FAIR ALLOCATION OF OPERATIONS AND MAKESPAN MINIMIZATION FOR MULTIPLE ROBOTIC AGENTS

BY

RAUNAK SENGUPTA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Advisers:

Professor Rakesh Nagi
Professor R.S. Sreenivas

Abstract

We study the problem of allocating a set of indivisible operations to a set of agents in a fair and efficient manner while also minimizing the makespan. We first present the Operation Trading Algorithm that generates allocations satisfying the DEQx (Duplicated Equitability up to any operation) fairness criterion while also guaranteeing an upper bound of 2 on the makespan for identical agents. The algorithm also guarantees an upper bound of 1.618 for 2 uniformly related agents and $\frac{1+\sqrt{4n-3}}{2}$ for n uniformly related agents. The pairwise approach used in this algorithm has the added advantages of being decentralizable, reactive and robust. A new protocol named as the Decentralized Random Group Formation (DRGF) Protocol is presented for implementing the Operation Trading Algorithm in a decentralized manner and for dealing with communication failures. We then define a relaxed version of the DEQ1 (Duplicated Equitability upto some operation) fairness criterion called partial-DEQ1. A market-based algorithm is presented to achieve partial-DEQ1 along with Pareto Optimality. Following this, it is shown that the algorithm also guarantees an upper bound of 1.618 on the makespan for 2 non-identical agents. Parametric pruning further improves the upper bound to 1.5, which is theoretically the best possible upper bound. To the best of our knowledge, these are the first algorithms designed to achieve the mentioned fairness criteria. The algorithms additionally guarantee upper bounds on the makespan. Finally, we show the efficacy of the algorithms in generating allocations with near optimal makespans by numerically evaluating the algorithms on randomly generated problem instances.

To my parents, for their love and support.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisors Prof. Rakesh Nagi and Prof. Ramavarapu S. Sreenivas for their continuous support, guidance, patience, and immense knowledge. Through their guidance, I was able to shape my thoughts and ideas and mature as a researcher.

I would like to thank Dr. Dustin Nottage and Dr. Ahmet Soylemezoglu from Construction Engineering Research Laboratory (CERL), U.S. Army for their support and also for providing me exposure to some of the cutting edge research in Autonomous Construction.

I would like to thank my friends Ayush Rajput, Anunay Sharma, Vivek Gupta, Anirudh Sharma, Akhil Gupta and Avirup Kundu for being by my side at all times. Their immense help, support and friendship cannot be expressed in words.

Most importantly, I would like to express my gratitude to my mom, Malancha Gupta and dad, Aniruddha Sengupta for their endless and unconditional love and support, and for instilling in me the right values and mindset.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are several applications that require a team of autonomous agents to intelligently cooperate, collectively make decisions and performs tasks. Parallel machine scheduling problems arise naturally in such systems and are thus an integral part of multi-agent systems. Such systems are also very often set in an unpredictable and hostile environment. An example would be a team of self-directed agents that execute construction tasks autonomously. Applications such as this demand scheduling algorithms that are preferably decentralizable, simple to perform, and do not require large communication overhead. Different objective functions have been considered for parallel machine scheduling in the literature such as: linear sum of costs, weighted completion time, makespan, etc.

The makespan is an important objective function, especially for time-critical applications since it is a direct measure of the overall completion time of a project. In many situations, a near optimal value of the makespan also implies the same work accomplished in a shorter amount of time, and thus, higher production efficiencies. Traditional methods to solve the makespan minimization problem for non-identical parallel machines primarily include centralized Linear Programming (LP) based methods [1, 2, 3]. The current state of the art in "centralized" algorithms guarantees an upper bound of $2-(1/n)$ where n is the number of agents [4]. This algorithm is an improvement over the algorithm due to [1], which is an LP-based method. Interestingly, majority of the results in the current literature on decentralized task scheduling for multi-agent systems do not consider the makespan objective.

It can be observed that if an allocation has loads that are as balanced as possible and there exists no other allocation that is strictly better (i.e., the allocation is Pareto Optimal), the corresponding makespan must be at the minimum. Thus, an alternative approach to tackle the problem of makespan minimization is to set up the problem in the fair allocation paradigm. The literature on fair allocation has recently seen a rise in algorithms that generate "approximately equitable" and Pareto Optimal allocations of indivisible goods among multiple non-identical agents. An allocation is said to be equitable if every agent derives equal utilities under the given allocation. Recently, algorithms to generate equitable allocations of chores were proposed in [5], where a chore is defined as an item with negative valuation. Tasks/operations in a multi-agent setting can be modelled in a similar way.

1.1 Motivation

Equitable allocation of operations is of importance in settings where the agents are either humans or robots obtained from different contractors. To satisfy every entity in such a scenario, equitable allocations are necessary. The allocation must however satisfy a global objective function at the same time, which in our case is the maximum completion time among all jobs (makespan).

In this thesis, we explore the possibility of generating allocations that satisfy certain equitability criteria and at the same time obtain an upper bound on the makespan by taking advantage of the similarities in both the problems. This approach also offers a new way of looking at the makespan minimization problem in an agent based manner which we believe will enable the design of efficient decentralized and robust algorithms.

1.2 Literature Review

The existence of a polynomial time algorithm with an upper bound lesser than 1.5 on the makespan would imply that $P = NP$ [1]. Reference [1] introduced a polynomial time LP-based algorithm that guarantees an upper bound of 2. This was further improved upon by reference [4] which presented an algorithm that guaranteed an upper bound of $2 - \frac{1}{n}$ where n is the number of agents. The existence of an algorithm that obtains a strict upper bound of 1.5 remains open. There are algorithms with better upper bounds for special cases of the problem such as the graph balancing problem [6]. The literature has efficient algorithms for makespan minimization of identical agents, such as the LPT rule [7] that guarantees an upper bound of $\frac{4}{3} - \frac{1}{3n}$. The Graham's list scheduling algorithm [7] guarantees an upper bound of $2 - \frac{1}{n}$ for identical agents with/without precedence constraints. Reference [8] showed that the classic LPT rule guarantees an approximation factor of $\frac{2n}{2n+1}$ for uniformly related agents. [9] analyzed a variant of the MULTIFIT algorithm and showed that it guarantees an approximation factor of 1.4, which is currently the best known approximation factor for uniformly related agents using a centralized approach.

Majority of the literature on Decentralized Task Scheduling for multi-agent systems use a linear sum of costs as the objective function. A large collection of results use market-based algorithms such as auctions for task allocation [10, 11, 12]. However, the auction algorithm can lead to arbitrarily poor solutions in a number of situations [13]. A different approach to task allocation is to use consensus algorithms [14, 15]. The thesis [16] introduced the algorithm Consensus Based Bundle Allocation (CBBA) which can be implemented in a decentralized manner and gives an upper bound of 2 for the linear sum of costs objective function. CBBA combines the advantages of both the consensus and auction algorithms.

References [17, 18] introduce decentralized gossip-based algorithms which minimize the makespan for a networked system of agents. These algorithms provide guarantees on the convergence time of the algorithm. Further, [18] provides an upper bound on the makespan which is a function of the largest operating time. These gossip based algorithms are however not defined for unrelated/non-identical parallel machines. [19, 20, 21] are metaheuristic approaches for minimizing the makespan. Metahueristics however do not accompany theoretically-grounded performance bounds. To the best of our knowledge, there do not exist decentralized algorithms for makespan minimization that guarantee a constant factor approximation on the optimal makespan.

Fair allocation of indivisible goods has received considerable interest in the recent years. Several notions of fairness have been defined in the literature such as Maximin Fair Share (MMS) [22, 23], Envy Freeness (EF) [24, 25], Equitability (EQ) [26], etc. In this thesis, we deal with the equitability fairness criterion. An allocation is said to be equitable if every agent derives equal utilities under the given allocation. When the goods are divisible, perfectly equitable allocations are guaranteed to exist [26]. However, for indivisible goods, such an allocation may not always exist. Reference [27] introduced relaxed versions of equitability with stronger existence results. These relaxed versions of EQ were formalized into the terms EQ1 and EQx in reference [28]. An allocation A is equitable up to one good (EQ1) if for every pair of agents, the disparity can be reversed by removing *some* good from the higher-valuation-agent's bundle. An allocation A is equitable up to any good (EQx) if for every pair of agents, the disparity can be reversed by removing *any* good from the higher-valuation-agent's bundle. Reference [28] presented algorithms to generate allocations that are EQ1 + Pareto Optimal, EQx, etc. Thesis [5] extended the definitions of EQ1 and EQx to the context of chores (goods with negative valuations). They showed that for chores, an EQx + Pareto Optimal allocation may not always exist. Following this, they presented the fairness notions DEQ1 and DEQx. An allocation A is said to be equitable upto one duplicated chore (DEQ1) if for every pair of agents, the disparity can be reversed by duplicating *some* chore from the lower-valuation-agent's bundle to the higher-valuation-agent's bundle. An allocation A is equitable upto any duplicated chore if the disparity can be reversed by duplicating *any* chore from the lower-valuation-agent's bundle to the higher-valuation-agent's bundle. Algorithms to generate DEQx and DEQ1 + Pareto Optimal allocations were left as open problems.

1.3 Our Contributions

The main contributions of this thesis are listed as follows:

- We present the Operation Trading Algorithm (OTA) which is guaranteed to generate a DEQx allocation. Although the existence of a DEQx allocation at all times was proven in [5], no systematic algorithm was shown to generate such allocations prior to this.
- The proposed algorithm guarantees an approximation factor less than 1.618 for 2 uniformly related agents and an approximation factor less than $\frac{1+\sqrt{4n-3}}{2}$ for n uniformly related agents, assuming a fully connected communication network.
- The algorithm also guarantees an approximation factor less than 2 for n identical agents assuming fully connected communication network.

- The DRGF Protocol is presented for implementing OTA(n) in a decentralized manner. The upper bounds do not hold if the agents are not fully connected. However, it has been shown that the algorithm generates near optimal allocations for most cases even under severe communication link failures (assuming that the agents cannot act as relays).

- It has also been shown that OTA(n) is fast and can start from any random allocation. Thus, the algorithm can be easily used for quick re-planning in case of various failures and changes in the environment.

- In OTA(n), an agent communicates only its personal allocation at a time to its neighbors instead of costs associated to all the operations in the mission like in CBBA [16] and other similar algorithms. This reduces the communication overhead significantly.

- This thesis presents a new equitability measure which we call partial-DEQ1 (p-DEQ1) which is a relaxed version of DEQ1 (Defined in Section 2). A market-based algorithm (referred to as MBA) similar to those presented in [28] [5] is proposed to generate allocations that are p-DEQ1 and Pareto optimal.

- It is then proven that for 2 non-identical agents, the market-based algorithm guarantees an upper bound of $\frac{\sqrt{5}+1}{2} \approx 1.618$ on the makespan.

- We then present how Parametric Pruning as in [1] can be combined with the presented algorithms to improve the performance on the makespan. It is proven that for 2 non-identical agents, the market-based algorithm combined with Parametric Pruning guarantees an upper bound of 1.5 on the makespan (this is the best possible upper bound that can be obtained by any polynomial time algorithm).

- We finally show through numerical results that the presented algorithms almost always generate allocations with a near-optimal makespan.

- It must be noted that unlike a number of heuristics in the literature, the presented algorithms do not have any hyper-parameters that need to be tuned.

The obtained results are also presented in Tables 1.1, 1.2 and 1.2.

