\begin{aligned} \mu &= \frac{(1+2\rho)m \mp \sqrt{(1+2\rho)^2m^2 - 4(m^2\rho + m\rho)}}{2} \\ &= \frac{(1+2\rho)m \mp \sqrt{(1+4\rho^2)m^2 - 4\rho m}}{2}. \end{aligned}$$

Again, from $m\rho > 1$ we have

$$\frac{(1+2\rho)m+\sqrt{(1+4\rho^2)m^2-4\rho m}}{2} > \frac{(1+2\rho)m}{2} \geq \frac{m+2}{2} > \frac{m+1}{2},$$

which violates the constraint of μ. Now we examine the feasibility of another root. Since $m\rho > 1$ and $m \geq 1$, $(m-1)(\rho m-1) \geq 0$, i.e., $1+\rho m^2-m-\rho m \geq 0$. Multiplying both sides by 4 and adding $m^2+4\rho^2m^2$ to both sides yields $m^2+4\rho^2m^2+4+4\rho m^2-4m-8\rho m \geq m^2+4\rho^2m^2-4\rho m$. Taking the square root of both sides we have $m+2\rho m-2 \geq \sqrt{(1+4\rho^2)m^2-4\rho m}$, which is equivalent to

$$\frac{(1+2\rho)m-\sqrt{(1+4\rho^2)m^2-4\rho m}}{2} \geq 1.$$

In addition, since $m \geq 1$, we have $m^2 \geq 1$. Adding $4\rho m(\rho m-1)$ to both sides results in $(2\rho m-1)^2 \leq (1+4\rho^2)m^2-4\rho m$. Since in our assumption $\rho m-1>0$, by taking square root of both sides we obtain that $(2\rho m-1) \leq \sqrt{(1+4\rho^2)m^2-4\rho m}$, which is equivalent to

$$\frac{(1+2\rho)m-\sqrt{(1+4\rho^2)m^2-4\rho m}}{2} \leq \frac{m+1}{2}.$$

Hence, in our **CASE 2.2**, there does always exist one and only one feasible root

$$\mu^* = \frac{(1+2\rho)m-\sqrt{(1+4\rho^2)m^2-4\rho m}}{2}. \tag{4.30}$$

to the equation $A(\mu,\rho) = B(\mu,\rho)$. So neither **CASE 2.2.1**, **CASE 2.2.3** nor **CASE 2.2.4** is feasible. We now consider **CASE 2.2.2**. In Figure 4.7, the optimal value of (4.15) exists at the point $\mu = \mu^*$. However, in Figure 4.8 and 4.9, the optimal value exists at the point $\mu = \mu_1$. Here we just need to search for an upper bound of the optimal value. Therefore in this case we still count the objective value at the point $\mu = \mu^*$ as the upper bound. Thus we have finished the analysis for **CASE 2.2**.

Now we return to analyze **CASE 1** and **CASE 2.1**. In both cases $A(\mu,\rho)$ is increasing in μ while $B(\mu,\rho)$ is decreasing. According to Lemma 4.7, to obtain

the minimum of the objective value of (4.29), we need to solve equation $A(\mu, \rho) = B(\mu, \rho)$. With the same argument as the analysis for **CASE 2.2** we obtain the same root μ^* as in (4.30). We need to check the feasibility of this root. It is obvious that we can use the same technique for **CASE 2.2** to prove that $\mu^* \geq 1$ and $\mu^* \leq (m+1)/2$ for both **CASE 1** and **CASE 2.1**. Hence we have the following Lemma:

Lemma 4.15 *For a fixed $\rho \in (\mu/m, 1)$, the optimal objective value of (4.29) is bounded by the optimal objective value for μ^* in (4.30).*

In this way, we just need to substitute μ^* in (4.30) to either $A(\mu, \rho)$ or $B(\mu, \rho)$, and minimize it over all $\rho \geq \mu^*/m$ to obtain the approximation ratio of our Algorithm II, same as the analysis for Algorithm I. We leave the analysis of approximation ratio and of setting of parameters to the Subsection 4.4.2.

4.4.2 Approximation ratio of Algorithm II

According to the analysis in Subsubsection 4.4.1.1, when $\rho \leq \mu/m$, by setting $\mu^* = 1$ and $\rho^* = 1/m$ we are able to obtain the optimal value of (4.25), i.e., the approximation ratio of Algorithm II. The ratio is m based on Lemma 4.14.

Now we investigate the case $\rho > \mu/m$ based on Subsubsection 4.4.1.2. Unfortunately, we will show in Subsection 4.4.3 that we are not able to use the technique in Section 4.3 to obtain the optimal value of (4.29) over ρ. However, we are able to use a specific value of ρ to obtain an improved approximation ratio, and we later show that it is asymptotically very close to the optimal choice. Thus in this case we fix the value of ρ as follows:

$$\hat{\rho}^* = 0.43. \tag{4.31}$$

By substituting it to (4.30) we set

$$\hat{\mu}^* = \frac{93m - \sqrt{4349m^2 - 4300m}}{100}. \tag{4.32}$$

We need to examine if $\hat{\rho}^*$ and $\hat{\mu}^*$ in (4.31) and (4.32) satisfy the assumption that $\hat{\rho}^* \geq \hat{\mu}^*/m$. Since $m \geq 3 > 4300/1849$, $2500 < 4349 - 4300/m$. Because both

sides are positive for $m \geq 3$, taking the square root we have $50 < \sqrt{4349 - 4300/m}$. Therefore $\hat{\rho}^* = 43/100 > (93 - \sqrt{4349 - 4300/m})/100 = \hat{\mu}^*/m$.

Lemma 4.16 *In the case that $\rho > \mu/m$ and $m \geq 3$, Algorithm II has an approximation ratio r is bounded by*

$$\frac{100}{43} + \frac{100}{43}\frac{(43m - 100)(57\sqrt{4349m^2 - 4300m} - 399m - 4300)}{139707m^2 - 174021m - 184900}.$$

Proof: It worth noting that the $\hat{\mu}^*$ in (4.32) can be a fractional number. Therefore we need to consider $\lceil \hat{\mu}^* \rceil$ and $\lfloor \hat{\mu}^* \rfloor$. Since we should minimize the objective function over μ, the approximation ratio with integer value of μ is bounded as follows:

$$r \leq \min\{\max\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*)\}, \max\{A(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\}\}.$$

According to the analysis in Subsubsection 4.4.1.2, here $A(\mu, \rho)$ is increasing in μ and $B(\mu, \rho)$ is decreasing in μ for a fixed ρ as we are in **CASE 1**. Thus the bound on approximation ratio is

$$r \leq \min\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\}$$

Furthermore, $\lceil \hat{\mu}^* \rceil \geq \hat{\mu}^* - 1$ and $\lfloor \hat{\mu}^* \rfloor \leq \hat{\mu}^* + 1$. Again, because $A(\mu, \rho)$ is increasing and $B(\mu, \rho)$ is decreasing, we have

$$\begin{aligned} A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*) &\leq A(\hat{\mu}^* + 1, \hat{\rho}^*); \\ B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*) &\leq B(\hat{\mu}^* - 1, \hat{\rho}^*). \end{aligned}$$

Thus we have the following bound on the ratio r:

$$\begin{aligned} r &\leq \min\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\} \\ &\leq \min\{A(\hat{\mu}^* + 1, \hat{\rho}^*), B(\hat{\mu}^* - 1, \hat{\rho}^*)\} \\ &\leq A(\hat{\mu}^* + 1, \hat{\rho}^*). \end{aligned}$$

Therefore here we shall find an upper bound on $A(\hat{\mu}^* + 1, \hat{\rho}^*)$, which is also an upper bound on the approximation ratio r. Substituting $\hat{\mu}^*$ in (4.32) and $\hat{\rho}^*$ in (4.31) in $A(\hat{\mu}^* + 1, \hat{\rho}^*)$ gives:

$$
\begin{aligned}
r &\leq A(\hat{\mu}^* + 1, \hat{\rho}^*) = \frac{1}{\hat{\rho}^*} + \frac{1}{\hat{\rho}^*} \frac{\hat{\rho}^* m - 1}{(1 - \hat{\rho}^*)(m - \hat{\mu}^* - 1) + 1} \\
&= \frac{100}{43} + \frac{100}{43} \frac{\frac{43m}{100} - 1}{\left(1 - \frac{43}{100}\right)\left(m - \frac{93m - \sqrt{4349m^2 - 4300m}}{100} - 1\right) + 1} \\
&= \frac{100}{43} + \frac{100}{43} \frac{(43m - 100)(57\sqrt{4349m^2 - 4300m} - 399m - 4300)}{139707m^2 - 174021m - 184900}.
\end{aligned}
$$

This is the claimed bound in the theorem. □

Combine Lemma 4.14 and Lemma 4.16 we have the following theorem of the approximation ratio of Algorithm II:

Theorem 4.2 *There exists an algorithm for the problem of* Scheduling Malleable Tasks with Precedence Constraints *with an approximation ratio*

$$
r \leq \begin{cases} 2, & \text{if } m = 2; \\ \dfrac{100}{43} + \dfrac{100}{43} \dfrac{(43m - 100)(57\sqrt{4349m^2 - 4300m} - 399m - 4300)}{139707m^2 - 174021m - 184900} & \text{otherwise.} \end{cases}
$$

Proof: We need to compare the minimum objective values in both cases $\rho \leq \mu/m$ and $\rho > \mu/m$. Thus we need to compare the ratio m in Lemma 4.14 and the value in Lemma 4.16. Suppose that

$$
m \geq \frac{100}{43} + \frac{100}{43} \frac{(43m - 100)(57\sqrt{4349m^2 - 4300m} - 399m - 4300)}{139707m^2 - 174021m - 184900}
$$

By moving the right hand side to the left hand side and simplification, we obtain

$$
\frac{(43m - 100)(139707m^2 - 134121m + 245100 - 5700\sqrt{4349m^2 - 4300m})}{139707m^2 - 174021m - 184900} \geq 0. \tag{4.33}
$$

Denote by NUM the numerator, and by DEN the denominator, of the left hand side of the above inequality, respectively. Solving equation $NUM = 0$ (which is equivalent to $43m - 100 = 0$ or $139707m^2 - 134121m + 245100 - 5700\sqrt{4349m^2 - 4300m} = 0$) we obtain the following five roots:

$$
\begin{aligned}
m_1 &= \frac{100}{43} \approx 2.32558; \\
m_2 &= \frac{1}{114}(71 + \sqrt{22241}) \approx 1.93100; \\
m_3 &= \frac{1}{86}(29 + \sqrt{18041}) \approx 1.89903; \\
m_4 &= \frac{1}{114}(71 - \sqrt{22241}) \approx -0.685387; \\
m_5 &= \frac{1}{86}(29 - \sqrt{18041}) \approx -1.22461.
\end{aligned}
$$

Solving equation $DEN = 0$ the roots are exactly m_2 and m_4 above. Thus both DEN and NUM are positive in the interval (m_1, ∞) according to Proposition 4.3. If $m \geq 3$, then the inequality holds. In this case we compute $\hat{\rho}^*$ and $\hat{\mu}^*$ by (4.31) and (4.32) (with rounding to integer) to obtain the approximation ratio bounded in Lemma 4.16. If $m = 2$, then the case $\rho > \mu/m$ is not valid if we set $\hat{\rho}^* = 0.43$. In this case we should use the strategy for the case $\rho \leq \mu/m$ and we set $\rho^* = 1/m = 1/2$ and $\mu^* = 1$ to obtain an approximation ratio 2. The theorem is proved. □

Similarly, we have the following corollary for the upper bound of approximation ratio:

Corollary 4.3 *For all $m \in \mathbb{N}$ and $m \geq 2$, the approximation ratio*

$$r \leq \frac{100}{43} + \frac{100(\sqrt{4349} - 7)}{2451}.$$

Furthermore, when $m \to \infty$, the upper bound in Theorem 4.2 tends to

$$\frac{100}{43} + \frac{100(\sqrt{4349} - 7)}{2451} \approx 4.730598.$$

Proof: It is obvious that when $m = 2$, the approximation ratio $r = 2$ fulfils the inequality. We now consider the case that $m \geq 3$.

Now we need to prove that

$$\frac{100}{43}+\frac{100}{43}\frac{(43m-100)(57\sqrt{4349m^2-4300m}-399m-4300)}{139707m^2-174021m-184900}$$
$$\leq \frac{100}{43}+\frac{100(\sqrt{4349}-7)}{2451},$$

which is equivalent to

$$\frac{(43m-100)(57\sqrt{4349m^2-4300m}-399m-4300)}{139707m^2-174021m-184900}\leq\frac{\sqrt{4349}-7}{57}.$$

Moving all terms on the left hand side to the right hand side and simplification gives

$$\frac{1}{57(139707m^2-174021m-184900)}[(\sqrt{4349}-7)(139707m^2-174021m-184900)$$
$$-57(43m-100)(57\sqrt{4349m^2-4300m}-399m-4300)]\geq 0.$$

Denote by DEN the denominator and by NUM the numerator of the left hand side of above inequality. The equation $DEN=0$ has two roots m_2 and m_4 as in Theorem 4.2. This shows that according to Proposition 4.3, if $m\geq 2$, then the denominator $DEN>0$. So if $NUM\geq 0$, the bound can be proved. The inequality $NUM\geq 0$ is equivalent to

$$\begin{aligned}139707\sqrt{4349}m^2 \;+\; & (9483147-174021\sqrt{4349})m-(23215700+184900\sqrt{4349})\\ \geq\; & 3249(43m-100)\sqrt{4349m^2-4300m}. \qquad (4.34)\end{aligned}$$

Denote by LHS the left hand side of above inequality, the equation $LHS=0$ has the following two roots by Proposition 4.2:

$$\begin{aligned}m_6 &= \frac{1}{4902}\left(3053-\frac{166371}{\sqrt{4349}}-\sqrt{\frac{206525885182+2977239074\sqrt{4349}}{4349}}\right)\\ &\approx -1.85526;\\ m_7 &= \frac{1}{4902}\left(3053-\frac{166371}{\sqrt{4349}}+\sqrt{\frac{206525885182+2977239074\sqrt{4349}}{4349}}\right)\\ &\approx 2.07157.\end{aligned}$$

Thus if $m \geq 3$, then $LHS \geq 0$ according to Proposition 4.3. So we can take the square of both side of the inequality (4.34). After simplification the inequality becomes:

$$\begin{aligned}
&(267271455786894 + 2649724035858\sqrt{4349})m^3 \\
&-(852494422797882 + 9787325047974\sqrt{4349})m^2 \\
&+(293463299648400 + 4573170898800\sqrt{4349})m \\
&+(687652381980000 + 8585165860000\sqrt{4349}) \geq 0.
\end{aligned}$$

It is easy to verify that when

$$\begin{aligned}
m &\geq (852494422797882 + 9787325047974\sqrt{4349}) \\
&\quad /(267271455786894 + 2649724035858\sqrt{4349}) \\
&\approx 3.3889,
\end{aligned}$$

it holds that

$$\begin{aligned}
&(267271455786894 + 2649724035858\sqrt{4349})m \\
&-(852494422797882 + 9787325047974\sqrt{4349}) \geq 0,
\end{aligned}$$

and the inequality (4.34) holds. Thus we just need to check the case when $m = 3$. Substitute $m = 3$ to the left hand side of the inequality (4.34) we have

$$LHS = 1100800(1010103363 + 5233741\sqrt{4349}) > 0.$$

Thus for all $m \geq 3$ the inequality (4.34) holds and the bound claimed in the corollary is proved.

When $m \to \infty$, the upper bound in Theorem 4.2 is

$$\begin{aligned}
&\frac{100}{43} + \frac{100}{43} \cdot \frac{(43m - 100)(57\sqrt{4349m^2 - 4300m} - 399m - 10000)}{139707m^2 - 174021m - 184900} \\
= \quad &\frac{100}{43} + \frac{100}{43} \cdot \frac{(43 - 100/m)(57\sqrt{4349 - 4300/m} - 399 - 10000/m)}{139707 - 174021/m - 184900/m^2}
\end{aligned}$$

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.500	2.0000	18	6	0.430	4.3249
3	2	0.430	2.7551	19	6	0.430	4.3083
4	2	0.430	3.1080	20	6	0.430	4.2938
5	2	0.430	3.3125	21	6	0.430	4.2880
6	2	0.430	3.4458	22	7	0.430	4.3857
7	3	0.430	3.7507	23	7	0.430	4.3685
8	3	0.430	3.7995	24	7	0.430	4.3531
9	3	0.430	3.8356	25	7	0.430	4.3897
10	3	0.430	3.9078	26	8	0.430	4.4281
11	4	0.430	4.0639	27	8	0.430	4.4113
12	4	0.430	4.0656	28	8	0.430	4.3961
13	4	0.430	4.0669	29	8	0.430	4.4663
14	4	0.430	4.1739	30	9	0.430	4.4593
15	5	0.430	4.2173	31	9	0.430	4.4433
16	5	0.430	4.2065	32	9	0.430	4.4287
17	5	0.430	4.1973	33	10	0.430	4.4995

Table 4.4: Bounds on approximation ratios for Algorithm II.

$$\begin{aligned} &\to \quad \frac{100}{43} + \frac{100}{43} \cdot \frac{43(57\sqrt{4349} - 399)}{139707} \\ &= \quad \frac{100}{43} + \frac{100(\sqrt{4349} - 7)}{2451} \approx 4.730598. \end{aligned}$$

This completes the proof. □

We here give the list of values of approximation ratios for our Algorithm II for $m = 2, \ldots, 33$ in Table 4.4. The method to compute the values is similar to that for the Algorithm I, while here the value of ρ is fixed (See Appendix for codes).

4.4.3 Asymptotic behaviour of approximation ratio

In Algorithm II we set $\hat{\rho}^* = 0.43$. However, the approximation ratio r can be improved by choosing the value of ρ^* depending on m, like that in Algorithm I. In this subsection we are going to study it.

Since μ^* in (4.30) is the minimizer of the objective function in (4.29). By substituting μ^* to $A(\mu,\rho)$ or $B(\mu,\rho)$ we can obtain two functions $A(\rho)$ or $B(\rho)$. Since our goal is to find the minimum value of $A(\mu,\rho)$ or $B(\mu,\rho)$ over all ρ, we need to solve the equation $A'(\rho)_\rho = 0$ or $B'(\rho)_\rho = 0$. Because $A(\rho) = B(\rho)$, we just need to consider one of them, say, $A(\rho)$. The first order partial derivative of $A(\rho)$ with respect to ρ is

$$\begin{aligned} A'(\rho)_\rho &= -\frac{1}{\rho^2} + \frac{1}{\rho^2[(1-\rho)(m-\mu^*)+1]^2}\{m\rho[(1-\rho)(m-\mu^*)+1] \\ &\quad -(\rho m-1)\{[(1-\rho)(m-\mu^*)+1]+\rho[-(m-\mu^*)-(1-\rho)(\mu^*)'_\rho]\}\Big\} \\ &= -\frac{1}{\rho^2} + \frac{(1-\rho)(m-\mu^*)+1+\rho(\rho m-1)[(m-\mu^*)+(1-\rho)(\mu^*)'_\rho]}{\rho^2[(1-\rho)(m-\mu^*)+1]^2}. \end{aligned}$$

Combine the two terms together and the denominator is positive. So the equation $A'(\rho)_\rho = 0$ can be simplified as follows:

$$-(1-\rho)^2(\mu^*)^2+(2m-4\rho m+\rho^2 m+1))\mu^*+\rho(1-\rho)(\rho m-1)(\mu^*)'_\rho+(-1+2\rho)m^2-m=0.$$

Here

$$\begin{aligned} \mu^* &= \frac{m(1+2\rho)}{2} - \frac{\sqrt{\Delta}}{2}; \\ (\mu^*)^2 &= \left(2\rho^2+\rho+\frac{1}{2}\right)m^2 - \rho m - \frac{(1+2\rho)m\sqrt{\Delta}}{2}; \\ (\mu^*)'_\rho &= m - \frac{m(2m\rho-1)}{\sqrt{\Delta}}, \end{aligned}$$

and $\Delta = (1+4\rho^2)m^2 - 4\rho m$. Substitute them to the equation and we obtain the following equation:

$$A_1\Delta + A_2\sqrt{\Delta} + A_3 = 0,$$

where the coefficients are as follows:

$$
\begin{aligned}
A_1 &= m\rho^3 - 2m\rho^2 + 2m\rho - \frac{m}{2} - \frac{1}{2};\\
A_2 &= -2m^2\rho^4 + (3m^2+m)\rho^3 + (-3m^2-m)\rho^2 + (2m^2+m)\rho - \frac{m^2}{2} - \frac{m}{2};\\
A_3 &= 2m^3\rho^4 + (-2m^3-3m^2)\rho^3 + (3m^2+m)\rho^2 - m\rho.
\end{aligned}
$$

To remove the square root, we can simplify the equation to an equivalent equation

$$(A_1\Delta + A_3)^2 - A_2^2\Delta = 0.$$

After simplification, it can be written as the following form:

$$f(\rho) = m^2(m-1)\rho^2 \sum_{i=0}^{6} c_i\rho^i = 0, \tag{4.35}$$

where the coefficients are as follows:

$$
\begin{aligned}
c_0 &= -1 + m^2;\\
c_1 &= -2(1+m^2);\\
c_2 &= -1 + 15m + 3m^2 - m^3;\\
c_3 &= 2m(-7 - 8m + m^2);\\
c_4 &= 16m(1+m);\\
c_5 &= -4m(1+5m);\\
c_6 &= 4m^2(1+m).
\end{aligned}
$$

Besides the two trivial double roots $\rho = 0$ to (4.35), we obtain an equation with a polynomial of the highest order of 6. Unfortunately in general there are no analytic roots for polynomial with order higher than 4. So we are not able to solve (4.35) to obtain the optimal ρ^* depending on m like in Section 4.3.

In fact we can estimate the asymptotic behaviour of the approximation ratio. When $m \to \infty$, equation (4.35) is:

$$0 = 1 - m^2 + 2(1+m^2)\rho + (1 - 15m - 3m^2 + m^3)\rho^2 + 2m(7 + 8m - m^2)\rho^3$$

$$
\begin{aligned}
& -16m(1+m)\rho^4 + 4m(1+5m)\rho^5 - 4m^2(1+m)\rho^6 \\
= \; & \left[\frac{1}{m^3} - \frac{1}{m} + 2\left(\frac{1}{m^3} + \frac{1}{m}\right)\rho + \left(\frac{1}{m^3} - \frac{15}{m^2} - \frac{3}{m} + 1\right)\rho^2 + 2\left(\frac{7}{m^2} + \frac{8}{m} - 1\right)\rho^3\right. \\
& \left. -16\left(\frac{1}{m^2} + \frac{1}{m}\right)\rho^4 + 4\left(\frac{1}{m^2} + \frac{5}{m}\right)\rho^5 - 4\left(\frac{1}{m} + 1\right)\rho^6\right] m^3 \\
\rightarrow \; & m^3(-4\rho^4 - 2\rho + 1)\rho^2.
\end{aligned}
$$

Thus we just need to consider the equation $-4\rho^4 - 2\rho + 1 = 0$. Solving it we have the following roots:

$$
\begin{aligned}
\rho_1 &= \frac{1}{2}\sqrt{\frac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}} - \frac{2}{(3(9+\sqrt{273}))^{1/3}}} \\
&\quad -\frac{1}{2}\left[\frac{-(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}} + \frac{2}{(3(9+\sqrt{273}))^{1/3}}\right. \\
&\quad \left. -\frac{1}{\sqrt{\frac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}} - \frac{2}{(3(9+\sqrt{273}))^{1/3}}}}\right]^{1/2} \\
&\approx -0.917543; \\
\rho_2 &= \frac{1}{2}\sqrt{\frac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}} - \frac{2}{(3(9+\sqrt{273}))^{1/3}}} \\
&\quad +\frac{1}{2}\left[\frac{-(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}} + \frac{2}{(3(9+\sqrt{273}))^{1/3}}\right. \\
&\quad \left. -\frac{1}{\sqrt{\frac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}} - \frac{2}{(3(9+\sqrt{273}))^{1/3}}}}\right]^{1/2} \\
&\approx -0.243276 - 0.75697i; \\
\rho_3 &= -\frac{1}{2}\sqrt{\frac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}} - \frac{2}{(3(9+\sqrt{273}))^{1/3}}}
\end{aligned}
$$

$$
\begin{aligned}
& -\frac{1}{2}\left[\frac{-(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}}+\frac{2}{(3(9+\sqrt{273}))^{1/3}}\right. \\
& \left.-\frac{1}{\sqrt{\dfrac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}}-\dfrac{2}{(3(9+\sqrt{273}))^{1/3}}}}\right]^{1/2} \\
\approx\ & -0.243276+0.75697i; \\
\rho_4 =\ & -\frac{1}{2}\sqrt{\frac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}}-\frac{2}{(3(9+\sqrt{273}))^{1/3}}} \\
& +\frac{1}{2}\left[\frac{-(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}}+\frac{2}{(3(9+\sqrt{273}))^{1/3}}\right. \\
& \left.-\frac{1}{\sqrt{\dfrac{(9+\sqrt{273})^{1/3}}{2\cdot 3^{2/3}}-\dfrac{2}{(3(9+\sqrt{273}))^{1/3}}}}\right]^{1/2} \\
\approx\ & 0.430991.
\end{aligned}
$$

The only feasible root here in the interval $\rho \in (0,1)$ is $\rho^* = 0.430991$. Substituting it to (4.30) the optimal $\mu^* \to 0.270875m$. With these data, from either $A(\mu,\rho)$ or $B(\mu,\rho)$ one have that

$$r \to 4.730577.$$

In Algorithm II we fix $\hat{\rho}^* = 0.43$ just because it is close to the asymptotic optimum ρ^*. The ratio of Algorithm II could be further improved by fix $\hat{\rho}^*$ to a better approximation to ρ^*. In this way we conjecture that there exists a 4.730577-approximation algorithm for the problem of Scheduling Malleable Tasks with Precedence Constraints. However, the analysis is complicated and Algorithm II has already a ratio 4.730598 very close to this asymptotic ratio.

We can also use numerical method to solve the min-max nonlinear program (4.29). We can construct a grid of ρ in the interval $(0,1)$, and μ in $[1, \lfloor (m+1)/2 \rfloor]$. The grid size for ρ is $\delta\rho$ and for μ is 1 as μ is an integer. We can compute the values of $A(\mu,\rho)$ and $B(\mu,\rho)$ on each grid point, and search for the minimum over

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.500	2.0000	18	5	0.401	4.2579
3	2	0.618	2.6180	19	6	0.484	4.2630
4	2	0.581	2.9610	20	6	0.467	4.2520
5	2	0.562	3.1717	21	6	0.429	4.2837
6	2	0.445	3.4139	22	7	0.480	4.3466
7	3	0.523	3.6617	23	7	0.481	4.3276
8	3	0.519	3.7104	24	7	0.451	4.3257
9	3	0.500	3.7500	25	7	0.420	4.3581
10	3	0.420	3.8867	26	8	0.477	4.3918
11	4	0.500	4.0000	27	8	0.469	4.3747
12	4	0.500	4.0000	28	8	0.440	4.3822
13	4	0.462	4.0198	29	8	0.414	4.4139
14	4	0.408	4.1196	30	9	0.475	4.4250
15	5	0.490	4.1653	31	9	0.456	4.4142
16	5	0.491	4.1526	32	9	0.431	4.4268
17	5	0.441	4.1787	33	9	0.409	4.4571

Table 4.5: Numerical results of min-max nonlinear program (4.29).

all grid points to decide the optimal objective values depending on m (See Appendix for codes). The results by setting $\delta\rho = 0.0001$ and $m = 2, \ldots, 33$ are in Table 4.5. Compared the results in Table 4.4 we can see that the solutions of our Algorithm II are already very close to the optimum.

4.5 Approximation algorithm III for the new model

In this section, we propose an approximation algorithm for the problem of Scheduling Malleable Tasks with Precedence Constraints in our new model under Assumption 4.1 and 4.3. We will show some nice properties of the work function in this model. In our Algorithm III, we do not use the idea of the discrete time-cost tradeoff problem in the first phase for solving the allotment problem. Instead, we formulate the allotment problem as a piecewise integer linear program. Then we

solve the linear programming relaxation and apply a new parameterized rounding technique. Similar to Algorithm I and II, the rounding parameter is also introduced to the second phase to formulate a min-max nonlinear program and to obtain the optimal approximation ratio.