Table 1.1: Upper Bounds and Fairness for different algorithms

	Identical Agents			
	2 Agents		n Agents	
	UB	Fairness	UB	Fairness
OTA(n)	1.33*	DEQx	$2 - \frac{2}{n+1}$*	DEQx
OTA(n) + Pr	1.33*	DEQx	$2 - \frac{2}{n+1}$*	DEQx
MBA	1.5	DEQ1	$2 - \frac{2}{n+1}$	p-DEQ1
MBA + Pr	1.5	DEQ1	$2 - \frac{2}{n+1}$	p-DEQ1
pairwise-MBA	1.5	DEQ1	-	DEQ1
pairwise-MBA + Pr	1.5	DEQ1	-	DEQ1

Pr: Pruning; UB: upper bound; *: Tight Bound; - : Not Known

Table 1.2: Upper Bounds and Fairness for different algorithms

	Uniformly Related Agents			
	2 Agents		n Agents	
	UB	Fairness	UB	Fairness
OTA(n)	1.619	DEQx	$\frac{1+\sqrt{4n-3}}{2}$	DEQx
OTA(n) + Pr	1.619	DEQx	$\frac{1+\sqrt{4n-3}}{2}$	DEQx
MBA	1.619*	DEQ1+PO	-	p-DEQ1+PO
MBA + Pr	1.5*	DEQ1 + PO	-	p-DEQ1+PO
pairwise-MBA	1.619*	DEQ1	-	DEQ1
pairwise-MBA + Pr	1.5*	DEQ1+PO	-	DEQ1+PO

Pr: Pruning; UB: upper bound; *: Tight Bound; - : Not Known

1.4 Organization of the thesis

The remainder of the thesis is structured as follows. Chapter 2 presents the preliminaries and formulations pertaining to the makespan minimization problem and the fair allocation problem. Chapter 3 presents the Operation Trading Algorithm for generating DEQx allocations along with all the necessary proofs. Chapter 4 presents the DRGF protocol for decentralization. Chapter 5 presents a Market Based Algorithm for generating partial-DEQ1 allocations along with all the necessary proofs and upper bounds on makespan. Following this, Chapter 6 shows how Parametric Pruning can be used to improve the performances of the algorithms. Chapter 7 presents all the numerical results and finally Chapter 8 presents the conclusions.

Table 1.3: Upper Bounds and Fairness for different algorithms

	Non-Identical Agents			
	2 Agents		n Agents	
	UB	Fairness	UB	Fairness
OTA(n)	-	DEQx	-	DEQx
OTA(n) + Pr	-	DEQx	-	DEQx
MBA	1.619*	DEQ1+PO	-	p-DEQ1+PO
MBA + Pr	1.5*	DEQ1 + PO	-	p-DEQ1+PO
pairwise-MBA	1.619*	DEQ1	-	DEQ1
pairwise-MBA + Pr	1.5*	DEQ1+PO	-	DEQ1+PO

Pr: Pruning; UB: upper bound; *: Tight Bound; - : Not Known

Chapter 2

Preliminaries

2.1 The Makespan Minimization Problem

The makespan minimization problem is a well-studied problem in the scheduling literature. Makespan is defined as the maximum time required to complete all the operations. This problem has numerous variations and assumptions. In this thesis we assume that there are n agents and m operations. In the most general case, the agents may have different processing times for the same operation. In such a case, the agents are said to be unrelated or non-identical. Let the processing time for operation j corresponding to agent i be $v_{i,j}$. Then, the makespan minimization problem for non-identical agents is captured by the following Integer Program:

$$min \;\; Z \tag{2.1}$$

$$\sum_{j=1}^{m} x_{i,j} v_{i,j} \leq Z \qquad \forall i \in \{1, 2, ..., n\}, \tag{2.2}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad \forall j \in \{1, 2, ..., m\}, \tag{2.3}$$

$$x_{i,j} \in \{0, 1\} \qquad i \in \{1, 2, ..., n\}, j \in \{1, 2, ..., m\}. \tag{2.4}$$

In this integer program, $x_{i,j} = 1$ if operation j is assigned to agent i and 0 otherwise. Z represents the makespan. As mentioned earlier, the makespan minimization problem tries to minimize the maximum completion time among the agents. When there is no precedence order among operations and all operations are available initially, this is equivalent to minimizing the maximal workload on an agent.

2.2 Makespan Minimization for Uniformly Related Agents

A special case of the makespan minimization problem that appears very often in practice is the case where the agents are uniformly related, i.e., the ratio of operating times between any two agents is a constant independent of the operation. This special case models agents with different speeds. Let the processing time for operation j by the slowest agent be t_j and the speed of agent i be f_i. We assume without loss of generality that the speeds of the agents are normalized with respect to the slowest agent (i.e. the speed of the slowest agent has a value of 1). Therefore, the time required by agent i to complete operation j is t_j/f_i. The makespan minimization problem for this setting is captured by the following Integer Program:

$$min \;\; Z \tag{2.5}$$

$$\sum_{j=1}^{m} x_{i,j}\frac{t_j}{f_i} \leq Z \qquad \forall i \in \{1,2,...,n\}, \tag{2.6}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad \forall j \in \{1,2,...,m\}, \tag{2.7}$$

$$x_{i,j} \in \{0,1\} \qquad i \in \{1,2,...,n\}, j \in \{1,2,...,m\}. \tag{2.8}$$

In this formulation, $x_{i,j} = 1$ if operation j is assigned to agent i and 0 otherwise. Z represents the makespan.

2.3 Equitability and Fairness Concepts

Let $[k]$ denote the set $\{1,2,...,k\}$. An instance of fair division of operations is given by $\langle [n],[m],V\rangle$ where $[n]$ denotes the set of agents, $[m]$ denotes the set of operations with strictly positive valuations and $V = \{v_1, v_2, ..., v_n\}$ are the valuation profiles for each of the agents. $v_{i,j}$ denotes the valuation (processing time) for operation j corresponding to agent i. An allocation A is a partition of $[m]$ into n disjoint subsets, $(A_1, A_2, ..., A_n)$, where A_i is the bundle given to agent i. $v_i(A_j)$ denotes how much agent i values bundle j. That is, $v_i(A_j)$ is the completion time of agent i if it were to process operations in agent

j's bundle. We invoke the concept of equitability from the Fair Allocation literature to equalize the completion times of agents and, in turn, minimize overall makespan. A formal proof of this equivalence is shown in Chapter 3.

Definition 1. *An allocation A is called equitable (EQ) if for all pairs of agents i, $k \in [n]$, we have $v_i(A_i) = v_j(A_j)$.*

However, an equitable allocation may not always exist. Thus, relaxed equitability criteria have been defined in the literature. Most of the research in fair allocation aims at proving the existence of allocations that satisfy the relaxed fairness criteria and subsequently generating allocations that achieve the relaxed fairness criteria through an efficient algorithm.

The concepts of DEQ1 and DEQx were presented in [5]. This reference considers a scenario where each agent has a negative valuation for each chore/operation, and each agent maximizes its personal total valuation.

In this thesis, we follow a different perspective. We consider a scenario where every agent has a strictly positive valuation for each of the operations and every agent tries to minimize its personal total valuation. The total valuation of an agent here represents the agent's "dis-utility." This is equivalent to trying to maximize the valuations if they were negative. This perspective also ties in with the makespan minimization problem as we shall see later. Hence, we provide the following alternative but equivalent definitions of DEQ1 and DEQx:

Definition 2. *An allocation A is equitable up to one duplicated operation (DEQ1) if for every pair of agents $i, k \in [n]$ such that $A_i \neq \phi$, there exists some operation $j \in A_i$ such that $v_i(A_i) \leq v_k(A_k \cup j)$.*

Definition 3. *An allocation A is equitable up to any duplicated operation (DEQx) if for every pair of agents $i, k \in [n]$ such that $A_i \neq \phi$ and for every operation $j \in A_i$, we have $v_i(A_i) \leq v_k(A_k \cup j)$.*

Additionally, we propose a new concept for fair allocation which we call partial-DEQ1 (p-DEQ1) which is defined as follows:

Definition 4. *An allocation A is partially equitable up to one duplicated operation (p-DEQ1) if for every agent $k \in [n]$, there exists an operation $j \notin A_k$ such that $max_i\{v_i(A_i)\} \leq v_k(A_k \cup j)$.*

Along with equitability, Pareto optimality is an equally important concept in the fair allocation literature. Pareto optimality is defined as follows:

Definition 5. *An allocation A is said to be Pareto dominated by another allocation B if $v_k(B_k) \leq v_k(A_k)$ for all agents $k \in [n]$ with at least one of the inequalities being strict. An allocation A is said to be Pareto Optimal (PO) if there exists no other allocation that Pareto dominates the allocation A. This Pareto Optimal front corresponds to the situation where every agent looks to minimize its personal valuation.*

Chapter 3

Generating DEQx Allocations: The Operation Trading algorithm

3.1 The Operation Trading Algorithm

Using the definitions provided in Section 2, we define the Minimax potential function as $\Phi(A_i, A_j) = max\{v_i(A_i), v_j(A_j)\}$. This is essentially the makespan (for two agents i and j). We now present a simple algorithm which we call the Operation Trading Algorithm (n) or OTA(n) (Algorithm 2) for n agents.

The algorithm starts with an initial allocation, or a random one if none exists. It then iterates through every possible pair of agents looking for improvements. For any pair of agents i and j, we obtain allocations A_i^{new} and A_j^{new} respectively by applying OTA(2) on i and j. Let the allocations for agents i and j before applying OTA(2) be A_i^{old} and A_j^{old}, respectively. If the new Minimax Potential function, $\Phi(A_i^{new}, A_j^{new})$ is lesser than $\Phi(A_i^{old}, A_j^{old})$, the old allocations get replaced by the new allocations. The algorithm terminates when there exists no pair of agents whose makepsan (Potential function) can be further improved. OTA(n) calls the algorithm OTA(2) repeatedly, so we now discuss OTA(2).

Consider any two agents i and j. In every iteration of the while loop, OTA(2) (Algorithm 1) finds an operation which if transferred from one agent to another leads to the maximum decrease of the potential function, Φ; i.e., a greedy approach is used. The algorithm converges once there exists no operation transfer that leads to a decrease in Φ. At any iteration, if $v_i(A_i) < v_j(A_j)$, the algorithm looks for operations to transfer from A_j to A_i and vice-versa if $v_i(A_i) > v_j(A_j)$.

In OTA(n), for every pair of agents OTA(2) is invoked, and the corresponding Minimax Potential function either improves or stays the same. Thus, the total valuation of the agent with the highest load cannot increase. Once, the

Algorithm 1: OTA(2)

```
Initialize allocation (perhaps randomly) $(A_i, A_j)$
flag = 0
$\Phi(A_i, A_j) = max(v_i(A_i), v_j(A_j))$
while flag == 0 do
    $P = \Phi(A_i, A_j)$
    if $v_i(A_i) < v_j(A_j)$ then
        $k = \underset{c \in A_j}{arg\,max}(P - \Phi(A_i \cup \{c\}, A_j \backslash \{c\}))$
        $\Delta P = (P - \Phi(A_i \cup \{k\}, A_j \backslash \{k\}))$
        if $\Delta P > 0$ then
            Transfer operation $k$ from $A_j$ to $A_i$
        else
            flag = 1
        end if
    else
        $k = \underset{c \in A_i}{arg\,max}(P - \Phi(A_i \backslash \{c\}, A_j \cup \{c\}))$
        $\Delta P = (P - \Phi(A_i \backslash \{k\}, A_j \cup \{k\}))$
        if $\Delta P > 0$ then
            Transfer operation $k$ from $A_i$ to $A_j$
        else
            flag = 1
        end if
    end if
end while
```

valuation of the agent with the highest load ceases to decrease, the valuation of the agent with the second highest load starts decreasing till it cannot decrease any more. Following this, the valuation of the agent with the third highest load starts decreasing and so on. Due to the discrete nature of the problem, this cannot continue indefinitely. Thus, the algorithm must always converge. This algorithm can be seen as pairs of agents trading operations between themselves until the potential value cannot decrease further, and hence the name Operation Trading Algorithm.