First we show that the work function is increasing in the number of processors:

Theorem 4.3 *For any malleable task J_j and m identical processors, if Assumption 4.3 for the processing time $p_j(l)$ of J_j holds for all $l = 0, \ldots, m$, then the work function $W_j(l) = lp_j(l)$ for task J_j is non-decreasing in l, i.e., $W_j(l') \geq W_j(l)$, for any integers $1 \leq l \leq l' \leq m$.*

Proof: We prove the theorem by induction. First, from the Assumption 4.3, we have that

$$\frac{1}{p_j(1)} \geq \frac{1}{2}\left[\frac{1}{p_j(2)} + \frac{1}{p_j(0)}\right].$$

Because $p_j(0) = \infty$, it holds that

$$\frac{1}{p_j(1)} \geq \frac{1}{2p_j(2)},$$

i.e., $2p_j(2) \geq p_j(1)$. Now we assume that it holds that $W_j(1) = p_j(1) \leq \ldots \leq W_j(l-1) = (l-1)p_j(l-1) \leq W_j(l) = lp_j(l)$. Again, from Assumption 4.3, we have

$$\frac{1}{p_j(l)} \geq \frac{1}{2}\left[\frac{1}{p_j(l+1)} + \frac{1}{p_j(l-1)}\right] \geq \frac{1}{2}\left[\frac{1}{p_j(l+1)} + \frac{l-1}{lp_j(l)}\right],$$

i.e.,

$$\frac{l+1}{lp_j(l)} \geq \frac{1}{p_j(l+1)},$$

which is equivalent to $lp_j(l) \leq (l+1)p_j(l+1)$. Then we conclude that for any $l = 1, \ldots, m-1$, $W_j(l) = lp_j(l) \leq (l+1)p_j(l+1) = W_j(l+1)$, which leads to a non-decreasing series $W_j(1), \ldots, W_j(m)$, and the proof is complete. □

Theorem 4.3 in fact is Assumption 4.2 on work functions. Now the Assumption 4.2 in the previous model is only a sequel of our Assumption 4.3. Furthermore, if

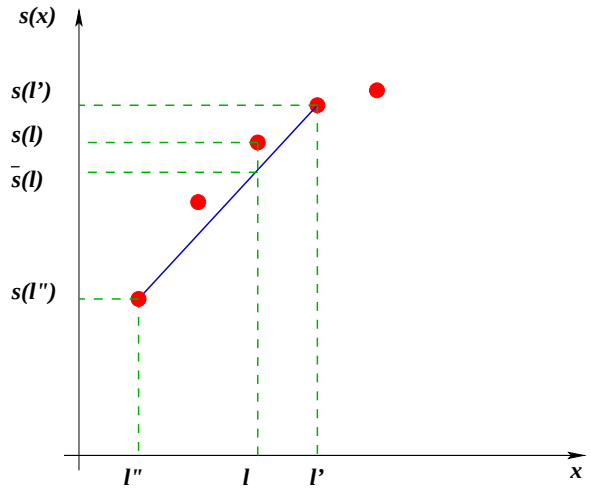


Figure 4.17: Speedup function $s_j(l)$.

we regard the work functions as functions in the corresponding processing time, i.e., $w_j(p_j(l)) = W_j(l)$, the following theorem shows that Assumption 4.1 and 4.3 leads to a nice property of the work functions:

Theorem 4.4 *If Assumption 4.1 and 4.3 hold for any malleable task J_j for any $l = 1, \ldots, m$, then the work function $w_j(p_j(l))$ is convex in the processing time $p_j(l)$.*

Proof: According to Assumption 4.3, the speedup function $s_j(l)$ is concave in the number l of processors. Therefore in the diagram of the speed function $s_j(l)$ versus l (See Figure 4.17), $s_j(l) \geq \bar{s}_j(l)$, where $\bar{s}_j(l)$ is the vertical coordinate of the intersection point of the straight line connecting points $(l'', s_j(l''))$ and $(l', s_j(l'))$ and the vertical straight line passing through point $(l, s_j(l))$, where $1 \leq l'' \leq l \leq l' \leq m$. Then we obtain the following inequality by (4.3) and by calculating the value of $\bar{s}_j(l)$:

$$\frac{p_j(1)}{p_j(l)} = s_j(l) \geq \bar{s}_j(l) = \frac{p_j(1)}{l' - l''} \left[\frac{l - l''}{p_j(l')} - \frac{l - l'}{p_j(l'')} \right]. \tag{4.36}$$

We now consider the diagram of the work function $w_j(p_j(l))$ versus processing time $p_j(l)$ (Figure 4.18). The straight line connecting points $(p_j(l''), w_j(p_j(l'')))$ and $(p_j(l'), w_j(p_j(l')))$ and the vertical straight line passing through point $(p_j(l), w_j(p_j(l)))$ intersect at one point which has the vertical coordinate $\bar{w}_j(p_j(l))$ as follows:

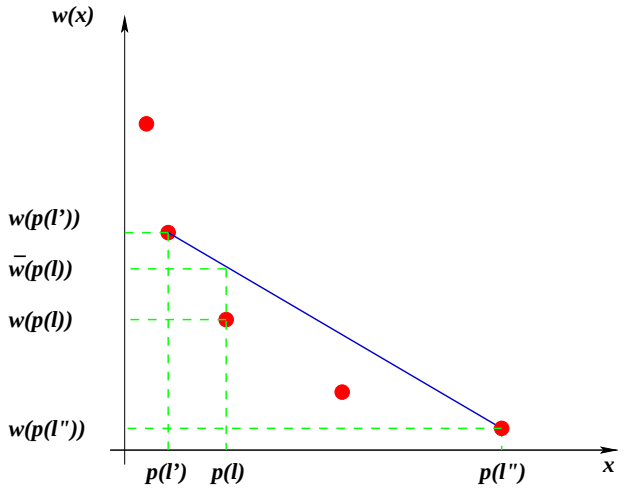


Figure 4.18: Work function $w_j(p_j(l))$.

$$\begin{aligned}
\bar{w}_j(p_j(l)) &= w_j(p_j(l'')) + \frac{p_j(l) - p_j(l'')}{p_j(l') - p_j(l'')}[w_j(p_j(l')) - w_j(p_j(l''))] \\
&= l''p_j(l'') + \frac{p_j(l) - p_j(l'')}{p_j(l') - p_j(l'')}[l'p_j(l') - l''p_j(l'')] \\
&= \frac{1}{p_j(l') - p_j(l'')}\{p_j(l)[l'p_j(l') - l''p_j(l'')] - (l' - l'')p_j(l')p_j(l'')\}.
\end{aligned} \tag{4.37}$$

From (4.36) we have that

$$\frac{l}{p_j(l'')} - \frac{l}{p_j(l')} \geq \frac{l'}{p_j(l'')} - \frac{l''}{p_j(l')} - \frac{l' - l''}{p_j(l)}.$$

Multiplying both sides by the positive factor $p_j(l)p_j(l')p_j(l'')$ yields

$$lp_j(l)[p_j(l') - p_j(l'')] \geq p_j(l)[l'p_j(l') - l''p_j(l'')] - (l' - l'')p_j(l')p_j(l'').$$

By multiplying both sides by the negative factor $1/[p_j(l') - p_j(l'')]$ due to Assumption 4.1, together with (4.37), we immediately obtain that $w_j(p_j(l)) = lp_j(l) \leq \bar{w}_j(p_j(l))$, which show that the work function $W_j(l) = w_j(p_j(l))$ is convex in processing time $p_j(l)$. □

Denote by x_j the (fractional) processing time of task J_j. For the discrete work function $w_j(p_j(l)) = W_j(l) = lp_j(l)$ in processing time, we define a continuous piecewise linear work function $w_j(x_j)$ as follows:

$$w_j(x_j) = \begin{cases} w_j(p_j(l)), & \text{if } x = p_j(l), l = 1, \ldots, m; \\ \dfrac{w_j(p_j(l+1)) - w_j(p_j(l))}{p_j(l+1) - p_j(l)} x_j & \text{if } x \in (p_j(l+1), p_j(l)), \\ + \dfrac{w_j(p_j(l))p_j(l+1) - w_j(p_j(l+1))p_j(l)}{p_j(l+1) - p_j(l)}, & \text{and } l = 1, \ldots, m-1. \end{cases} \tag{4.38}$$

In addition, in any schedule, we know that the makespan (maximum completion time) is an upper bound of the critical path length L and the total work W divided by m, i.e., $\max\{L, W/m\} \leq C_{\max}$. In the first phase of our algorithm, we solve the following piecewise linear program:

$$\begin{array}{lll} \min & C & \\ \text{s.t.} & C_i + x_j \leq C_j, & \text{for all } i \in \Gamma^-(j) \text{ and all } j; \\ & C_j \leq L, & \text{for all } j; \\ & L \leq C; & \\ & W/m = \sum_{j=1}^{n} w_j(x_j)/m \leq C; & \\ & x_j \in [p_j(m), p_j(1)], & \text{for all } j. \end{array} \tag{4.39}$$

In (4.39) the first set of constraints come from the precedence constraints, while the second set of constraints indicate that all tasks finish by the critical path length L. The goal is to minimize the makespan C.

We notice that the functions $w_j(x_j)$ are piecewise linear and it is a crucial issue whether (refplp) is polynomial time solvable. According to Theorem 4.4, the discrete work function $w_j(p_j(l))$ is convex in processing time $p_j(l)$. Therefore the continuous work function $w_j(x_j)$ is also convex in the fractional processing time x_j. Since $w_j(x_j)$ is piecewise linear, it can be written as follows:

$$\begin{aligned} w_j(x_j) &= \max_{l \in \{1,\ldots,m-1\}} \left\{ \frac{w_j(p_j(l+1)) - w_j(p_j(l))}{p_j(l+1) - p_j(l)} x_j \right. \\ &\quad \left. + \frac{w_j(p_j(l))p_j(l+1) - w_j(p_j(l+1))p_j(l)}{p_j(l+1) - p_j(l)} \right\} \\ &= \max_{l \in \{1,\ldots,m-1\}} \left\{ \frac{(l+1)p_j(l+1) - lp_j(l)}{p_j(l+1) - p_j(l)} x_j - \frac{p_j(l)p_j(l+1)}{p_j(l+1) - p_j(l)} \right\}, \end{aligned} \tag{4.40}$$

for any $x_j \in [p_j(m), p_j(1)]$. Thus we are able to modify the piecewise linear program (4.39) to following linear program:

$$\begin{array}{lll}
\min & C & \\
\text{s.t.} & C_i + x_j \le C_j, & \text{for all } i \in \Gamma^-(j) \text{ and all } j; \\
& C_j \le L, & \text{for all } j; \\
& \dfrac{(l+1)p_j(l+1) - lp_j(l)}{p_j(l+1) - p_j(l)} x_j - \dfrac{p_j(l)p_j(l+1)}{p_j(l+1) - p_j(l)} \le w_j, & \text{for } l = 1, \ldots, m-1 \text{ and all } j; \\
& L \le C; & \\
& W/m = \sum_{j=1}^{n} w_j/m \le C; & \\
& x_j \in [p_j(m), p_j(1)], & \text{for all } j.
\end{array} \tag{4.41}$$

The size of (4.41) is polynomial in m and n. Thus it is polynomial time solvable.

An example $\mathcal{E}$ of this model is that the processing time is a function $p(l) = l^{-d}$ for $0 < d < 1$. In this case the fractional work function is defined as follows: For all $l = \{1, \ldots, m\}$,

$$w(x) = \begin{cases} lp(l) = l^{-d+1} = p(l)^{1-1/d} = x^{1-1/d}, & \text{when } x = p(l); \\ \dfrac{(l+1)^{1-d} - l^{1-d}}{(l+1)^{-d} - l^{-d}} x & \\ \quad + \dfrac{l^{1-d}(l+1)^{-d} - (l+1)^{1-d} l^{-d}}{(l+1)^{-d} - l^{-d}}, & \text{when } x \in (p_j(l+1), p_j(l)). \end{cases} \tag{4.42}$$

When $d = 1/2$, the example is illustrated in Figure 4.19 and 4.20.

For task J_j, denote by x_j^* the corresponding optimal solution to (4.41). Suppose that $x_j^* \in (p_j(l+1), p_j(l))$. In the interval $[p_j(l+1), p_j(l)]$, we define a critical point l_c such that $l_c = l + 1 - \rho$ for $\rho \in [0, 1]$. The processing time $p_j(l_c)$ is given by

$$p_j(l_c) = p_j(l + 1 - \rho) = \rho p_j(l) + (1 - \rho) p_j(l + 1)$$

and its work is

$$w_j(p_j(l_c)) = (1 - \rho) w_j(p_j(l + 1)) + \rho w_j(p_j(l)) = (1 - \rho)(l + 1) p_j(l + 1) + \rho l p_j(l).$$

In our Algorithm III we take the following rounding technique for the fractional solution to the linear programming relaxation of (4.41): If $x_j^* \ge p_j(l_c)$ it will be

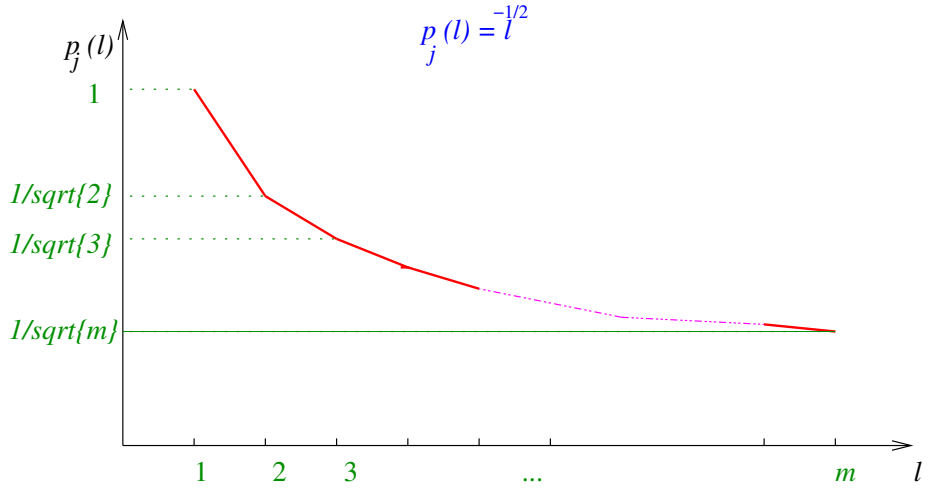


Figure 4.19: An example of our new model (processing time versus number of processors).

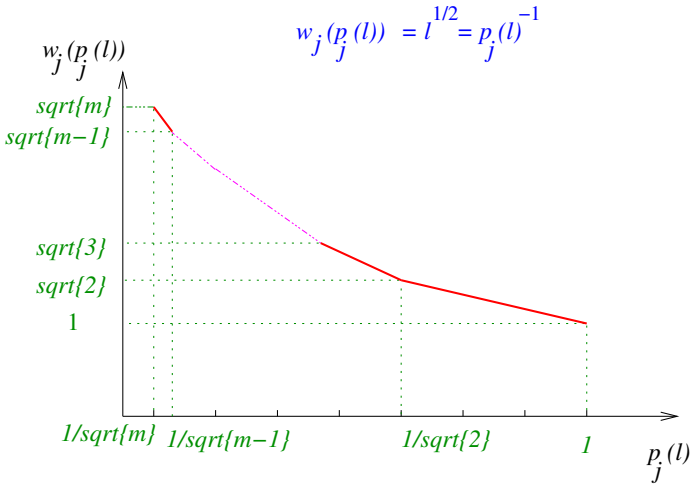


Figure 4.20: An example of our new model (work versus processoring time).

rounded up to $p_j(l)$ (i.e., $\rho = 1$), and otherwise down to $p_j(l+1)$ (i.e., $\rho = 0$). Then we develop an allotment α' for all jobs.

In the second phase of Algorithm III, we generate a new allotment α for all jobs based on the value of μ, and then apply the variant of list scheduling algorithm in Section 4.2 to obtain the final schedule.

Same as before, we denote by L, W, $C_{\max}$ and L', W', $C'_{\max}$ the critical path lengths, the total works and the makespans of the final schedule delivered by our algorithm and the schedule corresponding to the allotment α' generated in the first phase, respectively. Furthermore, we denote by $C^*_{\max}$ the optimal objective value of

(4.41), and L^*, W^* the (fractional) optimal critical path length and the (fractional) optimal total work in (4.41). Denote by OPT the overall optimal makespan (over all feasible schedules with integral number of processors allotted to all tasks). (4.12) still holds in this case. In allotments α and α', a task J_j is allotted l_j and l'_j processors, and their processing times are $p_j(l_j)$ and $p_j(l'_j)$, respectively. In the optimal (fractional) solution to (4.41), each task J_j has a fractional processing time x^*_j. We also define the fractional number of processors allotted as follows:

$$l^*_j = w_j(x^*_j)/x^*_j, \tag{4.43}$$

and Lemma 4.10 hold here.

In the first phase, for each job the increases of processing time and work after rounding are bounded by the following lemma:

Lemma 4.17 *For any job J_j, in the allotment α' its processing time*

$$p_j(l'_j) \leq \frac{2}{1+\rho} x^*_j,$$

and the its work

$$W_j(l'_j) = w_j(p_j(l'_j)) = l'_j p_j(l'_j) \leq \frac{2}{1+\rho} l^*_j x^*_j = \frac{2}{2-\rho} w_j(x^*_j),$$

where x^*_j is the optimal solution to the linear programming relaxation of (4.41). l^*_j and $w_j(x^*_j)$ are number of processors and work of J_j corresponding to the solution x^*_j.