Algorithm 2: OTA(n)

```
1: imp_flag = 1; Begin with allocations $(A_1, A_2, ..., A_n)$
2: while imp_flag == 1 do
3:   imp_flag = 0
4:   for i = 0 : N do
5:     for j = 0 : N do
6:       $P^{old} \leftarrow \Phi(A_i, A_j)$
7:       Apply OTA(2) on $i$ and $j$
8:       $A_i^{new} \leftarrow$ Alloc. obtained by OTA(2) for $i$
9:       $A_j^{new} \leftarrow$ Alloc. obtained by OTA(2) for $j$
10:      $P^{new} \leftarrow \Phi(A_i^{new}, A_j^{new})$
11:      if $P^{new} < P^{old}$ then
12:        imp_flag = 1
13:        $A_i \leftarrow A_i^{new}$
14:        $A_j \leftarrow A_j^{new}$
15:      end if
16:    end for
17:  end for
18: end while
```

Theorem 1. *OTA(n) always generates an allocation that is DEQx even for general valuation functions.*

Proof. Once OTA(n) converges, no pair of agents exists whose corresponding Minimax Potential function can be improved. Let us consider any pair of agents i and j.

The valuations for both the agents may or may not be equal. In the case that the valuations are equal, the allocation is already DEQx for the pair of agents i and j. In the case that they are not equal, without loss of generality we assume that $v_i(A_i) < v_j(A_j)$. Since the state has converged, transfer of any operation c from A_j to A_i must lead to an increase in the potential function $\Phi(A_i, A_j)$. This gives us the following:

$$max\{v_i(A_i \cup c), v_j(A_j \backslash c)\} > max\{v_i(A_i), v_j(A_j)\}, \forall c \in A_2$$

$$\implies v_i(A_i \cup c) > v_j(A_j) \qquad \forall c \in A_2. \tag{3.1}$$

By definition, Equation (3.1) directly implies a DEQx allocation for the pair i and j. Since Equation (3.1) holds for every pair of agents, OTA(n) always leads to a DEQx allocation. Note that the proof does not make any assumption about the valuation functions except that they are monotonically increasing in addition of operations. Thus, the algorithm generates a DEQx allocation even for general valuation functions. □

Theorem 2. *If the agents are uniformly related, all the possible allocations are Pareto Optimal, i.e., a decrease in the completion time of some agent is always accompanied by an increase in the completion time of some other agent.*

Proof. Let us assume that the statement is not true. Then, there must exist some allocation B that strictly Pareto dominates A. Thus, $v_i(B_i) \leq v_i(A_i) \forall i \in [n]$ and $v_i(B_i) < v_i(A_i)$ for at least one $i \in [n]$. Thus, we have:

$$\sum_{i=1}^{n} f_i v_i(B_i) < \sum_{i=1}^{n} f_i v_i(A_i). \tag{3.2}$$

However, we know that

$$\sum_{i=1}^{n} f_i v_i(B_i) = \sum_{i=1}^{n} f_i v_i(A_i) = \sum_{j=1}^{m} t_j, \tag{3.3}$$

where t_j is the operating time of operation o_j if the speed of agent was 1. This is in contradiction to Equation (3.2). Thus, our initial assumption was wrong and there exists no allocation B that Pareto dominates A.

□

3.2 Upper Bound of OTA(n) for Identical Agents

Theorem 3. *OTA(n) for n identical agents always leads to an allocation with makespan not more than* $\frac{\kappa n}{((\kappa-1)n+1)}z^*$ *where* κ *is the number of operations allocated to the agent with the highest valuation and* z^* *is the optimal makespan. Further, this upper bound is tight for a given value of* κ *and* n*.*

Proof. Let A be the allocation obtained from OTA(n). Without loss of generality, we assume agent 1 to be the agent with the highest completion time. Thus, $v_1(A_1)$ is also the value of makespan. Let the optimal makespan value be z^*.

Since, the agents are identical, we have:

$$\begin{aligned} z^* &\geq \frac{\sum_{i=1}^{n} v_i(A_i)}{n} && (3.4)\\ \implies v_1(A_1) &\leq z^* + \frac{\sum_{i=2}^{n}(v_1(A_1) - v_i(A_i))}{n} \\ \implies \frac{v_1(A_1)}{z^*} &\leq 1 + \frac{\sum_{i=2}^{n}(v_1(A_1) - v_i(A_i))}{nz^*} \\ &\leq 1 + \frac{\sum_{i=2}^{n}(v_1(A_1) - v_i(A_i))}{\sum_{i=1}^{n} v_i(A_i)}. && (3.5)\end{aligned}$$

Now, since the allocation is DEQx, we have that:

$$\begin{aligned} v_1(A_1) &\leq v_j(A_j \cup g) \quad \forall g \in A_1, \forall j \in [n]\backslash 1 \\ &\leq v_j(A_j) + v_{j,g} \quad \forall g \in A_1, \forall j \in [n]\backslash 1 \\ &\leq v_j(A_j) + \underset{g}{min}\{v_{jg}\} \quad \forall j \in [n]\backslash 1 \\ &\leq v_j(A_j) + \frac{v_1(A_1)}{\kappa} \quad \forall j \in [n]\backslash 1, && (3.6)\end{aligned}$$

where κ is the number of operations allocated to agent 1. It can be observed that $\underset{g}{min}\{v_{jg}\} \leq \frac{v_1(A_1)}{\kappa}$ and hence we obtain the inequality (3.6). Combining inequalities (3.5) and (3.6), we get

$$\frac{v_1(A_1)}{z^*} \leq \frac{\kappa n}{((\kappa-1)n+1)}. \quad (3.7)$$

Figure 3.1 presents an instance where this bound is tight. Here we have agent 1 with κ operations each of length of a and $n-1$ agents each with a large number of infinitesimally small operations such that the total length is

$(\kappa - 1)a$. It can be observed that the allocation is DEQx and thus a valid output of OTA(n).

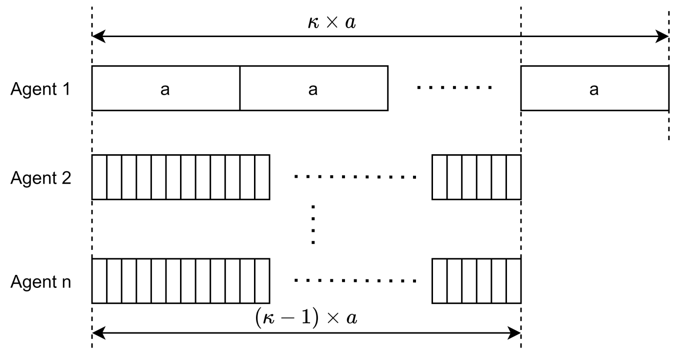


Figure 3.1: Example illustrating UB in Theorem 3 is tight

□

Corollary 3.1. *The upper bound (UB) on the makespan of n identical agents induced by OTA(n) is* $2 - \frac{2}{n+1}$.

Proof. Let agent 1 be the agent with the highest completion time. From Theorem 3, we know that the UB is given by $\frac{\kappa n}{((\kappa-1)n+1)}$ where κ is the number of operations allocated to agent 1. $\kappa = 1$ implies that agent 1 has only one operation. Since, the completion time of agent 1 is the highest, the completion time of the operation belonging to agent 1 is also the highest. Thus, the obtained makespan is an optimal one. Therefore, we only need to analyze cases for $\kappa \geq 2$.

$$\implies \frac{\kappa n}{((\kappa-1)n+1)} \leq \frac{2n}{n+1}$$
$$\implies \frac{\kappa n}{((\kappa-1)n+1)} \leq 2 - \frac{2}{n+1} = UB. \tag{3.8}$$

□

The upper bound in Corollary 3.1 occurs in the worst case when κ is not guaranteed to be greater than 2. However, there may be several applications where the lower bound on κ is higher. For example, if the sum of the 3 largest operations is less than the average load for identical agents, $\kappa \geq 4$ and the admitted upper bound is 4/3. As the lower bound on κ increases, the upper bound admitted by OTA(n) improves as well.

It must be observed that there are other algorithms for makespan minimization on identical agents such as the Graham's List Scheduling algorithm which admits an upper bound of $2 - \frac{1}{n}$ [7] and the Longest Processing Time rule [7] which has an upper bound of $\frac{4}{3} - \frac{1}{3n}$. However, these algorithms are not defined for non-identical agents. On the contrary, OTA(n) is designed to generate a DEQx allocation for non-identical agents with general valuation functions. Additionally, the algorithm admits an upper bound on the makespan for identical agents as well.

3.3 Upper Bound of OTA(n) for Uniformly Related Agents

Theorem 4. *OTA(n) generates an allocation with approximation factor $\frac{1+\sqrt{4n-3}}{2}$, where n is the number of uniformly related agents.*

Proof. Without loss of generality, we assume that agent 1 has the highest completion time. By design of OTA(n), for every agent i we have:

$$\begin{aligned} v_i(A_i) + \frac{f_1}{f_i} t_1^{min} &\geq v_1(A_1) \\ \implies f_i v_i(A_i) + f_1 t_1^{min} &\geq f_i v_1(A_1), \end{aligned} \tag{3.9}$$

where t_1^{min} is the smallest operation belonging to agent 1. The optimal makespan must be greater than or equal to the time taken by the fastest agent to complete the smallest operation in agent 1. This gives us:

$$z^* \geq \frac{f_1 t_1^{min}}{f_u}, \tag{3.10}$$

where u is the index of the fastest agent. The optimal makespan must also be larger that the weighted average of all the completion times. This gives us:

$$z^* \geq \frac{\sum_{i=1}^{n} f_i v_i(A_i)}{\sum_{i=1}^{n} f_i}. \tag{3.11}$$

By combining Equations (3.9) and (3.10), we get:

$$\begin{aligned} f_i v_i(A_i) + f_u z^* &\geq f_i v_1(A_1) \qquad \forall i \in 2, 3, ..., n \\ \implies \sum_{i=1}^{n} f_i v_i(A_i) + (n-1) f_u z^* &\geq v_1(A_1) \sum_{i=1}^{n} f_i. \end{aligned} \tag{3.12}$$

By combining Equations (3.11) and (3.12), we get:

$$\frac{v_1(A_1)}{z^*} \leq 1 + \frac{(n-1) f_u}{\sum_{i=1}^{n} f_i}. \tag{3.13}$$

Since Equation (3.9) holds for agent u as well, we have:

$$\begin{aligned} f_u v_u(A_u) + f_1 t_1^{min} &\geq f_u v_1(A_1) \\ \implies \sum_{i=1}^{n} f_i v_i(A_i) &\geq f_u v_1(A_1) \\ \implies z^* \sum_{i=1}^{n} &\geq f_u v_1(A_1) \\ \implies \frac{v_1(A_1)}{z^*} &\leq \frac{\sum_{i=1}^{n} f_i}{f_u}. \end{aligned} \tag{3.14}$$

From Equations (3.13) and (3.14), we get:

$$\frac{v_1(A_1)}{z^*} \leq \max\left(\frac{\sum_{i=1}^{n} f_i}{f_u}, 1 + \frac{(n-1) f_u}{\sum_{i=1}^{n} f_i}\right). \tag{3.15}$$

The approximation factor, $\frac{v_1(A_1)}{z^*}$ is maximum when the 2 arguments in the max function are equal. Let $y = \frac{\sum_{i=1}^{n} f_i}{f_u}$. Then, we get $y = 1 + \frac{n-1}{y}$. Solving this for y gives us $y = \frac{1+\sqrt{4n-3}}{2}$. Thus, the approximation factor is given by $\frac{1+\sqrt{4n-3}}{2}$. □

Corollary 4.1. *The makespan of the allocation obtained using OTA(2) does not exceed* $1.618 z^*$ *where* z^* *is the optimal makespan.*

Proof. If we put $n = 2$ in the approximation factor for OTA(n), we get $\frac{1+\sqrt{5}}{2} = 1.618$. □

We assume without loss of generality that the agent 1 has the highest completion time.