Proof: Suppose $x^*_j \in (p_j(k+1), p_j(k))$. In the domain $[p_j(k+1), p_j(k)]$, the critical point $k_c = k + 1 - \rho$. Its processing time is $p_j(k_c) = p_j(k+1-\rho) = \rho p_j(k) + (1-\rho)p_j(k+1)$ and its work is $w_j(p_j(k_c)) = (k+1-\rho)p_j(k+1-\rho) = (k+1-\rho)(\rho p_j(k) + (1-\rho)p_j(k+1))$. We consider the following two cases.

In the first case, $x^*_j \geq p_j(k_c)$. In the rounding the processing time is rounded up to $p_j(k)$, and the number of processors is reduced to $l'_j = k$. Therefore the work is also reduced due to Assumption 4.1. However, the increase of the processing time is

$$\begin{aligned}\frac{p_j(l'_j)}{x^*_j} &\leq \frac{p_j(k)}{p_j(k_c)} \\ &= \frac{p_j(k)}{\rho p_j(k) + (1-\rho)p_j(k+1)} \\ &\leq \frac{p_j(k)}{\rho p_j(k) + (1-\rho)kp_k(k)/(k+1)} \\ &= \frac{k+1}{k+\rho}.\end{aligned}$$

The second inequality holds also from Theorem 4.3, $kp_j(k) \leq (k+1)p_j(k+1)$. In the second case, $x^*_j < p_j(k_c)$. In the rounding the processing time is rounded down to $p_j(k+1)$), and the number of processors is increased to $l'_j = k+1$. Since more processors are allotted, according to Theorem 4.3 the work increases by the following factor:

$$\begin{aligned}\frac{w_j(p_j(l'_j))}{w_j(x^*_j)} &\leq \frac{(k+1)p_j(k+1)}{w_j(p_j(k_c))} \\ &= \frac{(k+1)p_j(k+1)}{(1-\rho)(k+1)p_j(k+1) + \rho k p_j(k))} \\ &\leq \frac{(k+1)p_j(k+1)}{(1-\rho)(k+1)p_j(k+1) + \rho k p_j(k+1))} \\ &= \frac{k+1}{k+1-\rho}.\end{aligned}$$

Since k is an integer, the above two factors can be further bounded by $(k+1)/(k+1-\rho) \leq 2/(2-\rho)$ and $(k+1)/(1+\rho) \leq 2/(1+\rho)$. This means that after the first phase, for each task J_j, the processing time increases by at most a factor of $2/(1+\rho)$ and the work increases by at most $2/(2-\rho)$. □

In the final schedule, the time interval $[0, C_{\max}]$ also consists of three types of time slots T_1, T_2 and T_3, same as Section 4.3. Thus the following lemma holds:

Lemma 4.18 $\frac{1+\rho}{2}|T_1| + \min\left\{\frac{\mu}{m}, \frac{1+\rho}{2}\right\}|T_2| \leq C^*_{\max}$.

Proof: We also construct a "heavy" path $\mathcal{P}$ in the same way as in Lemma 4.2. Now we examine the stretch of processing time for all jobs in $\mathcal{P}$ in the rounding procedure of the first phase and in the new allotment α of the second phase.

For any job J_j in $T_1 \cap \mathcal{P}$, the processing time of the fractional solution to the linear program in the first phase increases by at most a factor $2/(1+\rho)$. The processing time does not change in the second phase as in α' the job J_j is only allotted a number $l'_j \leq \mu$ of processors such that it can be in the time slot of T_1. Therefore for such kind of jobs we have $p_j(l_j) = p_j(l'_j) \leq 2x^*_j/(1+\rho)$.

For any job J_j in $T_2 \cap \mathcal{P}$, there are two cases. In the first case, in α' a job J_j is allotted $l'_j \leq \mu$ processors. This is same as the case before and we also have $p_j(l_j) \leq 2x^*_j/(1+\rho)$. In the second case, in α' a job J_j is allotted $l'_j > \mu$ processors, and $l_j = \mu$. Then there are two subcases according to the solution to the linear program. In the first subcase, in the fractional solution to (4.41) there are $l^*_j \geq \mu$ processors allotted. Since μ is an integer, we have $l^*_j \geq \lfloor l^*_j \rfloor \geq \mu \geq l_j$. Then $l_j p_j(l_j) = W_j(l_j) \leq W_j(\mu) \leq W_j(\lfloor l^*_j \rfloor) \leq w_j(x^*_j) = l^*_j x^*_j \leq W_j(\lceil l^*_j \rceil)$ due to Theorem 4.3 and (4.22). Because $l^*_j \leq m$, and $w_j(x^*_j) = x^*_j l^*_j \geq p_j(l_j) l_j = W_j(l_j)$, it holds that $p_j(l_j) \leq x^*_j l^*_j / l_j \leq x^*_j m/\mu$. In the second subcase, in the fractional solution there are $l^*_j < \mu$ processors allotted to J_j. Then in the rounding procedure in the first phase the processing time must be rounded down from x^*_j to $p_j(l'_j)$ as only in this way the assumption that $l'_j > \mu$ of this case can be satisfied. Then in the second phase, J_j is allotted μ processors and from Theorem 4.3 $p_j(l_j) l_j \leq p_j(l'_j) l'_j$. Since there are at most m processors allotted to J_j in α', we have $p_j(l_j) \leq p_j(l'_j) l'_j / l_j \leq p_j(l'_j) m/\mu \leq x^*_j m/\mu$. Therefore for any job J_j in $T_2 \cap \mathcal{P}$, $p_j(l_j) \leq x^*_j \max\{2/(1+\rho), m/\mu\}$.

Same as the argument in Lemma 4.2, the path $\mathcal{P}$ covers all time slots in the final schedule. In addition, in the schedule resulted from the fractional solution to the linear program, the jobs processed in T_1 in the final schedule contribute a total length of at least $\rho|T_1|$ to $L^*(\mathcal{P})$. In addition, the tasks processed in T_2 contribute a total length of at least $|T_2| \min\{(1+\rho)/2, \mu/m\}$ to $L^*(\mathcal{P})$. Since $L^*(\mathcal{P})$ is not more than the length $C^*_{\max}$ of the critical path in α^*, we have obtained the claimed inequality. □

The makespan of the resulting schedule is bounded by the following lemma:

Lemma 4.19 $(m-\mu+1)C_{\max} \leq \dfrac{2mC^*_{\max}}{2-\rho} + (m-\mu)|T_1| + (m-2\mu+1)|T_2|$.

Proof: Since in the second phase, for any job J_j, the allotted number of processors l_j is not more than l'_j, the number of processors in the first phase. Therefore according to Theorem 4.3 the total work is non-increasing, i.e., $W' \geq W$. According to Lemma 4.17, in the rounding procedure of the first phase, the total work only increases by at most a factor of $2/(2-\rho)$ from the total work of the optimum solution of (4.41). In this case we have that $W' \leq 2W^*/(2-\rho)$. Furthermore, from (4.12), $W \leq 2W^*/(2-\rho) \leq 2mC^*_{\max}/(2-\rho)$. Substituting it to the bound on $C_{\max}$ in Lemma 4.3 we obtain the claimed inequality. □

We take the same notations of normalized total length of time slots $x_i = |T_i|/C^*_{\max}$, $i = 1, 2, 3$. Similar to Section 4.3 and Section 4.4, the following lemma holds:

Lemma 4.20 *The optimal approximation ratio of Algorithm III is bounded by the optimal objective value of the following min-max nonlinear program*

$$\begin{array}{rll} \min_{\mu,\rho} \max_{x_1,x_2} & \dfrac{\dfrac{2m}{2-\rho} + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} & \\ s.t. & \dfrac{1+\rho}{2}x_1 + \min\left\{\dfrac{\mu}{m}, \dfrac{1+\rho}{2}\right\}x_2 \leq 1; & \\ & x_1, x_2 \geq 0; & \\ & \rho \in [0,1]; & \\ & \mu \in \left\{1, \ldots, \left\lfloor \dfrac{m+1}{2} \right\rfloor\right\}. & \end{array} \tag{4.44}$$

Proof: Divide the inequalities in Lemma 4.18 and Lemma 4.19 by $C^*_{\max}$, together with the definitions of x_i and approximation ratio,

$$\begin{aligned} r &= \sup \frac{C_{\max}}{OPT} \leq \sup \frac{C_{\max}}{C^*_{\max}} \\ &= \max_{x_1,x_2} \frac{2m(2-\rho) + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1}. \end{aligned}$$

On the other hand, we can select appropriate μ and ρ to minimize the ratio r. Hence, by combining them together with the other constraints for the variables according to their definitions, the approximation ratio is bounded by the objective value of (4.44). □

In the following we will solve the min-max nonlinear program (4.44) to get the optimal approximation ratio of Algorithm III.

4.5.1 Analysis of the min-max nonlinear program (4.44)

In order to solve (4.44), we need to consider two cases that either $\rho \leq 2\mu/m - 1$ or $\rho > 2\mu/m - 1$ to simplify the first constraint.

4.5.1.1 Solve (4.44) for the case $\rho \leq 2\mu/m - 1$

In this case, to guarantee that $\rho > 0$ it is required that

$$m/2 \leq \mu \leq (m+1)/2. \tag{4.45}$$

In addition, as $(1+\rho)/2 \leq \mu/m \leq (m+1)/2m$, it holds that

$$\rho \leq 1/m. \tag{4.46}$$

Thus we need to solve the following min-max nonlinear program:

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \quad & \frac{\frac{2m}{2-\rho} + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} \\ \text{s.t.} \quad & \frac{1+\rho}{2}x_1 + \frac{1+\rho}{2}x_2 \leq 1; \\ & x_1, x_2 \geq 0; \\ & \rho \in [0, 1/m]; \\ & \mu \in \left\{ \left\lceil \frac{m}{2} \right\rceil, \left\lfloor \frac{m+1}{2} \right\rfloor \right\}. \end{aligned} \tag{4.47}$$

We observe that the constraints on x_1 and x_2 in (4.47) forms a triangle, and the extreme points are $E_1 : (x_1, x_2) = (2/(1+\rho), 0)$, $E_2 : (x_1, x_2) = (0, 2/(1+\rho))$, and $E_3 : (x_1, x_2) = (0, 0)$. Since (4.47) is linear in x_1 and x_2, for a fixed pair of ρ and μ, the maximum value of the objective function exists at one of the extreme points. It is clear that the objective function can not attain the maximum value at E_3. So we

just consider E_1 and E_2. Denote by $A(\mu,\rho)$ and $B(\mu,\rho)$ the objective values at the extreme points E_1 and E_2, respectively. Then we have:

$$\begin{aligned} A(\mu,\rho) &= \frac{\frac{2m}{2-\rho}+\frac{2(m-\mu)}{1+\rho}}{m-\mu+1}; \\ B(\mu,\rho) &= \frac{\frac{2m}{2-\rho}+\frac{2(m-2\mu+1)}{1+\rho}}{m-\mu+1}. \end{aligned}$$

Since $\mu \geq 1$, $m-\mu \geq m-2\mu+1$ and $A(\mu,\rho) \geq B(\mu,\rho)$. Thus we just need to consider the problem of minimizing $A(\mu,\rho)$ over all feasible ρ and μ, i.e.,

$$\begin{aligned} \min_{\mu,\rho} \quad & A(\mu,\rho) = \frac{2(3-(2-\rho)\mu)}{(1+\rho)(2-\rho)(m-\mu+1)} \\ \text{s.t.} \quad & \rho \in [0, 1/m]; \\ & \mu \in \left\{ \left\lceil \frac{m}{2} \right\rceil, \left\lfloor \frac{m+1}{2} \right\rfloor \right\}. \end{aligned}$$

The first order partial derivative of $A(\mu,\rho)$ with respect to μ is:

$$\begin{aligned} A'(\mu,\rho)_\mu &= \frac{2}{(1+\rho)(2-\rho)} \cdot \frac{-(2-\rho)(m-\mu+1)+m(1+\rho)+(2-\rho)(m-\mu)}{(m-\mu+1)^2} \\ &= \frac{2}{(1+\rho)(2-\rho)} \cdot \frac{m-2+(m+1)\rho}{(m-\mu+1)^2}. \end{aligned}$$

Because $m \geq 2$ and $\rho > 0$, the denominator is nonnegative. Therefore we have $A'(\mu,\rho)_\mu \geq 0$. Thus we can choose a minimum feasible μ to minimize $A(\mu,\rho)$. Since in this case $(1+\rho)/2 \leq \mu/m$, we can set $\mu^* = m(1+\rho)/2$ and substitute it to $A(\mu,\rho)$. In this case we obtain a function $A(\rho)$ depending on ρ

$$\begin{aligned} A(\rho) &= \frac{2}{(1+\rho)(2-\rho)} \cdot \frac{m(1+\rho)+(2-\rho)(1-\rho)m/2}{(1-\rho)m/2+1} \\ &= \frac{2m(\rho^2-\rho+4)}{(2-\rho)(1+\rho)(m+2-m\rho)}. \end{aligned}$$

Now the problem is to minimize the above $A(\rho)$ over all $\rho \in [0, 1/m]$. The first order partial derivative of $A(\rho)$ with respect to ρ is:

$$A'(\rho)_\rho = \frac{2m(-12 + 2m + (20m + 24)\rho - 15m\rho^2 + 2m\rho^3 - m\rho^4)}{(2-\rho)^2(1+\rho)^2(m+2-m\rho)^2}.$$

We shall examine if $A'(\rho)_\rho$ is positive or not. It is obvious that the denominator is positive since $\rho \leq 1/m$. Denote by $Q(\rho) = -12 + 2m + (20m + 24)\rho - 15m\rho^2 + 2m\rho^3 - m\rho^4$. We now examine $Q(\rho)$. The first order partial derivative of $Q(\rho)$ with respect to ρ is:

$$Q(\rho)'_\rho = 24 + 20m - 30m\rho + 6m\rho^2 - 4m\rho^3.$$

Solving equation $Q(\rho)'_\rho = 0$, we obtain the following roots

$$\begin{aligned}
\rho_1 &= \frac{1}{2} - \frac{3\sqrt[3]{9}m}{2(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}} \\
&\quad + \frac{\sqrt[3]{3}(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}}{2m}; \\
\rho_2 &= \frac{1}{2} + \frac{3\sqrt[3]{9}(1 + i\sqrt{3})m}{4(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}} \\
&\quad - \frac{\sqrt[3]{3}(1 - i\sqrt{3})(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}}{2m}; \\
\rho_3 &= \frac{1}{2} + \frac{3\sqrt[3]{9}(1 - i\sqrt{3})m}{4(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}} \\
&\quad - \frac{\sqrt[3]{3}(1 + i\sqrt{3})(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}}{2m}.
\end{aligned}$$

The only real root is ρ_1. Since

$$\begin{aligned}
&\frac{\dfrac{3\sqrt[3]{9}m}{2(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}}}{\dfrac{\sqrt[3]{3}(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{1/3}}{2m}} \\
&= \frac{3\sqrt[3]{3}m^2}{(8m^2 + 2m^3 + m^2\sqrt{64 + 32m + 85m^2})^{2/3}}
\end{aligned}$$

$$\begin{aligned} &\leq \frac{3\sqrt[3]{3}m^2}{(2+\sqrt{85})^{2/3}m^2} \\ &= \frac{3\sqrt[3]{3}}{(2+\sqrt{85})^{2/3}} \\ &\approx 0.8633 < 1, \end{aligned}$$

we have

$$\begin{aligned} &\frac{3\sqrt[3]{9}m}{2(8m^2+2m^3+m^2\sqrt{64+32m+85m^2})^{1/3}} \\ &< \frac{\sqrt[3]{3}(8m^2+2m^3+m^2\sqrt{64+32m+85m^2})^{1/3}}{2m}. \end{aligned}$$

Thus it holds that $\rho_1 > 1/2$. Since $m \geq 2$, we obtain that if $\rho \in [0, 1/m]$, then $Q(\rho)'_\rho > 0$ because of Proposition 4.3, i.e., $Q(\rho)$ is increasing in ρ. Therefore we just need to check the value of $Q(\rho)$ at the end points 0 and $1/m$.

$$\begin{aligned} \max_{\rho\in[0,1/m]} Q(\rho) &= Q(\rho)|_{\rho=1/m} = -12 + 2m + (20m+24)/m \\ &\quad -15m/m^2 + 2m/m^3 - m/m^4 \\ &= 8 + 2m + 9/m + 2/m^2 - 1/m^3 \\ &\geq 8 + 2m + 9/m + 1/m^2 > 0 \\ \min_{\rho\in[0,1/m]} Q(\rho) &= Q(\rho)|_{\rho=0} = 2m - 12. \end{aligned}$$

Hence, when $m \geq 6$, $Q(\rho)$ is always nonnegative for all $\rho \in [0, 1/m]$, and when $m < 6$, $Q(\rho)$ can be negative in some subset of the interval $[0, 1/m]$. We need to consider the following two cases:

- **CASE 1**: $m \geq 6$;
- **CASE 2**: $m < 6$.