Corollary 4.2. *If in the allocation A obtained by OTA(n), all the completion times of agents 2, 3, ..., n are either greater than $\frac{v_1(A_1)}{2}$ or less than $\frac{v_1(A_1)}{2}$ simultaneously, the makespan $v_1(A_1) \leq 2z^*$, where z^* is the optimal makespan.*

Proof. We have the following 2 cases:

- **Case 1**: $v_j(A_j) \leq \frac{v_1(A_1)}{2} \quad \forall j \in [n]\backslash 1$.

  Since A is obtained using the OTA(n) algorithm, we have:

$$\begin{aligned} v_1(A_1) &\leq v_j(A_j \cup g) \quad \forall g \in A_1, \forall j \in [n]\backslash 1 \\ &\leq v_j(A_j) + \frac{f_i t_g}{f_j} \quad \forall g \in A_i, \forall j \in [n]\backslash 1 \\ \implies \frac{v_i(A_i)}{2} &\leq \frac{f_i t_{min}}{f_j} \quad \forall g \in A_i, \forall j \in [n]\backslash 1. \end{aligned}$$

  Since, at least one operation in agent 1 must be misplaced for the allocation to not be optimal, $\frac{f_i t_{min}}{f_j}$ becomes a lower bound on the optimal makespan z^*. Thus, $z^* \geq \frac{v_i(A_i)}{2}$. This implies that $v_1(A_1) \leq 2z^*$.

- **Case 2**: $v_j(A_j) \geq \frac{v_1(A_1)}{2} \quad \forall j \in [n]\backslash 1$.

  In this case, all completion times are larger than $\frac{v_1(A_1)}{2}$. Since all possible allocations are Pareto Optimal for uniformly related agents, there exists no allocation whose makespan is less than $\frac{v_1(A_1)}{2}$. Thus, we have $v_1(A_1) \leq 2z^*$.

□

Corollary 4.3. *If in the allocation A generated using OTA(n), the agent with the largest completion time is also the slowest agent and has at least 2 operations allocated to it, the upper bound on the makespan cannot exceed 2.*

Proof. Without loss of generality, we assume that agent 1 has the highest completion time. Since, the allocation is being generated using OTA(n), we have:

$$\begin{aligned} v_i(A_i) + \frac{f_1}{f_i} t_{min} &\geq v_1(A_1) \\ \implies v_i(A_i) + \frac{f_1}{\kappa f_i} &\geq v_1(A_1). \end{aligned} \tag{3.16}$$

where κ is the number of operations in agent 1 under allocation A.

A valid lower bound on the optimal makespan z^* is given by:

$$z^* \geq \frac{\sum_{i=1}^{n} f_i z_i}{\sum_{i=1}^{n} f_i} \tag{3.17}$$

$$\implies UB \leq \frac{1 + \frac{f_2}{f_1} + ... + \frac{f_n}{f_i}}{1 + \frac{f_2}{f_1} + \frac{f_3}{f_1} + ... + \frac{f_n}{f_1} - \frac{(n-1)}{\kappa}}. \tag{3.18}$$

The last inequality is obtained by combining (3.17) with (3.16). Now, since agent 1 is the slowest agent, we have $\frac{f_i}{f_1} \geq 1 \forall i \in [n]$. From this, we get:

$$UB \leq \frac{2n}{2n+1} < 2. \tag{3.19}$$

□

3.4 Upper Bounds for Practical Scenarios with Uniformly Related Agents

In most practical scenarios, the ratio between the speeds of any two agents is lower bounded by a small finite value. Thus, the ratios $\frac{f_i}{f_1}$ are lower bounded, where f_1 is the speed of agent 1 i.e., the agent with highest completion time. Note that agent 1 need not be either the fastest or the slowest agent. Further, the number of operations belonging to the agent with the highest completion time, κ also has a lower bound in most practical cases. We can use these bounds that occur in practical scenarios to analyze the worst case performance in said practical scenarios and also get an intuition behind the working of the algorithm. Below, we present the Upper Bounds for different values of lower bounds on κ and $\frac{f_i}{f_1}$ obtained from Equation (3.18).

The variable κ will be guaranteed to have a lower bound K in a setting if the sum of the operating times of K slowest operations on the slowest processor is less than the value of z^* given by Equation (3.17).

Table 3.1: Upper Bounds on the makespan for different lower bounds on κ and $\frac{f_i}{f_1}$.

Lower Bound on κ	$\frac{f_i}{f_1} \geq 1$	$\frac{f_i}{f_1} \geq \frac{3}{4}$	$\frac{f_i}{f_1} \geq \frac{2}{3}$	$\frac{f_i}{f_1} \geq \frac{1}{2}$
1	n	-	-	-
2	2.0	3.0	4.0	-
3	1.5	1.8	2.0	3.0
4	1.33	1.5	1.6	2.0
5	1.25	1.36	1.42	1.66
6	1.2	1.28	1.33	1.5
7	1.16	1.23	1.27	1.4
8	1.14	1.2	1.23	1.33
9	1.12	1.17	1.2	1.28
10	1.11	1.15	1.17	1.25

3.5 Advantages of the Operation Trading Algorithm

We observed in the previous section that OTA(n) generates a DEQx allocation for n non-identical agents. Apart from this, the algorithm also guarantees an upper bound of $2 - \frac{2}{n+1}$. We present a few observations about the practical importance of OTA(n):

- **Decentralizable:** Since this algorithm consists of forming pairs and exchanging operations between them, this can be implemented as a decentralized algorithm.
- **Parallelizable:** Since pairs of agents trading operations can be independent, the algorithm is highly parallelizable.
- **Dynamic:** If new pop-up(s) appear, they can be assigned arbitrarily to any agent. Then, the agent performs the operation trading with other agents to distribute its load.
- **Robust:** If an agent malfunctions, all of its operations can get assigned randomly to other agents which can then be followed by the OTA(n).
- **Stochastic Processing Times:** If an operation is taking unexpectedly longer time or finished faster than expected, the trading operations can help redistribute the load.
- **Partial Connectivity:** The algorithm can operate under partial connectivity by performing Operation Trading among connected agents

only. Although the performance may degrade, the algorithm will not fail completely.

In the next chapter, we present a simple implementation of OTA(n) in a decentralized manner, named as Dec-OTA(n). Later in the results section, we evaluate the performance of Dec-OTA(n) on a network of agents that are not fully connected and show the effectiveness of the algorithm empirically.

Chapter 4

Decentralized OTA(n) (Dec-OTA(n))

In OTA(n), iterations through every possible pair of agents are being performed by a central computer. Now, consider the setting where a set of uniformly related agents are present in a communication degraded environment and the agents must form groups in a decentralized manner to perform Operation Trading (OTA(2)) between themselves. We must design a protocol by which agents can trade operations with as many as agents as possible in a decentralized manner given some partially connected communication graph between the agents. To achieve this, we propose the following simple protocol for each agent, agent i:

1. Generate a random number, γ_i from a uniform distribution with a predefined range (same for every agent).
2. Broadcast the number γ_i to all neighbors in the communication graph.
3. Receive the set of random values, $\Gamma = \{\gamma_j : j \in nbr(i)\}$.
4. $k \leftarrow \arg\max_j \gamma_j \quad \forall \gamma_j \in \Gamma$.
5. Send allocation A_i to agent k for operation trading if $\gamma_k > \gamma_i$.
6. $H = \{j : j \text{ sent its allocation to } i\}$.
7. After the updated allocation A_i is received from k, perform OTA(n) with the set of agents $H \cup i$.
8. Return updated allocations of every agent in H.

The agents run multiple iterations of this protocol to minimize the makespan. It can be observed that since the protocol forms groups using random numbers at every iteration, every pair of agents that have an edge between themselves perform operation trading with each other with a positive probability.

Further, if two disconnected agents have a common agent in their neighborhoods, they get to trade operations with each other with a certain probability. We call this protocol the Decentralized Random Group Formation (DRGF) protocol and the overall algorithm Decentralized OTA(n) (Dec-OTA(n)). It must be noted that the upper bounds on the makespan derived for OTA(n) do not hold for Dec-OTA(n) when the agents are partially connected.

Theorem 5. *The DRGF protocol does not lead to a deadlock under the assumption that the communication graph does not change during the considered iteration of group formation.*

Proof. Let us assume that the DRGF protocol leads to a deadlock. This implies that every agent is waiting to receive its updated allocation from the agent it had sent its allocation to. This is possible only if a loop has formed along which every agent has sent its allocation to the next agent. Consider the loop of agents $1, 2, ..., k, 1$ where every agent has sent its allocation to the agent on its right. From this we get:

$$\gamma_k > \gamma_{k-1} > ... > \gamma_1. \tag{4.1}$$

Since, agent k sends its allocation to agent 1, we also have:

$$\gamma_1 > \gamma_k. \tag{4.2}$$

This is however in contradiction to the Equation (4.1). Thus, we can conclude that the DRGF protocol does not enter into a deadlock. □

Lemma 1. *In Dec-OTA(n), every agent trades operations with every other agent with probability 1 when we have a fully connected graph.*

Proof. If we have a fully connected graph, the agent with the highest random value receives all the allocations and performs the OTA(n) algorithm. Thus, every possible pair of agents is considered in this case. □

Corollary 5.1. *Dec-OTA(n) is guaranteed to converge after a finite number of iterations.*

Proof. To prove that the OTA(n) converges even when implemented in a decentralized manner on a partially connected graph, we use the same potential function argument that was used to prove the convergence of OTA(n).

Every time the OTA(n) function is called, the makespan either reduces or stays the same. Since the makespan cannot decrease indefinitely, it ceases to decrease after a finite number of iterations. Following this, the second highest completion time starts decreasing. After the second highest completion time stops decreasing, the third highest completion time starts decreasing and so on. When the lowest completion time stops decreasing, the algorithm converges. □

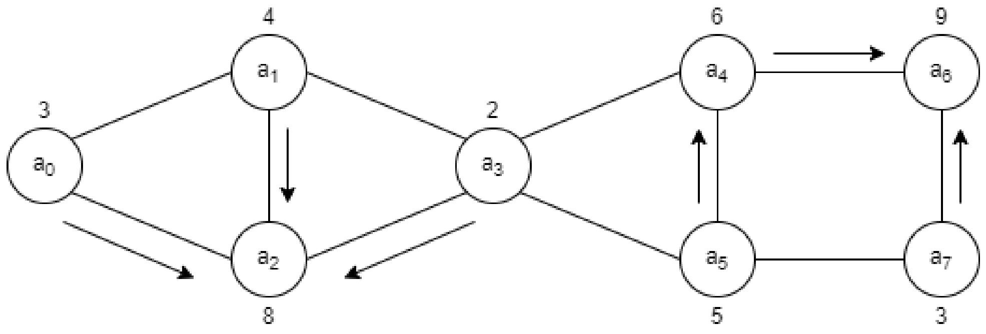


Figure 4.1: Example illustrating the DRGF protocol

Example 1. *Figure 4.1 illustrates an example of the DRGF protocol where some agent i is represented as node a_i. An undirected edge between 2 agents represents a 2 way communication link between them. Each agent generates a random number between 1 and 10 which is shown adjacent to the node. It can be observed that under the protocol, agents a_0, a_1 and a_3 send their allocations to a_2 for processing. Agents a_4 and a_7 send their allocations to a_6 while a_5 sends its allocation to a_4. a_4 waits for a_6 to update its allocation. Following this, a_4 performs OTA(n) using the allocations of a_4 and a_5. a_4 then returns the updated allocation of a_5 back to its owner. It can also be observed in this example that, even though a_0 and a_3 cannot communicate directly, they get to trade operations with each other via a_2.*

4.1 Limitations of the Operation Trading Algorithm

If the agents are identical or uniformly related, all possible allocations are Pareto Optimal. However, if we have non-identical agents, this is not true. Under such circumstances, OTA(n) may generate allocations that are far from the Pareto Optimal front. Thus, OTA(n) does not give any acceptable upper bound on the makespan for non-identical agents. This leads us to the next chapter where we present a Market Based Algorithm (referred to as MBA) to generate allocations that are guaranteed to satisfy the p-DEQ1 fairness criteria as well as Pareto Optimality. It is shown that guaranteeing these two properties automatically leads to an upper bound of 1.618 on the makespan for 2 non-identical agents. Further, the effectiveness of the algorithm in generating allocations with near optimal makespan for any number of non-identical agents is shown empirically in the results section.

Chapter 5

A market based algorithm for generating partially-DEQ1 + PO allocations

5.1 Preliminaries and Definitions for Market-Based Algorithm

Let $M = \langle [n], [m], V, s \rangle$ represent an instance consisting of n agents, a set of m divisible operations, a valuation V and a vector $s = (s_1, s_2, ..., s_n)$ where s_i is the total amount of money allocated to agent i. We also define a reward vector $r = (r_1, r_2, ..., r_m)$ where r_j is the reward corresponding to operation j. The Bang per Buck ratio of agent i for operation j is given by $\beta_{ij} = v_{ij}/r_j$. The Minimum Bang per Buck ratio of agent i is $\beta_i = min_j \beta_{ij}$. The Minimum Bang per Buck set of agent i is the set of all operations that minimize the Bang per Buck ratio for agent i, i.e., $\text{MnBB}_i = \{j \in [m] : v_{ij}/r_j = \beta_i\}$. Intuitively, the Bang per Buck ratio in this setting represents the cost incurred per unit of reward for performing an operation. The cost/valuation may represent the time taken to perform the operation or the effort required to perform the operation. Finally, as in Section 2.1, $x_{ij} = 1$ if operation j is assigned to agent i and 0 otherwise.

Theorem 6. *Given an instance $M = \langle [n], [m], V, s \rangle$ and a reward vector r, an allocation A is fractionally Pareto Optimal if the following conditions are satisfied:*

1. *For each operation $j \in [m]$, $\sum_{i=1}^{n} x_{ij} = 1$, i.e., all the operations are completely allocated.*
2. *For each agent $i \in [n]$, $\sum_{j=1}^{m} x_{ij} r_j = s_i$, i.e., each of the agents perform operations with total rewards worth exactly equal to the money given to it.*
3. *For every agent i and every operation j allocated to agent i, we have $j \in MnBB_i$, i.e., each agent's allocation is a subset of its MnBB set.*

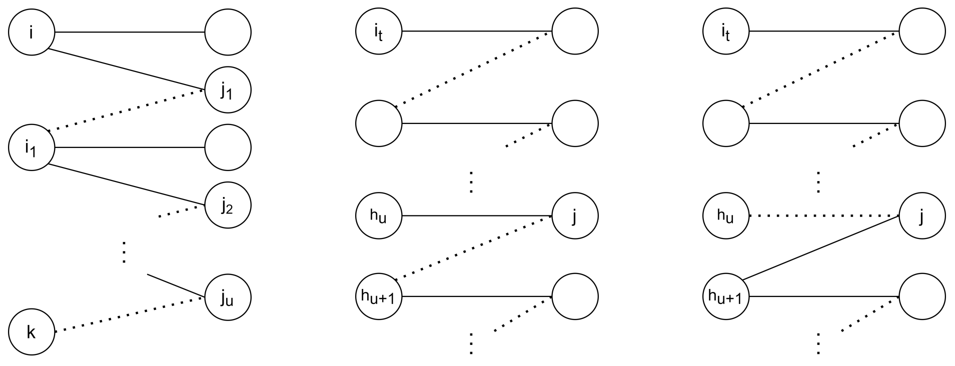
Proof. Let us consider an instance $M^{'} = \langle [n], [m], V^{'}, s\rangle$, where $V^{'} = -V$. Therefore, $M^{'}$ has negative valuations. Let A be the allocation under M satisfying the 3 conditions in the theorem. It is easy to observe that any Minimum Bang per Buck set in M is the Maximum Bang per Buck set in $M^{'}$ (Since $V^{'} = -V$). The 3 conditions thus imply a Fisher Market equilibrium for A under $M^{'}$. Using the first welfare theorem, it can be stated that the allocation A is fractionally PO under $M^{'}$ [29]. Since $V^{'} = -V$, the allocation A must be fractionally PO under the instance M as well. This proves the theorem. □

MnBB-allocation graph and alternating paths: Let A denote an integral allocation and r denote a reward vector. A MnBB allocation graph is an undirected bipartite graph G with vertex set $[n] \cup [m]$ and an edge between agent i and operation j if either $j \in A_i$ (called an allocation edge) or $j \in \text{MnBB}_i$ (called an MnBB edge). Any allocation is said to be MnBB consistent if each agent's allocation is a subset of its MnBB set. In such an allocation, the allocation edges are a subset of MnBB edges.

We define an alternating path $P = (i, j_1, i_1, j_2, \ldots, i_{u-1}, j_u, k)$ from agent i to agent k as a series of alternating allocations and MnBB edges such that $j_1 \in A_i \cap \text{MnBB}_{i_1}$, $j_2 \in A_{i_1} \cap \text{MnBB}_{i_2}$ and so on. If such a path exists, agent k is said to be reachable from agent i. The length of such a path is $2u$ since it consists of u allocation edges and u MnBB edges. Figure 5.1a shows the alternating path P along with the allocation edges (represented using solid edges) and MnBB edges (represented using dotted edges).

Reference Agent and Reachability Set: The agent with the largest total valuation is called the *reference agent*. All the alternating paths are constructed from this reference agent. Let the reference agent be denoted by h. We define the level of an agent k as half the length of the shortest alternating path from h to k if one exists, otherwise the level of k is set to n. The level of the reference agent is defined to be 0. The reachability set R_h of the reference agent h is defined as a level-wise collection of all agents that are reachable from h. Thus, $R_h = (R_h^0, R_h^1, R_h^2, \ldots)$, where R_h^u denotes the set of agents that are at level u with respect to agent h.

Violators and Path Violators: An agent k is said to be a *violator* if there does not exist an operation $j \in [m] \backslash A_k$ such that $v_k(A_k \cup j) \geq v_h(A_h)$. An allocation A is p-DEQ1 if and only if there are no violators.



(a) Alternating Path (b) Before Transfer of j (c) After Transfer of j

Figure 5.1: MnBB-allocation graphs and Alternating Paths

An agent $k \in R_i$ is a *path violator* with respect to the alternating path $P = (i, j_1, i_1, j_2, i_2, ..., i_{u-1}, j_u, k)$ if $v_k(A_k \cup j_u) < v_i(A_i)$. It must be noted that a path violator need not be a violator. However, if an agent is not a path violator, it is definitely not a violator as well. Similarly, given any $\epsilon > 0$, an agent k is an ϵ-path violator with respect to the alternating path P if $(1+\epsilon)v_k(A_k \cup j_u) < v_i(A_i)$.

ϵ-rounded instance: In an ϵ-rounded instance, the valuations are either 0 or a positive integral power of $(1+\epsilon)$, where $\epsilon > 0$. Let us assume that we have an instance I with valuations v_{ij} corresponding to every agent i and every operation j. From this, we generate an ϵ-rounded instance I' with valuations w_{ij} $\forall i \in [n]$ and $\forall j \in [m]$ using the following expression:

$$w_{ij} = \begin{cases} (1+\epsilon)^{\lceil log_{1+\epsilon} v_{ij} \rceil}, \text{if } v_{ij} > 0 \\ 0, \qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{5.1}$$

This expression rounds v_{ij} to the closest exponent of $(1+\epsilon)$.

5.2 Market Based Algorithm for partial-DEQ1 + PO allocation

In this section, we present an algorithm to generate a partial-DEQ1 allocation. We first generate an ϵ-rounded instance $I' = \langle [n], [m], W \rangle$ given an input instance $I = \langle [n], [m], V \rangle$. The instance I' is then used as an input to the algorithm.

The algorithm has three phases. In the first phase, each operation is assigned to the agent which has the lowest valuation for it. If the valuation represents time, the operation gets assigned to the agent that performs it the fastest. The rewards for each of the operations are initialized to the valuation of the operation corresponding to the agent it is allocated to. Since the rewards are equal to the valuations in this initialization procedure, the Bang per Buck ratio for each of the agents is also 1. If the allocation at the end of Phase 1 is ϵ-p-DEQ1, the algorithm returns this allocation and terminates; otherwise, there must exist an ϵ-violator and we proceed to Phase 2.

Phase 2 performs a level-by-level search for an ϵ-path violator in the reachability set of the reference agent starting from level $u = 1$. As soon as an ϵ-path violator is found along some alternating path P, the algorithm performs a transfer of operation from the agent preceding the ϵ-path violator to the ϵ-path violator along the path P. Since, the transfer takes place along the MnBB edges, the resultant allocation is also MnBB consistent and thus PO. Figures 5.1b and 5.1c show the transfer of operation j from i_u to i_{u+1}. After the transfer takes place, the algorithm restarts the Phase 2. Phase 2 terminates if there are no ϵ-path violators or if the allocation is ϵ-p-DEQ1. If there are no ϵ-path violators, the algorithm moves to Phase 3.

In Phase 3, the rewards for all the reachable operations are raised uniformly until a previously non-reachable agent becomes reachable due to a new MnBB edge. After an edge has been added, the algorithm again goes back to Phase 2.

5.3 Convergence and Complexity of Proposed MBA

The proof of convergence of this algorithm relies on several intermediate results. We first start with the analysis of Phase 2 of the algorithm. Let h_t denote the reference agent at time step t and R_{h_t} denote the reachability set at time step t. The level of an agent i is defined as

$$level(i,t) = \begin{cases} u & \text{if } \ i \in R^u_{h_t} \\ n & \text{otherwise.} \end{cases} \tag{5.2}$$

The level of the reference agent is thus 0. The level of any agent belonging to the reachability set cannot be more than $n-1$. An operation j is said to

Algorithm 3: Market Based Algorithm

```
1:  Phase 1 : Initialization
2:  Assign $j \in [m]$ to agent $i$ if $i \in \underset{k \in [n]}{arg\,min}(w_{kj})$
3:  $r \leftarrow \{\ r_j = w_{ij}$, if $j \in A_i$, $\forall j \in [m]\ \}$
4:  if  $A$ is $\epsilon - p - DEQ1$ then
5:      return $A$
6:  end if
7:  Phase 2 : Remove path violators in reachable agents
8:  $h \leftarrow$ reference agent in $A$
9:  $R_h \leftarrow$ Reachability set of $h$
10: $u = 1$
11: while $R_h^u$ is non-empty and $A$ is not $\epsilon$-p-DEQ1 do
12:     if $i \in R_h^u$ is $\epsilon$ path violator along alternating path
        $P = (h, j_1, i_1, ..., i_{u-1}, j, i)$ then
13:         $A_i \leftarrow A_i \cup \{j\}$
14:         $A_{i_{u-1}} \leftarrow A_{i_{u-1}} \setminus \{j\}$
15:         Repeat Phase 2 starting from Line 8
16:     else
17:         $u \leftarrow u + 1$
18:     end if
19: end while
20: if  $A$ is $\epsilon - p - DEQ1$ then
21:     return $A$
22: end if
23: Phase 3 : Price Rise to add edges
24: $\Delta \leftarrow \min_{i \in [n] \setminus R_h, j \in A_{R_h}} \frac{w_{ij}}{\beta_i \times r_j}$,
    where $A_{R_h} = \cup_{i \in R_h} A_i$ (set of reachable operations)
25: for all operation $j \in A_{R_h}$ do
26:     $r_j \leftarrow \Delta.r_j$
27: end for
28: Repeat Phase 2 starting from Line 8
```

be critical to an agent i at level u and at time t if j is allocated to the agent i and is also a part of some alternating path of length $u+1$ originating from the reference agent. G_{it} denotes the set of all critical operations belonging to agent i at time step t.