In **CASE 1**, $A'(\rho)_\rho$ is nonnegative. So $A(\rho)$ is nondecreasing in ρ. To obtain the minimum value of $A(\rho)$ we should set $\rho = 0$. So

$$\min_{\rho\in[0,1/m]} A(\rho) = A(\rho)|_{\rho=0} = \frac{2m\cdot 4}{2\cdot 1\cdot(m+2)} = \frac{4m}{m+2}.$$

Therefore in this case the approximation ratio is $r = 4m/(m+2)$ by setting $\rho^* = 0$ and $\mu^* = m/2$.

In **CASE 2**, the minimum value of $A(\rho)$ is at neither the point $\rho = 0$ nor $\rho = 1/m$, but a point $\bar{\rho} \in (0, 1/m)$. Thus we shall check all possibilities for $m = 2, 3, 4, 5$. We here use Proposition 4.2 for solving quadratic equations for these cases.

When $m = 2$, $\mu \in [m/2, (m+1)/2] = [1, 3/2]$, so we choose $\mu^* = 1$. So

$$A(\rho) = \frac{4/(2-\rho) + 2(2-1)/(1+\rho)}{2-1+1} = \frac{4+\rho}{2+\rho-\rho^2},$$

and its first order partial derivative with respect to ρ is

$$A'(\rho)_\rho = \frac{\rho^2+8\rho-2}{(2+\rho-\rho^2)^2}.$$

Solving equation $A'(\rho)_\rho = 0$, we obtain $\rho = -4 \mp 3\sqrt{2}$. Since $\rho \in [0, 1/m] = [0, 1/2]$, we obtain that the optimal $\rho^* = 3\sqrt{2} - 4$. The approximation ratio

$$r \le A(\rho)|_{\rho=3\sqrt{2}-4} = 1 + 2\sqrt{2}/3 \approx 1.942809.$$

When $m = 3$, $\mu \in [m/2, (m+1)/2] = [3/2, 2]$, so we choose $\mu^* = 2$. So

$$A(\rho) = \frac{6/(2-\rho) + 2(3-2)/(1+\rho)}{3-2+1} = \frac{5+2\rho}{2+\rho-\rho^2},$$

and its first order partial derivative with respect to ρ is

$$A'(\rho)_\rho = \frac{2\rho^2+10\rho-1}{(2+\rho-\rho^2)^2}.$$

Solving equation $A'(\rho)_\rho = 0$, we obtain $\rho = (-5 \mp 3\sqrt{3})/2$. Since $\rho \in [0, 1/m] = [0, 1/3]$, we obtain that the optimal $\rho^* = (3\sqrt{3} - 5)/2$. The approximation ratio

$$r \le A(\rho)|_{\rho=(3\sqrt{3}-5)/2} = 2(2+\sqrt{3})/3 \approx 2.488034.$$

When $m = 4$, $\mu \in [m/2, (m+1)/2] = [2, 5/2]$, so we choose $\mu^* = 2$. So

$$A(\rho) = \frac{8/(2-\rho) + 2(4-2)/(1+\rho)}{4-2+1} = \frac{4}{3}\frac{4+\rho}{2+\rho-\rho^2}.$$

Same as in the case of $m = 2$, we obtain the optimal $\rho^* = 3\sqrt{2} - 4 \in [0, 1/m] = [0, 1/4]$. The approximation ratio

$$r \leq A(\rho)|_{\rho=3\sqrt{2}-4} = 4(1 + 2\sqrt{2}/3)/3 \approx 2.590412.$$

When $m = 5$, $\mu \in [m/2, (m+1)/2] = [5/2, 3]$, so we choose $\mu^* = 3$. So

$$A(\rho) = \frac{10/(2-\rho) + 2(5-3)/(1+\rho)}{5-3+1} = \frac{6+2\rho}{2+\rho-\rho^2},$$

and its first order partial derivative with respect to ρ is

$$A'(\rho)_\rho = \frac{2(\rho^2+6\rho-1)}{(2+\rho-\rho^2)^2}.$$

Solving equation $A'(\rho)_\rho = 0$, we obtain $\rho = -3 \mp \sqrt{10}$. Since $\rho \in [0, 1/m] = [0, 1/5]$, we obtain that the optimal $\rho^* = \sqrt{10} - 3$. The approximation ratio

$$r \leq A(\rho)|_{\rho=\sqrt{10}-3} = 2(7 + 2\sqrt{10})/9 \approx 2.961012.$$

Combine all above cases we have the following lemma:

Lemma 4.21 *In the case that $\rho \leq 2\mu/m - 1$, the approximation ratio of Algorithm III is*

$$r \leq \begin{cases} 1 + 2\sqrt{2}/3, & \text{if } m = 2; \\ 2(2+\sqrt{3})/3, & \text{if } m = 3; \\ 4(1+2\sqrt{2}/3)/3, & \text{if } m = 4; \\ 2(7+2\sqrt{10})/9, & \text{if } m = 5; \\ 4m/(m+2), & \text{otherwise.} \end{cases}$$

4.5.1.2 Solve (4.44) for the case $\rho > 2\mu/m - 1$

In this case we need to solve the following min-max nonlinear program:

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \quad & \frac{\frac{2m}{2-\rho} + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} \\ \text{s.t.} \quad & \frac{1+\rho}{2}x_1 + \frac{\mu}{m}x_2 \le 1; \\ & x_1, x_2 \ge 0; \\ & \rho \in \left(\max\left\{\frac{2\mu}{m} - 1, 0\right\}, 1\right]; \\ & \mu \in \left\{1, \ldots, \left\lfloor \frac{m+1}{2} \right\rfloor\right\}. \end{aligned} \tag{4.48}$$

We notice that the constraints on x_1 and x_2 in (4.48) forms a triangle, and the extreme points are $E_1 : (x_1, x_2) = (2/(1+\rho), 0)$, $E_2 : (x_1, x_2) = (0, m/\mu))$, and $E_3 : (x_1, x_2) = (0, 0)$. Since (4.48) is linear in x_1 and x_2, for a fixed pair of ρ and μ, the maximum value of th eobjective function exists at one of the extreme points. It is clear that the objective function can not attain the maximum value at E_3. So we just consider E_1 and E_2. Denote by $A(\mu, \rho)$ and $B(\mu, \rho)$ the objective values at the E_1 and E_2, respectively. Then we have:

$$\begin{aligned} A(\mu,\rho) &= \frac{2(1+\rho)m + 2(2-\rho)(m-\mu)}{(1+\rho)(2-\rho)(m-\mu+1)}; \\ B(\mu,\rho) &= \frac{2m\mu + (2-\rho)m(m-2\mu+1)}{\mu(2-\rho)(m-\mu+1)}. \end{aligned}$$

The first order partial derivative of $A(\mu, \rho)$ with respect to μ is

$$\begin{aligned} A'(\mu,\rho)_\mu &= \frac{2[-(2-\rho)(m-\mu+1) + (1+\rho)m + (2-\rho)(m-\mu)]}{(1+\rho)(2-\rho)(m-\mu+1)^2} \\ &= \frac{2((1+\rho)m - (2-\rho))}{(1+\rho)(2-\rho)(m-\mu+1)^2}. \end{aligned}$$

It is obvious that the denominator is always positive. The numerator is nonnegative when $(1+\rho)m - (2-\rho) \ge 0$, i.e., $\rho \ge (2-m)/(m+1)$. This inequality is always true as $\rho \ge 0$ and $m \ge 2$. Thus $A'(\rho)_\mu$ is always nonnegative. So for any $m \ge 2$, $A(\mu, \rho)$ is increasing in μ.

Furthermore, the first order partial derivative of $B(\mu,\rho)$ with respect to μ is

$$\begin{aligned} B'(\mu,\rho)_\mu &= \frac{1}{(2-\rho)\mu^2(m-\mu+1)^2}\{[2m-2(2-\rho)m]\mu(m-\mu+1) \\ &\quad -[2m\mu+(2-\rho)m(m-2\mu+1)](m-2\mu+1)\} \\ &= \frac{-2(1-\rho)m\mu^2+2(2-\rho)m(m+1)\mu-(2-\rho)m(m+1)^2}{(2-\rho)\mu^2(m-\mu+1)^2}. \end{aligned}$$

When $\rho = 1$,

$$B'(\mu,\rho)_\mu = \frac{m(m+1)[2\mu-(m+1)]}{\mu^2(m-\mu+1)^2} \leq 0$$

as $\mu \leq (m+1)/2$. So $B(\mu,\rho)$ is decreasing in μ. Now we consider the case that $\rho < 1$. Solving the quadratic equation $B'(\mu,\rho)_\mu = 0$ by Proposition 4.2, we obtain the following roots:

$$\begin{aligned} \mu &= \frac{-2(2-\rho)m(m+1) \mp \sqrt{4(2-\rho)^2m^2(m+1)^2-8(1-\rho)m(2-\rho)m(m+1)^2}}{-4(1-\rho)m} \\ &= \frac{2-\rho \pm \sqrt{\rho(2-\rho)}}{1-\rho}\cdot\frac{m+1}{2}. \end{aligned}$$

Since $\rho < 1$, $(\rho-1)^2 = \rho^2-2\rho+1 > 0$. So $1 > \rho(2-\rho)$. Because both sides are positive, we can take the square roots of both sides to obtain $1 > \sqrt{\rho(2-\rho)}$. Thus $2-\rho-\sqrt{\rho(2-\rho)} > 1-\rho$. So we obtain that

$$\frac{2-\rho+\sqrt{\rho(2-\rho)}}{1-\rho} > \frac{2-\rho-\sqrt{\rho(2-\rho)}}{1-\rho} > 1.$$

Therefore the roots of the equation $B'(\mu,\rho)_\mu = 0$ violate the constraint $\mu \leq (m+1)/2$. So there is no feasible root for this equation. Since in the numerator of $B'(\mu,\rho)_\mu$ the coefficient of the term of μ^2 is negative, we have $B'(\mu,\rho)_\mu < 0$ for all feasible pair ρ and μ from Proposition 4.3. According to Lemma 4.7, if we find a solution to the equation $A(\mu,\rho) = B(\mu,\rho)$, the value is the approximation ratio. The equation $A(\mu,\rho) = B(\mu,\rho)$ is as follows:

$$\frac{2(1+\rho)m + 2(2-\rho)(m-\mu)}{(1+\rho)(2-\rho)(m-\mu+1)} = \frac{2m\mu + (2-\rho)m(m-2\mu+1)}{\mu(2-\rho)(m-\mu+1)}.$$

After simplification (with elimination of positive common factors $(2-\rho)(m-\mu+1)$ and $2-\rho$), the equation is equivalent to

$$2\mu^2 - (4+2\rho)m\mu + (1+\rho)m(m+1) = 0.$$

According to Proposition 4.2, the roots to this equation are

$$\begin{aligned}\mu &= \frac{(4+2\rho)m \pm \sqrt{(4+2\rho)^2m^2 - 8(1+\rho)m(m+1)}}{4} \\ &= \frac{(2+\rho)m \pm \sqrt{(\rho^2+2\rho+2)m^2 - 2(1+\rho)m}}{2}.\end{aligned}$$

Since $m \geq 1$, $(\rho^2+2\rho+2)m^2 - 2(1+\rho)m \geq \rho^2m^2$. So

$$\frac{(2+\rho)m + \sqrt{(\rho^2+2\rho+2)m^2 - 2(1+\rho)m}}{2} \geq \frac{2m+2\rho m}{2} > \frac{m+1}{2},$$

which violates the constraint that $\mu \leq (m+1)/2$. So the only root to the equation $A(\mu,\rho) = B(\mu,\rho)$ is

$$\mu^* = \frac{(2+\rho)m - \sqrt{(\rho^2+2\rho+2)m^2 - 2(1+\rho)m}}{2}. \tag{4.49}$$

Similar to the analysis of Subsubsection 4.4.1.2, we have the following lemma:

Lemma 4.22 *For a fixed $\rho > 2\mu/m - 1$, the optimal objective value of (4.48) is bounded by the optimal objective value for μ^* in (4.49).*

4.5.2 Approximation ratio of Algorithm III

According to the analysis in Subsubsection 4.5.1.1, when $\rho \leq 2\mu/m - 1$, if $m \geq 6$, we just need to set $\rho^* = 0$ and $\mu^* = \lceil m/2 \rceil$ to obtain the optimal value of (4.44), i.e., the approximation ratio of Algorithm III. For the special cases of $m = 2, 3, 4, 5$,

optimal ρ^* and μ^* can be chosen according to Subsubsection 4.5.1.1. The ratio is listed in Lemma 4.21.

Now we investigate the case $\rho > 2\mu/m - 1$ based on Subsubsection 4.5.1.2. Unfortunately, we will show in Subsection 4.5.3 that we are not able to use the technique in Section 4.3 to obtain the optimal value of (4.29) over ρ. Thus, similar to Section 4.4, in this case we still can fix the value of ρ to obtain an improved approximation ratio. We also show that it is asymptotically the optimal choice. The value of ρ is set as follows:

$$\hat{\rho}^* = 0.26. \tag{4.50}$$

By substituting it to (4.49) we set

$$\hat{\mu}^* = \frac{113m - \sqrt{6469m^2 - 6300m}}{100}. \tag{4.51}$$

We need to examine if $\hat{\rho}^*$ and $\hat{\mu}^*$ in (4.50) and (4.51) satisfy the assumption that $\hat{\rho}^* \geq 2\hat{\mu}^*/m - 1$. Since $m \geq 2 > 6300/3969$, $2500 < 6469 - 6300/m$. Because both sides are positive, taking the square root we have $50 < \sqrt{6369 - 6300/m}$. Therefore $\hat{\rho}^* = 13/50 > (63 - \sqrt{6369 - 6300/m})/50 = 2\hat{\mu}^*/m - 1$.