Lemma 2. *There can be at most $O(poly(m,n))$ consecutive swaps in Phase 2 before either the identity of the reference agent changes or a Phase 3 step occurs.*

Proof. Consider a potential function $F(t)$ defined as

$$F(t) = \sum_{i \in [n]} (m(n - level(i,t)) + |G_{it}|), \tag{5.3}$$

where $|G_{it}|$ is the cardinality of set G_{it}. Note that the value of $|G_{it}|$ cannot exceed m. The smallest value of $level(i,t) = 0$. Thus, the value of $F(t)$ cannot exceed $\sum_{i\in[n]} m(n+1) = mn(n+1)$ which is a polynomial in terms of both m and n. Also, the value of $F(t)$ can never be negative.

Let i_u and i_{u+1} be two agents on an alternating path at level u and $u+1$, respectively. Now, consider a transfer of operation j from agent i_u to i_{u+1}. Since this transfer does not affect the part of alternating path that comes between agent h_t and i_u, the level of i_u stays the same. However, the agent loses an operation from its set of critical operations. Thus, $|G_{i_u(t+1)}| = |G_{i_u t}| - 1$. This leads to a decrease in the potential function $F(t)$ by 1. Now, let us consider the change in the potential function contributed by agent i_{u+1}. After the transfer takes place, the agent i_{u+1} is not connected to h_t via the initial alternating path anymore since we now have a solid edge between j and i_{u+1}. This is illustrated in the Figures 5.1b and 5.1c. There are two possibilities remaining. The first is that the agent $i_{u+1} \notin R_{h_{t+1}}$. In this case, the level of agent i_{u+1} increases and $|G_{i_{u+1},t}|$ becomes 0. The potential value therefore decreases. In the other case, there is a different alternating path between h_t and i_{u+1}. The length of such an alternating path must be at least as much as the initial alternating path and thus, the level of i_{u+1} either stays the same or increases. It can be observed that the operation j cannot be a part of any alternating path and thus the newly added operation j does not contribute towards $|G_{i_{u+1}t}|$. Thus, the contribution of agent i_{u+1} to the potential function is either 0 or negative.

Now, we finally consider the rest of the agents. The swap may introduce new alternating paths. New alternating paths are introduced only when the level of an agent increases. It can be observed that the level of an agent cannot decrease since that would imply the existence of a shorter alternating path than the one considered at time step t. The level term in the potential function is scaled by a value of m. Thus, the decrease in $F(t)$ due to increase in the level of an agent cannot be negated by the increase in the number of critical operations introduced due to the new alternating paths. Again, the potential function either stays the same or decreases.

This proves that after every swap, the potential function must decrease by at least 1. Since, the potential function is upper bounded by $poly(m, n)$, there can be at most $O(poly(m, n))$ consecutive swaps in Phase 2 before either the identity of the reference agent changes or Phase 3 occurs. □

Lemma 3. *Consider any set of consecutive Phase 2 steps during the execution of the Market Based algorithm. Assume that the reference agent h turns to a non-reference agent during the time step t. Let $t' > t$ be the first time step after t at which h once again becomes a reference agent. Then, either A_h^t is a strict superset of $A_h^{t'}$ or $w_h(A_h^{t'}) < \frac{1}{1+\epsilon} w_h(A_h^t)$ where A_h^t is the bundle of operations belonging to agent i at time step t.*

Proof. Agent h turns from a reference agent to a non-reference agent at time step t by losing an operation. Thus, A_h^{t+1} is a strict subset of A_h^t. If the agent never gains an operation between iterations $t+1$ to t', we have that A_h^t is a strict superset of $A_h^{t'}$. Let us now assume the case where agent h gains at least one operation between the iterations $t+1$ to t'.

Among all the time steps between $t+1$ and t', let τ be the last time step at which agent h gains an operation. Let h_τ be the reference agent at time step τ. Since the total valuation of the reference agent never increases, we have

$$w_{h_\tau}(A_{h_\tau}^\tau) \leq w_h(A_h^t). \tag{5.4}$$

Let g be the operation gained by agent h at time step τ. An agent that gains an operation must be an ϵ-path violator. Thus,

$$(1+\epsilon) w_h(A_h^\tau \cup g) < w_{h_\tau}(A_{h_\tau}^\tau). \tag{5.5}$$

Since h does not gain any operation between τ and t', we get

$$w_h(A_h^{t'}) \leq w_h(A_h^{\tau} \cup g). \tag{5.6}$$

From Equations (5.4), (5.5) and (5.6), we have

$$w_h(A_h^{t'}) < \frac{1}{1+\epsilon} w_h(A_h^t).$$

□

Lemma 4. *The total valuation of the reference agent decreases by a factor of at least* $(1+\epsilon)$ *after* $O(poly(m,n))$ *consecutive iterations of Phase 2.*

Proof. From Lemma 3, we know that the total valuation of the reference agent either decreases by a factor of $1+\epsilon$ or loses an operation when it cycles back to being the reference agent. After n changes in the identity of the reference agent, there must be some agent that cycles back to being the reference agent. An agent cannot lose more than m operations consecutively. Thus, after $O(mn)$ changes in the identity of the reference agent, the total valuation of the reference agent must decrease by a factor of at least $1+\epsilon$. We also know from Lemma 2 that the identity of the reference agent changes after $O(poly(m,n))$ iterations. Thus, the total valuation of the reference agent must decrease by a factor of at least $1+\epsilon$ after $O(poly(m,n))$ iterations. □

Lemma 5. *There can be at most* $O(poly(m,n,1/\epsilon))$ *consecutive iterations of Phase 2 before Phase 3 starts.*

Proof. In Phase 1 of the algorithm, each operation is assigned to the agent with minimum valuation for it. In the worst case, all the operations get assigned to a single agent. Thus, the largest total valuation for an agent is given by

$$T_{min} = \sum_{j \in [m]} \min_i \{w_{ij}\}. \tag{5.7}$$

Ideally, this load should get equally distributed between all the agents. Thus, the total valuation of the reference agent can never be lower than T_{min}/n. From Lemma 4, we know that the total valuation of the reference agent decreases by a factor of $1+\epsilon$ after $O(poly(m,n))$ consecutive iterations of Phase 2. If the maximum number of iterations of Phase 2 before Phase 3 starts is μ, we have

$$\frac{T_{min}}{n}(1+\epsilon)^{\frac{\mu}{poly(m,n)}} \leq T_{min}$$

$$\implies \mu \geq \frac{ln(n)}{ln(1+\epsilon)} poly(m,n). \tag{5.8}$$

Now, for every $\epsilon \in (0,1)$, we have $\frac{1}{ln(1+\epsilon)} \leq \frac{2}{\epsilon}$. Thus, there can be at most $O(poly(m,n,1/\epsilon))$ consecutive iterations of Phase 2. □

We now move to the analysis of Phase 3 of the algorithm. Let E_t be the set of all ϵ-violators at time step t. Then, with h_t as the reference agent at time step t, E_t is defined as:

$$E_t = \{k \in [n] : (1+\epsilon)w_k(A_k^t \cup j) < w_{h_t}(A_{h_t}^t); \forall j \in [m] \backslash A_k^t\}.$$

Lemma 6. *We always have $E_{t'} \subseteq E_t$ if $t < t'$.*

Proof. Let t and t' be two consecutive Phase 3 time steps and A_k^t be the bundle of operations belonging to agent k at time step t. Proving the result for two consecutive Phase 3 time steps is enough. Since, t and t' are two consecutive Phase 3 time steps, time steps $t+1$, $t+2$, ..., $t'-1$ are in Phase 2.

We shall prove the Lemma using contradiction. Let us assume that there exists an agent $k \in E_{t'} \backslash E_t$. Thus, agent k turned into an ϵ-violator in some time step τ between t and t', i.e., in Phase 2. The only way agent k can turn into an ϵ-violator is through the transfer of some operation c from agent k at level l to an agent at level $l+1$. This will occur only if agent k was not an ϵ-path violator at time step τ. Thus, at time step τ, there exists an operation c' belonging to the agent before k in the alternating path such that

$$(1+\epsilon)w_k(A_k^\tau \cup c') \geq w_{h_\tau}(A_{h_\tau}^\tau). \tag{5.9}$$

After agent k loses operation c to become an ϵ-violator, we get

$$(1+\epsilon)w_k(A_k^\tau \backslash c \cup c') < w_{h_{\tau+1}}(A_{h_{\tau+1}}^{\tau+1}).$$

It can be observed that the transfer of operation c does not alter the alternating path between the reference agent and the agent k. Also, since the identity and the allocation of the reference agent does not change, $A_{h_{\tau+1}}^{\tau+1} = A_{h_\tau}^\tau$.

Agent k is now the only path violator. We have

$$(1+\epsilon)w_k(A_k^\tau \backslash c \cup c') < w_{h_\tau}(A_{h_\tau}^\tau).$$

Thus, in the iteration $\tau+1$, the operation c' gets transferred to agent k. The new allocation of agent k is $A_k^{\tau+1} = A_k^\tau \backslash c \cup c'$. From Equation (5.9), we get that agent k is non-ϵ-violator since adding the operation c to $A_k^{\tau+1}$ increases the total valuation of agent k to more than $w_{h_\tau}(A_{h_\tau}^\tau)$. This is a contradiction to our initial assumption that $k \in E_{t'} \backslash E_t$ and thus proves the lemma. □

Lemma 7. *Let t and t' be two Phase 3 time steps such that $t < t'$. Then, for any $k \in E_{t'}$, $A_k^{t'} \supseteq A_k^t$ where A_k^t is the bundle of operations belonging to agent k at time step t.*

Proof. Let us assume that there exists an operation $c \in A_k^t \backslash A_k^{t'}$. This implies that agent k transferred operation c to some other agent at some time step τ between t and t'. This can occur only if agent k was a non-ϵ-path violator at time step τ. We know that agent k is an ϵ-violator at time step t'. Thus, agent k turned from a non-ϵ-violator to an ϵ-violator. However, from Lemma 6 we know that any agent that is a non-ϵ-violator cannot become an ϵ-violator in the next Phase 3 time step which is a contradiction. Thus, our initial assumption was wrong and there exists no operation c such that $c \in A_k^t \backslash A_k^{t'}$. □

Lemma 8. *For any Phase 3 time step t, $E_t \cap R_{h_t} = \phi$.*

Proof. Let us consider a Phase 3 time step t. If there exists an agent k that belongs to $E_t \cap R_{h_t}$, then agent k is an ϵ-path violator that is reachable. This implies that Phase 2 has not converged yet, which is a contradiction. Thus, $E_t \cap R_{h_t} = \phi$ for any Phase 3 time step. □

Lemma 9. *For every agent $i \in [n]$, we have $\beta_i^t \geq \frac{1}{w_{max}}$ where β_i^t is the MnBB ratio of agent i at time step t and $w_{max} = \max_{i,j}\{w_{ij}\}$.*

Proof. The market based algorithm converges only if the allocation is p-DEQ1. Hence, if the allocation moves to Phase 3, there must exist some agent k which is an ϵ-violator. Let this Phase 3 time step be t. From Lemma 6, 7 and 8, it can be concluded that agent k has never seen an increase in price since the start of the algorithm. Thus, $\beta_k^t = 1$. For every operation j, we have

$$\begin{aligned} \frac{w_{kj}}{r_j} &\geq \beta_k^t = 1 \\ \implies r_j &\leq w_{kj} \\ \implies r_j &\leq w_{max}. \end{aligned} \tag{5.10}$$

where $w_{max} = \max_{i,j}\{w_{ij}\}$. By assumption, all the valuations in the original instance I are integral. Thus, for every agent i and operation $j \in A_i^t$, we get

$$\begin{aligned} &1 \leq v_{ij} \leq w_{ij} \leq w_{max} \\ \implies &\frac{w_{ij}}{p_j} \geq \frac{w_{ij}}{w_{max}} \geq \frac{1}{w_{max}} \\ \implies &\beta_i^t \geq \frac{1}{w_{max}}. \end{aligned} \tag{5.11}$$

□

Lemma 10. *There can be at most $O(poly(n, 1/\epsilon)ln(v_{max}))$ Phase 3 steps during the execution of the Market Based Algorithm for obtaining a p-DEQ1 allocation, where $v_{max} = \max_{i,j}\{v_{ij}\}$.*

Proof. The proof of this Lemma is identical to Lemma 4 in Reference [28] and is thus omitted. □

Lemma 11. *Given as input any ϵ-rounded instance I' with additive and strictly positive valuations, the Market Based Algorithm generates an allocation A that is ϵ-p-DEQ1 and PO in $O(poly(m, n, 1/\epsilon)ln(v_{max}))$ time steps, where $v_{max} = \max_{i,j}\{v_{ij}\}$.*

Proof. It can be observed that the Market Based algorithm (Algorithm 3) converges only if the allocation is ϵ-p-DEQ1. By design of the algorithm, the allocation generated is MnBB consistent in every time step and is thus also Pareto Optimal. From Lemma 5 and 10, it can be concluded that the algorithm converges after $O(poly(m, n, 1/\epsilon)ln(v_{max}))$ time steps. Thus, given as input any ϵ-rounded instance I' with additive and strictly positive valuations, the Market Based Algorithm generates an allocation A that is ϵ-p-DEQ1 and PO in $O(poly(m, n, 1/\epsilon)ln(v_{max}))$ time steps. □

5.4 Proof of Correctness of MBA

We now prove that the algorithm indeed generates an allocation which is p-DEQ1.

Lemma 12. *Given any instance I, its ϵ-rounded version I' and $\epsilon > 0$, an allocation A that is PO for I' is ϵ-PO for I.*

Proof. Let us assume for contradiction that there exists an allocation B that ϵ-Pareto Dominates the allocation A in I and that the allocation A is PO in I'. Thus, for every agent $k \in [n]$, we have $(1+\epsilon)v_k(B_k) \leq v_k(A_k)$ and for some agent $i \in [n]$, we have $(1+\epsilon)v_i(B_i) < v_i(A_i)$. Since, I' is the ϵ-rounded version of I, we have $v_{ij} \leq w_{ij} \leq (1+\epsilon)v_{ij}$ for every agent $i \in [n]$ and every operation $j \in [m]$. Thus, for every agent $k \in [n]$, we have $w_k(B_k) \leq w_k(A_k)$ and for some agent i, we have $w_i(B_i) < w_i(A_i)$. This implies that B Pareto dominates A in I' which is a contradiction. $\square$

Lemma 13. *Given any fair division instance $I = \langle [n], [m], V \rangle$ with additive, integral and strictly positive valuations, I' its ϵ-rounded version and for any $0 < \epsilon < \frac{1}{2mv_{max}}$, the allocation A returned by the market based algorithm for the input I' is Pareto Optimal for I.*

Proof. Let us assume that there exists an allocation B that Pareto Dominates the obtained allocation A in the instance I. We then have $v_k(B_k) \leq v_k(A_k)$ for all agent $k \in [n]$ and $v_i(B_i) < v_i(A_i)$ for some agent i. Since, we have assumed integral allocations, we have $v_i(B_i) \leq v_i(A_i) + 1$.

Now, since I' is an ϵ-rounded version of I, we have $v_{kj} \leq w_{kj} \leq (1+\epsilon)v_{kj}$ for all agents $k \in [n]$ and operations $j \in [m]$. Subsequently, we get

$$\begin{aligned} w_{kj} &\leq (1+\epsilon)v_{kj} \\ \implies \min_j \frac{w_{kj}}{p_j} &\leq (1+\epsilon)\min_j \frac{v_{kj}}{p_j} \\ \implies \beta_k &\leq (1+\epsilon)\alpha_k \\ \implies \frac{1}{\alpha_k} &\leq \frac{(1+\epsilon)}{\beta_k}, \end{aligned} \tag{5.12}$$

where α_k and β_k are the MnBB ratios of agent k in I and I'. Let us define $p(A_k)$ as the total reward agent k receives for performing the operations in the bundle A_k. Therefore,

$$\begin{aligned} \frac{v_k(A_k)}{\alpha_k} \leq \frac{w_k(A_k)}{\alpha_k} \leq \frac{(1+\epsilon)w_k(A_k)}{\beta_k} \\ \implies \frac{v_k(A_k)}{\alpha_k} \leq (1+\epsilon)p(A_k). \end{aligned} \tag{5.13}$$

We get the last inequality from the fact that the allocation A is MnBB consistent which implies that $p(A_k) = \frac{w_k(A_k)}{\beta_k}$. Let us now consider the allocation B. By definition of MnBB ratio, we get

$$\begin{aligned} \alpha_k \leq \frac{v_k(B_k)}{p(B_k)} \\ \implies \alpha_k p(B_k) \leq v_k(A_k) \\ \implies p(B_k) \leq (1+\epsilon)p(A_k). \end{aligned} \tag{5.14}$$

Similarly for agent i, we get

$$p(B_i) \leq (1+\epsilon)p(A_i) - \frac{1}{\alpha_i}. \tag{5.15}$$

Let the total reward over all the operations be $p([m])$. Note that this value depends only on the reward vector and is independent of the allocation. Thus,

$$\begin{aligned}
p([m]) &= \sum_{k\in[n]} p(B_k) \\
&= p(B_i) + \sum_{k\in[n]\setminus i} p(B_k) \\
&\leq (1+\epsilon)p(A_i) - \frac{1}{\alpha_i} + (1+\epsilon)\sum_{k\in[n]\setminus i} p(A_k) \\
&\leq (1+\epsilon)\sum_{k\in[n]} p(A_k) - \frac{1}{\alpha_i} \\
\implies p([m]) &\leq (1+\epsilon)p([m]) - \frac{1}{\alpha_i} \\
\implies 1 &\leq \epsilon\alpha_i p([m]) \\
\implies 1 &\leq \epsilon p([m]) \\
\implies \epsilon &\geq \frac{1}{m w_{max}} \qquad \text{(From Equation (5.10))} \\
\implies \epsilon(1+\epsilon) &\geq \frac{1}{m v_{max}}. \qquad (5.16)
\end{aligned}$$

We consider $0 < \epsilon < \frac{1}{2mv_{max}}$. Substituting this in Equation (5.16), we get $\epsilon > 1$ which is a contradiction since $\frac{1}{2mv_{max}} < 1$. Thus, our initial assumption that there exists an allocation B that Pareto dominates allocation A is incorrect and we get that the allocation A is Pareto Optimal for I. □

Lemma 14. *For an input instance I, its ϵ-rounded instance I' and $\epsilon > 0$, an allocation A that is ϵ-p-DEQ1 for I' is 3ϵ-p-DEQ1 for I.*

Proof. Since I' is the ϵ-rounded version of I, we have that $v_{kj} \leq w_{kj} \leq (1+\epsilon)v_{kj}$ for every agent k and every operation j. Let i be the reference agent (i.e., the agent with highest total valuation). Since A is ϵ-p-DEQ1, for every agent $k \in [n]$ there exists some operation j such that

$$\begin{aligned}
&(1+\epsilon)w_k(A_k \cup j) \geq w_i(A_i) \\
\implies &(1+\epsilon)^2 v_k(A_k \cup j) \geq v_i(A_i). \qquad (5.17)
\end{aligned}$$

From Lemma 13, we know that $\epsilon < \frac{1}{2mv_{max}} < 1$. Since $\epsilon < 1$, we get $(1+\epsilon)^2 < 1 + 3\epsilon$. Thus, for every agent $k \in [n]$ there exists some operation j such that

$$(1 + 3\epsilon)v_k(A_k \cup j) \geq v_i(A_i). \tag{5.18}$$

This implies that the allocation A is 3ϵ-p-DEQ1 in I. □

Lemma 15. *Given any instance I and $\epsilon < \frac{1}{3mv_{max}}$, an allocation A is 3ϵ-p-DEQ1 for I if and only if it is p-DEQ1 for I.*

Proof. Let i be the reference agent under the allocation A. Since A is 3ϵ-p-DEQ1 for I, for every agent $k \in [n]$, we have

$$\begin{aligned}
(1 + 3\epsilon)v_k(A_k \cup j) &\geq v_i(A_i) \\
\implies v_i(A_i) - v_k(A_k \cup j) &\leq 3\epsilon v_k(A_k \cup j) \\
&\leq 3\epsilon m v_{max} \\
&< 1.
\end{aligned}$$

We get the last result since we are considering $\epsilon < \frac{1}{3mv_{max}}$. Since the valuations are integral, $v_i(A_i) - v_k(A_k \cup j) < 1$ implies that $v_i(A_i) - v_k(A_k \cup j) \leq 0$. This is precisely the p-DEQ1 condition. □

Theorem 7. *Given any fair division instance $I = \langle [n], [m], V \rangle$ with additive, integral and strictly positive valuations, an allocation that is partially equitable up to one duplicated operation (p-DEQ1) and Pareto Optimal (PO) always exists and can be computed in $O(poly(m, n, 1/\epsilon)ln(v_{max}))$ time, where $v_{max} = max_{i,j}\{v_{ij}\}$.*

Proof. Let I' be the ϵ-rounded version of the input instance I. From Lemma 11, 13 and 15, it can be concluded that for $\epsilon < \frac{1}{3mv_{max}}$ and using I' as input, the Algorithm 3 computes an allocation A which is Pareto Optimal and p-DEQ1 in $O(poly(m, n, 1/\epsilon)ln(v_{max}))$ time, where $v_{max} = max_{i,j}\{v_{ij}\}$. □

5.5 Upper Bounds on Makespan for 2 Non-Identical Agents

Theorem 8. *The allocation generated by the Market Based Algorithm for 2 non-identical agents admits an upper bound of $\frac{\sqrt{5}+1}{2} \approx 1.618$ on the makespan.*

Proof. Let us consider 2 agents 1 and 2 and m operations. Let the obtained allocation after the Market Based Algorithm converges be A. Let the makespan of this allocation be $z = max(v_1(A_1), v_2(A_2))$. By design of the algorithm, there exists an alternating path between the agents which includes some operation k. Since the algorithm has converged, the allocation A is p-DEQ1 and PO. Thus, transferring the operation k along the alternating path must increase the value of makespan. It can be observed that the operation k can be fractionally allocated to both the agents in a way such that both their valuations become equal. Let this fractional allocation be A'. Additionally, this allocation A' is Pareto Optimal since the transfer of the fractional operation takes place along the alternating path. This implies that no other allocation with lower makespan can be obtained using fractional allocations. The allocation A' is thus an optimal solution to the Linear Program corresponding to the makespan. The makespan of A' is thus a lower bound on the optimal makespan. Let this lower bound be z^*. Let x_{1k} and x_{2k} be the fractions of the operation k assigned to agents 1 and 2. We can then write the following:

$$z = z^* + \min(v_{1k}x_{2k}, v_{2k}x_{1k}). \tag{5.19}$$

The fractions must sum to 1. Thus, $x_{1k} + x_{2k} = 1$. Also, the value of the fractional operation k allocated to agent 2 must be less or equal to the optimal makespan corresponding to the LP relaxation. Thus, $v_{2k}x_{2k} \leq z^*$. Combining these two constraints with Equation (5.19), gives us

$$z \leq z^* + \min(v_{1k}(1 - x_{1k}), x_{1k}\frac{z^*}{(1 - x_{1k})}). \tag{5.20}$$