Lemma 4.23 *In the case that $\rho \geq 2\mu/m - 1$, Algorithm III has an approximation ratio r upper bounded by*

$$\frac{100}{63} + \frac{100}{345303}\frac{(63m - 87)(\sqrt{6469m^2 - 6300m} + 13m)}{m^2 - m}.$$

Proof: It worth noting that the $\hat{\mu}^*$ in (4.51) can be a fractional number. Therefore we need to consider $\lceil \hat{\mu}^* \rceil$ and $\lfloor \hat{\mu}^* \rfloor$. Since we should minimize the objective function over μ, the approximation ratio with integer value of μ is bounded as follows:

$$r \leq \min\{\max\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*)\}, \max\{A(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\}\}.$$

According to the analysis in Subsubsection 4.5.1.2, here $A(\mu, \rho)$ is increasing in μ and $B(\mu, \rho)$ is decreasing in μ for a fixed ρ. Thus the bound on approximation ratio is

$$r \leq \min\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\}$$

Furthermore, $\lceil \hat{\mu}^* \rceil \leq \hat{\mu}^* + 1$ and $\lfloor \hat{\mu}^* \rfloor \geq \hat{\mu}^* - 1$. Again, because $A(\mu, \rho)$ is increasing and $B(\mu, \rho)$ is decreasing, we have

$$\begin{aligned} A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*) &\leq A(\hat{\mu}^* + 1, \hat{\rho}^*); \\ B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*) &\leq B(\hat{\mu}^* - 1, \hat{\rho}^*). \end{aligned}$$

Thus we have the following bound on the ratio r:

$$\begin{aligned} r &\leq \min\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\} \\ &\leq \min\{A(\hat{\mu}^* + 1, \hat{\rho}^*), B(\hat{\mu}^* - 1, \hat{\rho}^*)\} \\ &\leq A(\hat{\mu}^* + 1, \hat{\rho}^*). \end{aligned}$$

Therefore here we shall find an upper bound on $A(\hat{\mu}^* + 1, \hat{\rho}^*)$, which is also an upper bound on the approximation ratio r. Substituting $\hat{\mu}^*$ in (4.51) and $\hat{\rho}^*$ in (4.50) in $A(\hat{\mu}^* + 1, \hat{\rho}^*)$ gives:

$$\begin{aligned} r \leq A(\hat{\mu}^* + 1, \hat{\rho}^*) &= \frac{2}{1+\hat{\rho}^*} + \frac{2}{1+\hat{\rho}^*} \frac{(1+\hat{\rho}^*)m - (2-\hat{\rho}^*)}{(2-\hat{\rho}^*)(m-\hat{\mu}^*)} \\ &= \frac{100}{63} + \frac{100}{63} \frac{63m - 87}{87\left(\frac{\sqrt{6469m^2 - 6300m} - 13m}{100}\right) + 1} \\ &= \frac{100}{63} + \frac{100}{345303} \frac{(63m-87)(\sqrt{6469m^2 - 6300m} + 13m)}{m^2 - m}. \end{aligned}$$

This is the claimed bound in the theorem. □

Combine Lemma 4.21 and Lemma 4.23 we have the following theorem of the approximation ratio of Algorithm III:

Theorem 4.5 *If the work functions $W_j(x)$ are convex with respect to processing times for $j = 1, \ldots, n$, then there exists an algorithm for the problem of* Scheduling Malleable Tasks with Precedence Constraints *with an approximation ratio*

$$r \leq \begin{cases} 1+2\sqrt{2}/3, & \text{if } m=2; \\ 2(2+\sqrt{3})/3, & \text{if } m=3; \\ 4(1+2\sqrt{2}/3)/3, & \text{if } m=4; \\ 2(7+2\sqrt{10})/9, & \text{if } m=5; \\ \dfrac{100}{63}+\dfrac{100}{345303}\dfrac{(63m-87)(\sqrt{6469m^2-6300m}+13m)}{m^2-m}, & \text{otherwise.} \end{cases}$$

Proof: We need to compare the minimum objective values in both cases $\rho \leq 2\mu/m-1$ and $\rho > 2\mu/m-1$. Thus for $m \geq 6$ we need to compare $4m/(m+2)$ and the value in Lemma 4.23. Suppose that

$$\frac{4m}{m+2} \geq \frac{100}{63}+\frac{100}{5481}\frac{(63m-87)(\sqrt{6469m^2-6300m}+13m-100)}{63m^2-37m-100}.$$

By moving the right hand side to the left hand side and simplification, we obtain

$$\frac{62601m^3-112501m^2+142700m-25(63m^2+13m-58)\sqrt{6469m^2-6300m}}{(m+2)(m^2-m)} \geq 0.$$

Denote by NUM the numerator, and by DEN the denominator, of the left hand side of the above inequality, respectively. It is obvious that $DEN > 0$ for any $m \geq 2$. Now we consider NUM. Solving equation $NUM = 0$ numerically we obtain the following roots:

$$\begin{aligned} m_1 &= 1.35285; \\ m_{2,3} &= 2.27502 \mp 1.68612i; \\ m_{4,5} &= -0.230259 \mp 0.779709i. \end{aligned}$$

It means that when $m \geq 2 > m_1$, both NUM and DEN are positive according to Proposition 4.3. Therefore for any integer $m \geq 6$ the $\hat{\rho}^*$ and $\hat{\mu}^*$ should be taken by (4.50) and (4.51) to obtain the approximation ratio bounded in Lemma 4.23. Then we need to compare the ratios according to Lemma 4.21 and Lemma 4.23 for $m = 2, 3, 4, 5$. When $m = 2$, by Lemma 4.23, $r \leq 2.384810$, which is greater than the bound in Lemma 4.21. So we should take $\rho^* = 3\sqrt{2}-4$ and $\mu^* = 1$, with an

approximation ratio $r \leq 1 + 2\sqrt{2}/3$. When $m = 3$, by Lemma 4.23, $r \leq 2.755556$, which is greater than the bound in Lemma 4.21. So we should take $\rho^* = (3\sqrt{3}-5)/2$ and $\mu^* = 2$, with an approximation ratio $r \leq 2(2 = \sqrt{3})/3$. When $m = 4$, by Lemma 4.23, $r \leq 2.908646$, which is greater than the bound in Lemma 4.21. So we should take $\rho^* = 3\sqrt{2} - 4$ and $\mu^* = 2$, with an approximation ratio $r \leq 4(1 + 2\sqrt{2})/3$. When $m = 4$, by Lemma 4.23, $r \leq 2.993280$, which is greater than the bound in Lemma 4.21. So we should take $\rho^* = \sqrt{10} - 3$ and $\mu^* = 3$, with an approximation ratio $r \leq 2(7 + 2\sqrt{10})/9$. The theorem is proved. □

Then the following corollary holds for the upper bound on the approximation ratios:

Corollary 4.4 *For all $m \in \mathbb{N}$ and $m \geq 2$, the approximation ratio*

$$r \leq \frac{100}{63} + \frac{100(\sqrt{6469} + 13)}{5481}.$$

Furthermore, when $m \to \infty$, the upper bound in Theorem 4.5 tends to

$$\frac{100}{63} + \frac{100(\sqrt{6469} + 13)}{5481} \approx 3.291919.$$

Proof: It is obvious that when $m \leq 6$, the approximation ratios fulfils the inequality. We now consider the case that $m \geq 6$.

Similarly to the proof of Corollary 4.2, we need to show that

$$\frac{100}{63}+\frac{100}{5481}\frac{(63m-87)(\sqrt{6469m^2-6300m}+13m-100)}{m^2-m} \leq \frac{100}{63}+\frac{100(\sqrt{6469}+13)}{5481},$$

which is equivalent to

$$\frac{63m-87}{63}\frac{\sqrt{6469m^2-6300m}+13m}{m^2-m} \leq \sqrt{6469}+13.$$

Since $m^2 - m > 0$ for all $m \geq 2$, we can multiply both sides by $63(m^2 - m)$. Then simplification gives

$$(21m-29)\sqrt{6469m^2-6300m} \leq 21\sqrt{6469}m^2 - 21\sqrt{6469}m + 104m.$$

When $m \geq 2$, $21m - 29 > 0$ and $m^2 - m > 0$, so both sides are positive. We can take square of both sides and obtain the following inequality:

$$(2475942 + 2184\sqrt{6469})m^2 + (315337 - 2184\sqrt{6469})m - 2649150 \geq 0. \qquad (4.52)$$

It is easy to verify that $2475942+2184\sqrt{6469} \approx 2651601.325 > 2649150$ and $315337-2184\sqrt{6469} \approx 139677.675 > 0$. Therefore for any $m \geq 1$ the inequality (4.52) holds. So the inequality in the corollary is proved.

When $m \to \infty$, the upper bound in Theorem 4.5

$$\begin{aligned}
& \frac{100}{63} + \frac{100}{5481}\frac{(63m-87)(\sqrt{6469m^2-6300m}+13m-100)}{63m^2-37m-100} \\
= \; & \frac{100}{63} + \frac{100}{5481}\frac{(63-87/m)(\sqrt{6469-6300/m}+13-100/m)}{63-37/m-100/m^2} \\
\to \; & \frac{100}{63} + \frac{100}{5481}\frac{63(\sqrt{6469}+13)}{63} \\
= \; & \frac{100}{63} + \frac{100(\sqrt{6469}+13)}{5481} \approx 3.291919.
\end{aligned}$$

□

We here give the list of values of approximation ratios for our Algorithm III for $m = 2, \ldots, 33$ in Table 4.6 (See Appendix for codes). Here it is worth noting that we still take $\hat{\rho}^* = 0.26$ for $m = 5$, as the bound in Lemma 4.23 is only an upper bound on the objective value of (4.44) with $\rho = \hat{\rho}^* = 0.26$. In fact with the rounded value of $\lfloor \hat{\mu}^* \rfloor$ or $\lceil \hat{\mu}^* \rceil$ the objective values are lower than the bound in Lemma 4.23 as listed above.

4.5.3 Asymptotic behaviour of approximation ratio for convex work functions

In Algorithm III we set $\hat{\rho}^* = 0.26$. However, the approximation ratio r can be improved by choosing the value of ρ^* depending on m, like that in Algorithm I and in Subsection 4.4.3. In this subsection we are going to study it.

Recall that μ^* in (4.49) is the minimizer of the objective function in (4.44). By substituting μ^* to $A(\mu, \rho)$ or $B(\mu, \rho)$ we can obtain two functions $A(\rho)$ or $B(\rho)$.

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.243	1.9428	18	7	0.260	3.1792
3	2	0.098	2.4880	19	7	0.260	3.1451
4	2	0.243	2.5904	20	7	0.260	3.1160
5	2	0.260	2.6868	21	8	0.260	3.1981
6	3	0.260	2.9146	22	8	0.260	3.1673
7	3	0.260	2.8790	23	8	0.260	3.1404
8	3	0.260	2.8659	24	8	0.260	3.2110
9	4	0.260	3.0469	25	9	0.260	3.1843
10	4	0.260	3.0026	26	9	0.260	3.1594
11	4	0.260	2.9693	27	9	0.260	3.2123
12	5	0.260	3.1130	28	10	0.260	3.1976
13	5	0.260	3.0712	29	10	0.260	3.1746
14	5	0.260	3.0378	30	10	0.260	3.2135
15	6	0.260	3.1527	31	11	0.260	3.2085
16	6	0.260	3.1149	32	11	0.260	3.1870
17	6	0.260	3.0834	33	11	0.260	3.2144

Table 4.6: Bounds on approximation ratios for Algorithm III.

Since our goal is to find the minimum value of $A(\rho)$ or $B(\rho)$ over all ρ, we need to solve the equation $A'(\rho)_\rho = 0$ or $B'(\rho)_\rho = 0$. Because $A(\rho) = B(\rho)$, we just need to consider one of them, say, $A(\rho)$. The first order partial derivative of $A(\rho)$ with respect to ρ is

$$\begin{aligned} A'(\rho)_\rho &= \left[\frac{2m}{(2-\rho)(m-\mu^*+1)} + \frac{2}{1+\rho} - \frac{2}{(1+\rho)(m-\mu^*+1)}\right]'_\rho \\ &= \frac{2m((m-\mu^*+1)+(2-\rho)(\mu^*)'_\rho)}{(2-\rho)^2(m-\mu^*+1)^2} - \frac{2}{(1+\rho)^2} \\ &\quad + \frac{2((m-\mu^*+1)-(1+\rho)(\mu^*)'_\rho)}{(1+\rho)^2(m-\mu^*+1)^2} \end{aligned}$$

Combine the two terms together and the denominator is positive. So the equation

$A'(\rho)_\rho = 0$ can be simplified as follows:

$$
\begin{aligned}
&-(2-\rho)^2(\mu^*)^2 + [(\rho^2 - 10\rho + 7)m + (2-\rho)^2]\mu^* \\
&+(1+\rho)(2-\rho)[(1+\rho)m - (2-\rho)](\mu^*)'_\rho + 3(2\rho-1)m(m+1) = 0.
\end{aligned}
$$

Here

$$
\begin{aligned}
\mu^* &= \frac{m(2+\rho)}{2} - \frac{\sqrt{\Delta}}{2}; \\
(\mu^*)^2 &= \frac{(\rho^2+3\rho+3)m^2 - (\rho+1)m}{2} - \frac{(2+\rho)m\sqrt{\Delta}}{2}; \\
(\mu^*)'_\rho &= \frac{m}{2} - \frac{(\rho+1)m^2 - m}{2\sqrt{\Delta}},
\end{aligned}
$$

and $\Delta = (\rho^2 + 2\rho + 2)m^2 - 2(1+\rho)m$. Substituting them to the equation and we obtain the following equation:

$$A_1\Delta + A_2\sqrt{\Delta} + A_3 = 0,$$

where the coefficients are as follows (with elimination of a common factor 1/2):

$$
\begin{aligned}
A_1 &= m\rho^3 + (-3m-1)\rho^2 + (6m+4)\rho + (m-4); \\
A_2 &= m[-m\rho^4 + (m+1)\rho^3 + (-3m-2)\rho^2 + (2m+8)\rho + (-2m+2)]; \\
A_3 &= m[(m^2+m)\rho^4 + (m^2-3m-1)\rho^3 + (-3m^2-3m+3)\rho^2 \\
&\quad +(-5m^2+7m)\rho + (-2m^2+6m-4)].
\end{aligned}
$$

To remove the square root, we can simplify the equation to an equivalent equation

$$(A_1\Delta + A_3)^2 - A_2^2\Delta = 0.$$

After simplification, it can be written as the following form:

$$m^2(1+m)(1+\rho)^2 \sum_{i=0}^{6} c_i\rho^i = 0, \tag{4.53}$$

where the coefficients are as follows:

$$
\begin{aligned}
c_0 &= -8(m-1)^2(m-2);\\
c_1 &= 8(m-1)(m-2)(3m-2);\\
c_2 &= 21m^3-59m^2+16m+24;\\
c_3 &= 2(m+1)(7m^2-7m-4);\\
c_4 &= 3m^3-7m^2+15m+1;\\
c_5 &= 2m(3m^2-4m-1);\\
c_6 &= m^2(m+1).
\end{aligned}
$$

Then with eliminating the common factor $(\rho+1)^2$ we are able to obtain an equation with highest order of 6. Unfortunately in general there are no analytic roots for polynomial with order higher than 4. So we are not able to solve (4.53) to obtain the optimal ρ^* depending on m like in Section 4.3.

In fact we can estimate the asymptotic behaviour of the approximation ratio. When $m \to \infty$, equation (4.53) is:

$$
\begin{aligned}
0 &= -8(m-1)^2(m-2+8(m-1)(m-2)(3m-2)\rho\\
&\quad +(21m^3-59m^2+16m+24)\rho^2+2(m+1)(7m^2-7m-4)\rho^3\\
&\quad +(3m^3-7m^2+15m+1)\rho^4+2m(3m^2-4m-1)\rho^5+m^2(m+1)\rho^6\\
&= m^3\left[-8\left(1-\frac{1}{m}\right)^2\left(1-\frac{2}{m}\right)+8\left(1-\frac{1}{m}\right)\left(1-\frac{2}{m}\right)\left(3-\frac{2}{m}\right)\rho\right.\\
&\quad +\left(21-\frac{59}{m}+\frac{16}{m^2}+\frac{24}{m^3}\right)\rho^2+2\left(1+\frac{1}{m}\right)\left(7-\frac{7}{m}-\frac{4}{m^2}\right)\rho^3\\
&\quad \left.+\left(3-\frac{7}{m}+\frac{15}{m^2}+\frac{1}{m^3}\right)\rho^4+2\left(3-\frac{4}{m}-\frac{1}{m^2}\right)\rho^5+\left(1+\frac{1}{m}\right)\rho^6\right]\\
&\to m^3(\rho^6+6\rho^5+3\rho^4+14\rho^3+21\rho^2+24\rho-8).
\end{aligned}
$$

Thus we just need to consider the equation $\rho^6+6\rho^5+3\rho^4+14\rho^3+21\rho^2+24\rho-8$. Solving it by numerical methods, we have the following roots:

$$\rho_1 = -5.8353;$$

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.243	1.9428	18	6	0.143	3.1065
3	2	0.098	2.4880	19	7	0.328	3.1384
4	2	0.243	2.5904	20	7	0.300	3.1092
5	2	0.200	2.6389	21	7	0.167	3.1273
6	3	0.243	2.9142	22	8	0.331	3.1600
7	3	0.292	2.8777	23	8	0.304	3.1330
8	3	0.250	2.8571	24	8	0.185	3.1441
9	3	0.000	3.0000	25	9	0.333	3.1765
10	4	0.310	2.9992	26	9	0.308	3.1515
11	4	0.273	2.9671	27	9	0.200	3.1579
12	4	0.067	3.0460	28	10	0.335	3.1895
13	5	0.318	3.0664	29	10	0.310	3.1663
14	5	0.286	3.0333	30	10	0.212	3.1695
15	5	0.111	3.0802	31	10	0.129	3.1972
16	6	0.325	3.1090	32	11	0.312	3.1785
17	6	0.294	3.0776	33	11	0.222	3.1794

Table 4.7: Numerical results of min-max nonlinear program (4.48).

$$\begin{aligned} \rho_{2,3} &= -0.949632 \pm 0.89448i; \\ \rho_4 &= 0.261917; \\ \rho_{5,6} &= 0.72544 \pm 1.60027i. \end{aligned}$$

The only feasible root here in the interval $\rho \in (0, 1)$ is $\rho^* = 0.261917$. Substituting it to (4.49) the optimal $\mu^* \to 0.325907m$. With these data, from either A or B one have that

$$r \to 3.291913.$$

In Algorithm III we fix $\hat{\rho}^* = 0.26$ just because it is close to the asymptotic optimal ρ^*. The ratio of Algorithm III could be further improved by fix $\hat{\rho}^*$ to a better approximation to ρ^*. In this way we conjecture that there exists a 3.291913-approximation algorithm for the problem of Scheduling Malleable Tasks with Precedence

CONSTRAINTS However, the analysis is complicated and Algorithm III has already a ratio 3.291919 very close to this asymptotic ratio.

We can also use numerical method to solve the min-max nonlinear program (4.48). We can construct a grid of ρ in the interval $[0, 1]$, and μ in $[1, \lfloor (m+1)/2 \rfloor]$. The grid size for ρ is $\delta\rho$ and for μ is 1 as μ is an integer. We can compute the values of $A(\mu, \rho)$ and $B(\mu, \rho)$ on each grid point, and search for the minimum over all grid points to decide the optimal objective values depending on m (See Appendix for codes). The results by setting $\delta\rho = 0.0001$ and $m = 2, \ldots, 33$ are in Table 4.7. Compared the results in Table 4.6 we can see that the solutions of our Algorithm III are already very close to the optimum.

Bibliography

[1] A. Agarwal, D. Chaiken, K. Johnson, D. Kranz, J. Kubiatowicz, K. Kurihara, B.-H. Lim, G. Maa, and D. Nussbaum, The MIT Alewife machine: a large-scale distributed-memory multiprocessor, *Proceedings of the 1st Workshop on Scalable Shared Memory Multiprocessors*, 1991.

[2] C. Ambuehl, A. E. F. Clementi, M. D. Ianni, A. Monti, G. Rossi and R. Silvestri, The range assignment problem in non-homogeneous static ad-hoc networks, *Proceedings of the 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks*, WMAN 2004.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, Proof verification and hardness of approximation problems, *Journal of the ACM*, 45 (1998), 501-555.

[4] A. Baltz and A. Srivastav, Fast approximation of multicast congestion, manuscript, 2001.

[5] A. Baltz and A. Srivastav, Fast approximation of minimum multicast congestion - implementation versus theory, *Proceedings of the 5th Conference on Algorithms and Complexity*, CIAC 2003, LNCS 2653, 165-177.

[6] Y. Bartal, Probabilistic approximation of metric spaces and its algorithmic applications, *Proceedings of the 37th IEEE Annual Symposium on Foundations of Computer Science*, FOCS 1996, 184-193.

[7] Y. Bartal, On approximating arbitrary metrics by tree metrics, *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, STOC 1998, 161-168.

[8] E. M. Belding-Royer, C.-K. Toh, A review of current routing protocols for ad-hoc mobile wireless networks, *IEEE Personal Communications Magazine*, April (1999) 46-55.

[9] P. Berman and V. Ramaiyer, Improved approximations for the Steiner tree problem, *Journal of Algorithms*, 17 (1994), 381-408.

[10] M. Bern and P. Plassmann, The Steiner problem with edge lengths 1 and 2, *Information Professing Letters*, 32 (1989), 171-176.

[11] E. Blayo, L. Debreu, G. Mounié and D. Trystram, Dynamic load balancing for ocean circulation with adaptive meshing, *Proceedings of the 5th European Conference on Parallel Computing*, Euro-Par 1999, LNCS 1685, 303-312.

[12] R. L. Burden and J. D. Faires, Numerical analysis (6th edition), Brooks/Cole Publishing Company, 1997.

[13] G. Calinescu, S. Kapoor, A. Olshevsky, A. Zelikovsky, Network lifetime and power assignment in ad hoc wireless networks, *Proceedings of the 11th Annual European Symposium*, ESA 2003, LNCS 2832, 114-126.

[14] G. Calinescu, S. Kapoor, and M. Sarwat, Bounded hops power assignment in ad-hoc wireless networks, *Proceedings of the IEEE Wireless Communications and Networking Conference*, WCNC 2004.

[15] R. Carr and S. Vempala, Randomized meta-rounding, *Proceedings of the 32nd ACM Symposium on the Theory of Computing*, STOC 2000, 58-62.

[16] M. Charikar, C. Chekuri, A. Goel, S. Guha and S. Plotkin, Approximating a finite metric by a small number of tree metrics, *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, FOCS 1998, 379-388.

[17] I. Chlamtac, M. Conti, J. J.-N. Liu, Mobile ad hoc networking: imperactives and challenges, *Ad Hoc Networks*, 1 (2003), 13-64.

[18] M. Chlebík and J. Chlebíková, Approximation hardness of the Steiner tree problem, *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*, SWAT 2002, LNCS 2368, 170-179.

[19] J. Chlebíková, D. Ye and H. Zhang, Assign ranges in general ad-hoc networks, *Proceedings of the 1st International Conference on Algorithmic Applications in Management*, AAIM 2005, LNCS 3521, 411-421.

[20] A. E. F. Clementi, P. Crescenzi, P. Penna, G. Rossi and P. Vocca, On the complexity of computing minimum energy consumption broadcast subgraph. *Proceedings of the 18th Annual Symposium on Theoretical Aspect of Computer Science*, STACS 2001, LNCS 2010, 121-131.

[21] A. E. F. Clementi, A. Ferreira, P. Penna, S. Perennes, R. Silvestri, The minimum range assignment problem on linear radio networks, *Algorithmica*, 35(2) 2003, 95-110.

[22] A. E. F. Clementi, G. Huiban, P. Penna, G. Rossi and Y. C. Verhoeven, Some recent theoretical advances and open questions on energy consumption in ad-hoc wireless networks, *Proceedings of the 3rd International Workshop on Approximation and Randomized Algorithms in Communication Networks*, ARACNE 2002, 23-38.

[23] A. E. F. Clementi, M. D. Ianni and R. Silvestri, The minimum broadcast range assignment problem on linear multi-hop wireless networks, *Theoretical Computer Science*, 1-3 (299) 2003, 751-761.

[24] A. E. F. Clementi, P. Penna, and R. Silvestri, On the power assignment problem in radio networks, *Technical Report TR00-054, Electronic Colloquium on Computational Complexity (ECCC)*, 2000.

[25] D. E. Culler, R. Karp, D. Patterson, A. Sahay, E.Santos, K. Schauser, R. Subramonian and T. von Eicken, LogP: A practical model of parallel computation, *Communications of the ACM*, 39 (11), 1996, 78-85.

[26] D. E. Culler, J. P. Singh and A. Gupta, Parallel computer architecture: A hardware/software approach, Morgan Kaufmann Publishers, San Francisco, USA, 1999.

[27] G. Dantzig, Maximization of a linear function of variables subject to linear inequalities, *in Activity Analysis of Production and Allocation, Tj. C. Koopmans (eds.)*, (1951), 339-347.

[28] K. Dhamdhere, A. Gupta and R. Ravi, Approximation algorithms for minimizing average distortion, *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science*, STACS 2004, LNCS 2996, 234-245.

[29] J. Du and J. Leung, Complexity of scheduling parallel task systems, *SIAM Journal on Discrete Mathematics*, 2 (1989), 473-487.

[30] D. Z. Du, B. Lu, H. Q. Ngo and P. Pardalos, The Steiner tree problem, *in Encyclopedia of Optimization, C. A. Floudas and P. M. Pardalos (eds.)*, Kluwer Academic Publisher, 2001.

[31] P.-F. Dutot, G. Mounié and D. Trystram, Scheduling parallel tasks approximation algorithms, *in Handbook of Scheduling: Algorithms, Models, and Performance Analysis, J. Y-T. Leung (eds.)*, CRC Press, Boca Raton, USA, 2004.

[32] G. Even, J. S. Naor, S. Rao and B. Schieber, Fast approximate graph partitioning algorithms, *SIAM Journal on Computing*, 6 (1999), 2187-2214.

[33] J. Fakcharoenphol, S. Rao and K. Talwar, A tight bound on approximating arbitrary metrics by tree metrics, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, STOC 2003, 448-455.

[34] L. Fleischer, Approximating fractional multicommodity flow independent of the number of commodities, *SIAM Journal on Discrete Mathematics*, 13 (2000), 505-520.

[35] L. Fleischer, A fast approximation scheme for fractional covering problems with variable upper bounds, *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, SODA 2004, 994-1003.

[36] R. Floren, A note on "A faster approximation algorithm for the Steiner problem in graphs", *Information Processing Letters*, 38 (1991), 177-178.

[37] D. R. Fulkerson, A network flow computation for project cost curves, *Management Science*, 7 (1961), 167-178.

[38] C. Gröpl, S. Hougardy, T. Nierhoff and H. J. Prömel, Approximation algorithms for the Steiner tree problem in graphs, *in Steiner Trees in Industry, X. Cheng and D.-Z. Du (eds)*, Kluwer Academic Publishers, (2001), 235-279.

[39] M. Garey and R. Graham, Bounds for multiprocessor scheduling with resource constraints, *SIAM Journal on Computing*, 4 (1975), 187-200.

[40] M. Garey and D. Johnson, Computer and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, NY, 1979.

[41] N. Garg and R. Khandekar, Fractional covering with upper bounds on the variables: solving LPs with negative entries, *Proceedings of the 12th Annual European Symposium on Algorithms*, ESA 2004.

[42] N. Garg and J. Könemann, Fast and simpler algorithms for multicommodity flow and other fractional packing problems, *Proceedings of the 39th IEEE Annual Symposium on Foundations of Computer Science*, FOCS 1998, 300-309.

[43] E. N. Gilbert and H. O. Pollak, Steiner minimal trees, *SIAM Journal on Applied Mathematics*, 16 (1968), 1-29.

[44] R. L. Graham, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal*, 45 (1966), 1563-1581.

[45] M. D. Grigoriadis and L. G. Khachiyan, Fast approximation schemes for convex programs with many blocks and coupling constraints, *SIAM Journal on Optimization*, 4 (1994), 86-107.

[46] M. D. Grigoriadis and L. G. Khachiyan, Coordination complexity of parallel price-directive decomposition, *Mathematics of Operations Research*, 2 (1996), 321-340.

[47] M. D. Grigoriadis and L. G. Khachiyan, Approximate minimum-cost multicommodity flows in $O(\varepsilon^{-2}knm)$ time, *Mathematical Programming*, 75 (1996), 477-482.

[48] M. D. Grigoriadis, L. G. Khachiyan, L. Porkolab and J. Villavicencio, Approximate max-min resource sharing for structured concave optimization, *SIAM Journal on Optimization*, 11 (2001), 1081-1091.

[49] M. Grötschel, L. Lovász and A. Schrijver, The ellipsoid method and its consquences in combinatorial optimization, *Combinatorica*, 1 (1981) 169-197.

[50] A. Gupta, Steiner points in tree metrics don't (really) help, *Proceedings of the 12th ACM/SIAM Annual Symposium on Discrete Algorithms*, SODA 2001, 220-227.

[51] S. Hougardy and H. J. Prömel, A 1.598 approximation algorithm for the Steiner problem in graphs, *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 1999, 448-453.

[52] J. J. Hwang, Y. C. Chow, F. D. Anger and C. Y. Lee, Scheduling precedence graphs in systems with interprocessor communication times, *SIAM Journal on Computing*, 18(2) 1989, 244-257.

[53] K. Jansen, Scheduling malleable parallel tasks: an asymptotic fully polynomial-time approximation scheme, *Proceedings of the 10th European Symposium on Algorithms*, ESA 2002, LNCS 2461, 562-573.

[54] K. Jansen, Approximation algorithms for the general max-min resource sharing problem: faster and simpler, *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory*, SWAT 2004, LNCS 3111, 311-322

[55] K. Jansen, Approximation algorithms for the mixed fractional packing and covering problem, *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science*, TCS 2004.

[56] K. Jansen, Approximation algorithms for fractional covering and packing problems, and applications, manuscript, 2004.

[57] K. Jansen and L. Porkolab, Linear-time approximation schemes for scheduling malleable parallel tasks, *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 1999, 490-498.

[58] K. Jansen and L. Porkolab, On preemptive resource constrained scheduling: polynomial-time approximation schemes, *Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization*, IPCO 2002, LNCS 2337, 329-349.

[59] K. Jansen and R. Solis-Oba, An asymptotic fully polynomial time approximation scheme for bin covering, *Proceedings of the 13th International Symposium on Algorithms and Computation*, ISAAC 2002, LNCS 2518, 175-186.

[60] K. Jansen and H. Zhang, Approximation algorithms for general packing problems with modified logarithmic potential function, *Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science*, TCS 2002, 255-266.

[61] K. Jansen and H. Zhang, An approximation algorithm for the multicast congestion problem via minimum Steiner trees, *Proceedings of the 3rd International Workshop on Approximation and Randomized Algorithms in Communication Networks*, ARACNE 2002, 77-90.

[62] K. Jansen and H. Zhang, Scheduling Malleable Tasks with Precedence Constraints, *to appear in Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA 2005.

[63] K. Jansen and H. Zhang, An Approximation Algorithm for Scheduling Malleable Tasks under General Precedence Constraints, *manuscript*, 2004.

[64] T. Kalinowski, I. Kort and D. Trystram, List scheduling of general task graphs under LogP, *Parallel Computing*, 26(9) 2000, 1109-1128.

[65] G. Karakostas, Faster approximation schemes for fractional multicommodity flow problems, *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithm*, SODA 2002, 166-173.

[66] N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica*, 4 (1984) 373-395.

[67] R. M. Karp, Reducibility among combinatorial problems, *in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher (eds)*, Plenum Press, NY, (1972), 85-103.

[68] R. M. Karp and C. Papadimitriou, On linear characterization of combinatorial optimization problems, *SIAM Journal on Computing*, 11 (1982), 620-632.

[69] M. Karpinski and A. Zelikovsky, New approximation algorithms for the Steiner tree problems, *Journal of Combinatorial Optimization*, 1 (1997), 47-65.

[70] J. E. Kelley, Critical path planning and scheduling: Mathematical bases, *Operations Research*, 9 (1961), 296-320.

[71] C. Kenyon and E. Rémila, Approximate strip packing, *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS 1996, 31-36.

[72] L. Khachiyan, A polynomial algorithm in linear programming, *Soviet Mathematics Doklady*, 20 (1979), 191-194.

[73] R. Khandekar, Lagrangian relaxation based algorithms for convex programming problems, *PhD thesis*, 2004.

[74] L. M. Kirousis, E. Kranakis, D. Krizanc and A. Pelc, Power consumption in packet radio networks, *Theoretical Computer Science*, (243) 2000, 289-305.

[75] P. Klein, S. Plotkin, C. Stein and E. Tardos, Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts, *SIAM Journal on Computing*, 23 (1994), 466-487.

[76] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceedings of the American Mathematical Society*, 7 (1965), 48-50.

[77] G. S. Lauer, Packet radio routing, (chap. 11 of Routing in communications networks, Martha Steenstrup(eds.)), Printice-Hall, Englewood Cliffs, NJ, 1995.

[78] J. K. Lenstra, D. B. Shmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, 24 (1990), 259-272.

[79] J. K. Lenstra and A. H. G. Rinnooy Kan, Complexity of scheduling under precedence constraints, *Operations Research*, 26 (1978), 22-35.

[80] R. Lepère, G. Mounié and D. Trystram, An approximation algorithm for scheduling trees of malleable tasks, *European Journal of Operational Research*, 142(2) 2002, 242-249.

[81] R. Lepère, D. Trystram and G. J. Woeginger, Approximation algorithms for scheduling malleable tasks under precedence constraints, *International Journal of Foundations of Computer Science*, 13(4) 2002, 613-627.

[82] J. Y-T. Leung, Handbook of scheduling: algorithms, models, and performance analysis, CRC Press, Boca Raton, USA, 2004.

[83] W. Ludwig and P. Tiwari, Scheduling malleable and nonmalleable parallel tasks, *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms*, SODA 1994, 167-176.

[84] J. Matoušek, Bi-lipschitz embeddings into low dimensional Euclidean spaces, *Comment. Math. Univ. Carolinae*, 31(3) 1990, 589-600.

[85] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs, *Information Processing Letters*, 27 (1988), 125-128.

[86] G. Mounié, C. Rapine and D. Trystram, Efficient approximation algorithms for scheduling malleable tasks, *Proceedings of the 11th Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA 1999, 23-32.

[87] G. Mounié, C. Rapine and D. Trystram, A 3/2-dual approximation algorithm for scheduling independent monotonic malleable tasks, *manuscript*.

[88] C. E. Perkins, P. Bhagwat, Highly dynamic destination-sequence distence-vector routing (DSDV) for mobile computers, *Computer Communications Review*, October (1994), 234-244.

[89] S. A. Plotkin, D. B. Shmoys and E. Tardos, Fast approximation algorithms for fractional packing and covering problems, *Mathematics of Operations Research*, 2 (1995), 257-301.

[90] G. N. S. Prasanna, Structure Driven Multiprocessor Compilation of Numeric Problems, *Technical Report MIT/LCS/TR-502*, Laboratory for Computer Science, MIT, April 1991.

[91] G. N. S. Prasanna and B. R. Musicus, Generalised multiprocessor scheduling using optimal control, *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA 1991, 216-228.

[92] G. N. S. Prasanna and B. R. Musicus, Generalized multiprocessor scheduling for directed acyclic graphs, *Proceedings of Supercomputing 1994*, 237-246.

[93] G. N. S. Prasanna and B. R. Musicus, The Optimal Control Approach to Generalized Multiprocessor Scheduling, *Algorithmica*, 15(1), (1996), 17-49.

[94] H. J. Prömel, A. Steger, A new approximation algorithm for the Steiner tree problem with performance ratio 5/3, *Journal of Algorithms*, 36 (2000), 89-101.

[95] H. J. Prömel and A. Steger, The Steiner tree problem, a tour through graphs, algorithms and complexity, Vieweg Verlag, Wiesbaden, 2001.

[96] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Technical Journal*, 36 (1957), 1389-1401.

[97] Y. Rabinovich and R. Raz, Lower bounds on the distortion of embedding finite metric spaces in graphs, *Discrete and Computational Geometry*, 19(1) 1998, 79-94.

[98] T. Radzik, Fast deterministic approximation for the multicommodity flow problem, *Mathematical Programming*, 78 (1997), 43-58.

[99] P. Raghavan, Probabilistic construction of deterministic algorithms: Approximating packing integer programs, *Journal of Computer and System Science*, 37 (1988), 130-143.

[100] P. Raghavan and C. Thompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs, *Combinatorica*, 7 (1987), 365-374.

[101] S. Rao and A. W. Richa, New approximation techniques for some ordering problems, *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 1998, 211-218.

[102] G. Robins and A. Zelikovsky, Improved Steiner tree approximation in graphs, *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2000, 770-779.

[103] G. Rossi, The range assignment problem in static ad-hoc wireless networks, *Ph.D. Thesis*, 2003.

[104] M. Skutella, Approximation algorithms for the discrete time-cost tradeoff problem, *Mathematics of Operations Research*, 23 (1998), 909-929.

[105] M. Thimm, On the approximability of the Steiner tree problem, *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*, MFCS 2001, LNCS 2136, 678-689.

[106] J. Turel, J. Wolf and P. Yu, Approximate algorithms for scheduling parallelizable tasks, *Proceedings of the 4th Annual Symposium on Parallel Algorithms and Architectures*, SPAA 1992, 323-332.

[107] S. Vempala and B. Vöcking, Approximating multicast congestion, *Proceedings of the tenth International Symposium on Algorithms and Computation*, ISAAC 1999, LNCS 1741, 367-372.

[108] J. Villavicencio and M. D. Grigoriadis, Approximate structured optimization by cyclic block-coordinate descent, *Applied Mathematics and Parallel Computing - Festschrift for Klaus Ritter, H. Fischer, B. Riedmüller and S. Schäffler (eds.)*, Physica-Verlag, Heidelberg, 1996, 359-371.

[109] J. Villavicencio and M. D. Grigoriadis, Approximate Lagrangian decomposition with a modified Karmarkar logarithmic potential, *Network Optimization,*

P. Pardalos, D. W. Hearn and W. W. Hager (eds), Lecture Notes in Economics and Mathematical Systems 450, Springer-Verlag, Berlin, 1997, 471-485.

[110] D. Ye and H. Zhang, The range assignment problem in static ad-hoc networks on metric spaces, *Proceedings of the 11th Colloquium on Structural Information and Communication Complexity*, Sirocco 2004, LNCS 3104, 291-302.

[111] N. E. Young, Randomized rounding without solving the linear program, *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, SODA 1995, 170-178.

[112] N. E. Young, Sequential and parallel algorithms for mixed packing and covering, *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, FOCS 2001, 538-546.

[113] A. Zelikovsky, An 11/6-approximation algorithm for the network Steiner problem, *Algorithmica*, 9 (1993), 463-470.

[114] H. Zhang, Packing: scheduling, embedding and approximating metrics, *Proceedings of the 2004 International Conference on Computational Science and its Applications*, ICCSA 2004, LNCS 3045, 764-775.

[115] H. Zhang, Solving packing problem with weaker block solvers, *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science*, TCS 2004.

Appendix: Fortran Codes of the Numerical Computation

In this appendix we give the source codes in Fortran language to compute the values in Table 4.4, 4.5, 4.6 and 4.7. One can easily change the values of parameters to test our results for other corresponding cases.

The code for Table 4.4 is as follows:

```
PROGRAM MAIN
PARAMETER(MM=33,RHO=0.43)
OPEN(4,FILE='bound_II.dat')
DO 10 M=2, MM
  XMU=0.5*((1.0+2.0*RHO)*M-SQRT((1.0+4.0*RHO*RHO)*M*M-4.0*RHO*M))
  MUL=INT(XMU)
  MUR=MUL+1
  AL=(1.0+(RHO*M-1.0)/((1.0-RHO)*(1.0*M-MUL)+1.0))/RHO
  BNL=(1.0-RHO)*M*(M-2.0*MUL+1.0)+1.0*M*MUL
  BDL=MUL*((1.0-RHO)*(M-MUL)+1.0)
  BL=BNL/BDL
  AR=(1.0+(RHO*M-1.0)/((1.0-RHO)*(1.0*M-MUR)+1.0))/RHO
  BNR=(1.0-RHO)*M*(M-2.0*MUR+1.0)+1.0*M*MUR
  BDR=MUR*((1.0-RHO)*(M-MUR)+1.0)
  BR=BNR/BDR
  OBJL=MAX(AL,BL)
  OBJR=MAX(AR,BR)
  IF(OBJL.LE.OBJR) THEN
    MU=MUL
    R=OBJL
  ELSE
    MU=MUR
```

```
          R=OBJR
        ENDIF
        WRITE(4,100) M, RHO, MU, R
 10   CONTINUE
100   FORMAT(1X, I2, 2X, F10.6, 2X, I2, F12.7)
      CLOSE(4)
      END
```

The code for Table 4.5 is as follows:

```
      PROGRAM MAIN
      PARAMETER(MM=33,NRHO=10000)
      OPEN(4,FILE='numerical_II.dat')
      DRHO=1.0/NRHO
      DO 10 M=1, MM
        MUM=INT(M+1/2)
        R=100000
        DO 20 MU=1,MUM
          DO 30 N=1,NRHO-1
            RHO=N*DRHO
            A=(1.0+(RHO*M-1.0)/((1.0-RHO)*(M-MU)+1.0))/RHO
            BN=(1.0-RHO)*M*(M-2.0*MU+1.0)+1.0*M*MU
            BD=MU*((1.0-RHO)*(M-MU)+1.0)
            B=BN/BD
            OBJ=MAX(A,B)
            IF(OBJ.LT.R) THEN
              RHOOPT=RHO
              MUOPT=MU
              R=OBJ
            ENDIF
 30       CONTINUE
 20     CONTINUE
        WRITE(4,100) M, RHOOPT, MUOPT, R
 10   CONTINUE
100   FORMAT(1X, I2, 2X, F10.6, 2X, I2, F12.7)
      CLOSE(4)
      END
```

The code for Table 4.6 is as follows:

```
      PROGRAM MAIN
      PARAMETER(MM=33,RHO=0.26)
      OPEN(4,FILE='bound_III.dat')
      DO 10 M=1, MM
        XMU=0.5*((2.0+RHO)*M-SQRT((RHO*RHO+2*RHO+2)*M*M-2*(1.0+RHO)*M))
        MUL=INT(XMU)
        MUR=MUL+1
        ANL=(1.0+RHO)*M+(2.0-RHO)*(1.0*M-MUL)
        ADL=(1.0+RHO)*(2.0-RHO)*(M-MUL+1.0)
        AL=2.0*ANL/ADL
        BNL=2.0*M*MUL+(2.0-RHO)*M*(M-2.0*MUL+1.0)
        BDL=MUL*(2.0-RHO)*(M-MUL+1.0)
        BL=BNL/BDL
        ANR=(1.0+RHO)*M+(2.0-RHO)*(1.0*M-MUR)
        ADR=(1.0+RHO)*(2.0-RHO)*(M-MUR+1.0)
        AR=2.0*ANR/ADR
        BNR=2.0*M*MUR+(2.0-RHO)*M*(M-2.0*MUR+1.0)
        BDR=MUR*(2.0-RHO)*(M-MUR+1.0)
        BR=BNR/BDR
        OBJL=MAX(AL,BL)
        OBJR=MAX(AR,BR)
        IF(OBJL.LE.OBJR) THEN
          MU=MUL
          R=OBJL
        ELSE
          MU=MUR
          R=OBJR
        ENDIF
        WRITE(4,100) M, RHO, MU, R
 10   CONTINUE
100   FORMAT(1X, I2, 2X, F10.6, 2X, I2, F12.7)
      CLOSE(4)
      END
```

Finally, the code for Table 4.7 is as follows:

```
      PROGRAM MAIN
      PARAMETER(MM=33,NRHO=10000)
      OPEN(4,FILE='numerical_III.dat')
      DRHO=1.0/NRHO
      DO 10 M=1, MM
        MUM=INT(M+1/2)
        R=100000
        DO 20 MU=1,MUM
          DO 30 N=0,NRHO
            RHO=N*DRHO
            AN=(1.0+RHO)*M+(2.0-RHO)*(M-MU)
            AD=(1.0+RHO)*(2.0-RHO)*(M-MU+1.0)
            A=2.0*AN/AD
            BN=2.0*M*MU+(2.0-RHO)*M*(M-2.0*MU+1.0)
            BD=MU*(2.0-RHO)*(M-MU+1.0)
            B=BN/BD
            OBJ=MAX(A,B)
            IF(OBJ.LT.R) THEN
              RHOOPT=RHO
              MUOPT=MU
              R=OBJ
            ENDIF
 30       CONTINUE
 20     CONTINUE
        WRITE(4,100) M, RHOOPT, MUOPT, R
 10   CONTINUE
100   FORMAT(1X, I2, 2X, F10.6, 2X, I2, F12.7)
      CLOSE(4)
      END
```

Curriculum Vitae

February 2, 1974	Born in Hunan, China
1980 – 1985	student at the Huiyuanjie Primal School, Nanjing, China
1985 – 1986	student at the Xufuxiang Primal School, Nanjing, China
1986 – 1989	student at the Middle School No. 50, Nanjing, China
1989 – 1992	student at the High School Affiliated to the Nanjing Normal University, Nanjing, China
1992 – 1995	bachelor student at the Department of Aerodynamics, the Nanjing University of Aeronautics and Astronautics, Nanjing, China
July 1995	degree of Bachelor of Engineering in Aerodynamics
1995 – 1997	master student at the Department of Aerodynamics, the Nanjing University of Aeronautics and Astronautics, Nanjing, China
March 1998	degree of Master of Engineering in Aerodynamics
1998 – 2000	master student at the Department of Mathematics, the Hong Kong University of Science and Technology, Hong Kong, China
November 2001	degree of Master of Philosophy in Mathematics
Since 2001	PhD student at the Institute of Computer Science and Applied Mathematics, the Christian-Albrechts-University of Kiel, Kiel, Germany