Let us assume without loss of generality that $v_{1k} \leq v_{2k}$. This also implies that $v_{1k} \leq z^*$. This gives us:

$$\begin{aligned} z &\leq z^* + \min(v_{1k}(1 - x_{1k}), x_{1k}\frac{z^*}{(1 - x_{1k})}) \\ &\leq z^* + \min(z^*(1 - x_{1k}), x_{1k}\frac{z^*}{(1 - x_{1k})}) \\ \implies \frac{z}{z^*} &\leq 1 + \min(1 - x_{1k}, \frac{x_{1k}}{1 - x_{1k}}). \end{aligned} \tag{5.21}$$

The maximum of this term occurs when the 2 terms inside the min operator are equal i.e. $1 - x_{1k} = \frac{x_{1k}}{1-x_{1k}}$. Solving the equation gives us $x_{1k} = \frac{3-\sqrt{5}}{2}$. Substituting this value gives us

$$UB = \frac{z}{z^*} \leq \frac{\sqrt{5}+1}{2}. \tag{5.22}$$

□

In the next chapter, we prove that combining parametric pruning as in [1] with MBA leads to an upper bound of 1.5 for two non-identical agents, which is theoretically the best possible upper bound.

Chapter 6

Parametric Pruning leads to improved performance

In this chapter, we discuss the concept of Parametric Pruning and how it improves the performance of the algorithms introduced in this paper. Specifically, we shall prove that for 2 non-identical agents, combining Parametric Pruning with the Market Based Algorithm improves the upper bound from 1.618 to 1.5 which is the best possible upper bound [1].

Parametric Pruning was used in reference [1] to obtain schedules for non-identical parallel machines. In this method, we first define some threshold T. All the values of $v_{i,j} > T$ are 'pruned off' by setting them to a very large value M, i.e., they are not considered for generating the schedule. Following this, we define the linear program LP1(T) as:

$$\sum_{j:(i,j)\in S_T} x_{i,j} v_{i,j} \leq T \qquad \forall i \in [n],$$

$$\sum_{i:(i,j)\in S_T} x_{i,j} = 1 \qquad \forall j \in [m],$$

$$x_{i,j} \geq 0 \qquad \forall i \in [n], \forall j \in [m],$$

where S_T is the set of agent-operation pairs such that the operating time is not greater than T. In reference [1], it has been shown that the smallest value of T which generates a feasible solution for LP1 is a lower bound on the makespan. Let this value be $T^* = min\{T : LP1(T) \text{ is feasible}\}$. We further define $v'_{i,j}$ $\forall i \in [n], \forall j \in [m]$ as:

$$v'_{i,j} = \begin{cases} v_{i,j} & \text{, if } v_{i,j} < T^* \\ M & \text{, otherwise.} \end{cases}$$

Theorem 9. *For two non-identical agents, performing the Market Based Algorithm on the set of operating times generated using the threshold T^* guarantees an upper bound of 1.5 on the makespan.*

Proof. Let the 2 agents be 1 and 2. We already know that a feasible solution to LP1 exists with the threshold as T^*. Once MBA converges, we obtain an allocation A with an alternating path from agent 1 to agent 2 (We have assumed without loss of generality that the total valuation/completion time of agent 1 is greater than or equal to that of agent 2). Let the index of the operation in the alternating path be k. It can be observed that the operation k can be fractionally allocated to both the agents in a way such that both their valuations are equal. Let this fractional allocation be A'. Additionally, this allocation A' is Pareto Optimal since the transfer takes place along the alternating path. Thus, A' is an optimal solution to LP1(T^*). Let the fraction of the operation k transferred from agent 1 to agent 2 be x and the makespan corresponding to the fractional allocation A' be z. Since, the allocation A' is feasible, $v'_{1,k} \leq T^*$. We thus have the following two cases:

- Case 1 [$v'_{2,k} \leq T^*$]:

  In this case, we have the following equations:

$$v_1(A_1) \leq v_2(A_2) + v'_{2,k}, \tag{6.1}$$
$$z = v_1(A_1) - xv'_{1,k}, \tag{6.2}$$
$$z = v_2(A_2) + xv'_{2,k}, \tag{6.3}$$
$$v_1(A_1) \geq v_2(A_2), \tag{6.4}$$
$$v'_{1,k} \leq T^*, \tag{6.5}$$
$$v'_{2,k} \leq T^*. \tag{6.6}$$

  From Equations (6.2) and (6.3), we can write:

$$\begin{aligned} x &= \frac{v_1(A_1) - v_2(A_1)}{v'_{1,k} + v'_{2,k}} \\ \implies x &\leq \frac{v'_{2,k}}{v'_{1,k} + v'_{2,k}} \text{ [From Eq. (6.1)]} \\ \implies xv'_{1,k} &\leq \frac{v'_{1,k}v'_{2,k}}{v'_{1,k} + v'_{2,k}} \\ \implies xv'_{1,k} &\leq \frac{T^*}{2} \text{ [From Eqs. (6.6) and (6.5)].} \end{aligned} \tag{6.7}$$

  Since A' is the optimal solution to LP1 for the threshold T^*, we also have that:

$$\begin{aligned}
&z \leq T^* \\
\implies &v_1(A_1) \leq T^* + xv'_{1,k} \text{ [From Eq. (6.2)]} \\
\implies &v_1(A_1) \leq T^* + \frac{T^*}{2} \text{ [From Eq. (6.7)]} \\
\implies &UB = \frac{v_1(A_1)}{T^*} \leq \frac{3}{2}. \qquad (6.8)
\end{aligned}$$

- Case 2 $[v'_{2,k} > T^*]$:

  In this case, we have the same equations as Eqs. (6.1), (6.2), (6.3), (6.4) and (6.5). Additionally, we also have that $v'_{2,k} = M$. We again generate a fractional allocation. In this case, the fraction transferred from agent 1 to agent 2 would be infinitesimally small since $v'_{2,k}$ is a very large number. Thus the upper bound in this case is $1 + \epsilon$, where ϵ is a very small value.

From the 2 cases, it can be concluded that the upper bound is $\frac{3}{2} = 1.5$. $\square$

However, we would not know the value of T^* beforehand. Thus, we now present an algorithm that generates the required allocation by searching through different values of the threshold.

Let $T_1 = \max_i \min_j \{v_{i,j}\}$ where i represents agents and j represents operations. It can be observed that any value of threshold less than T_1 would generate an infeasible allocation. Let T_{arr} be a list of all the values of $v_{i,j} \geq T_1$ arranged in ascending order. The lower bound T^* must lie in between some T_t and T_{t+1} or exactly on some T_t. It can be observed that for any value of threshold T such that $T_t \leq T < T_{t+1}$, the allocation generated would be the same. Thus, the allocation corresponding to T^* would be the same as that for T_t.

Since T^* is the smallest value with a feasible solution to LP1, any value smaller than T_t would generate an infeasible solution to LP1. Either T_t or T_{t+1} would be the first threshold value to generate a feasible solution, following which we need not search further. Based on these observations, we propose Algorithm 4 to generate an allocation with upper bound of 1.5.

Whether this approach can be used to generate allocations with an upper bound for more than 2 agents is not known and is left as an open problem.

Algorithm 4: Market Based Algorithm + Pruning (for 2 agents)

```
1: $T_1 = \max_i \min_j \{v_{i,j}\}$
2: $T_{arr} \leftarrow \{v_{i,j} : v_{i,j} \geq T_1\}$
3: Sort elements of $T_{arr}$ in ascending order
4: $A_{list} \leftarrow \phi$
5: while $i \leq |T_{arr}|$ do
6:    Set $v_{i,j} = M \quad \forall v_{i,j} > T_{arr,i}$
7:    $A \leftarrow$ Market Based Algorithm
8:    $A_{list} \leftarrow A_{list} \cup A$
9:    $A' \leftarrow$ Optimal fractional allocation
10:   $Z' \leftarrow$ Makespan of fractional allocation $A'$
11:   if $Z' < T_{arr,i}$ then
12:      Break Loop
13:   end if
14: end while
15: Output the allocation with best makespan from $A_{list}$
```

Chapter 7

Results

In this section, we numerically evaluate the Dec-OTA(n) algorithm on randomly generated instances by comparing the obtained makespan to the optimal makespan. The experiments were conducted on a Lenovo Yoga i9 laptop with 3 GHz 11th Gen Intel Core i7 CPU and 16GB RAM. The programs were coded in Python on Jupyter notebook. The optimizer used to obtain the optimal makespans is Gurobi Optimizer version 9.0.

Along with OTA(n) (Algorithm 2) and MBA (Algorithm 3), we also consider pairwise-Market based Algorithm. In the pairwise-MBA, we consider all possible pairs (exactly like in OTA(n)) and generate an allocation using MBA for every pair. We continue this until no pair exists which can be improved using MBA. We consider this particular variation of MBA because unlike the original MBA, pairwise-MBA can be decentralized easily. It must be noted that the pairwise-MBA is not guaranteed to generate a Pareto Optimal allocation. It is easy to observe that for 2 agents, partial-DEQ1 and DEQ1 are equivalent. Thus, in the final allocation generated by pairwise-MBA, every pair of agents is DEQ1. This implies that the allocation generated by pairwise-MBA also satisfies DEQ1 overall.

7.1 Numerical Results on Identical Agents

The results of the three algorithms on identical agents are presented in Table 7.1 along with a comparison with the Longest Processing Time (LPT) rule. We have chosen the LPT rule for comparison mainly because of its popularity, simplicity and extremely good upper bound ($\frac{4}{3} - \frac{1}{3n}$). There do exist algorithms which have better upper bounds [30][31], but they come at the cost of higher running times and no visible improvement on most practical applications. Each entry is the ratio of obtained makespan and optimal makespan averaged over 5 random instances. The number of agents (n) considered are 2, 5, 10 and 20. The number of operations (m) considered are 25, 50, 100, 200 and 500. The processing times are integers between 1 and 50 generated using a uniform probability distribution. The important observations are listed below:

- All the algorithms generate near optimal solutions for almost all of the cases.
- MBA performs better than OTA(n) and pairwise-MBA even though it has the weakest guarantee in terms of fairness. MBA is guaranteed to generate only a partial-DEQ1 allocation compared to the DEQx guarantee of OTA(n) and DEQ1 guarantee of pairwise-MBA.
- The performance improves as the number of operations increase with respect to the number of agents. This can be explained using the expression derived in Theorem 3.
- The LPT rule performs much better than the Fair Allocation algorithms presented in this paper. The performance of MBA however is almost as good as the LPT rule.

7.2 Numerical Results on Uniformly Related agents and Non-Identical Agents

For uniformly related agents and non-identical agents, we consider Parametric Pruning as well. For each algorithm, we generate results with and without Pruning. Additionally, we also present the performance of the Lenstra'a algorithm [1] for comparison. We have chosen Lenstra's algorithm even though it is old because it is still one of the very few algorithms which guarantees a constant factor approximation of 2 on the makespan on unrelated parallel machines in polynomial time. The Lenstra's algorithm is the most widely studied algorithm for this particular problem in the literature. The only other algorithm that we know of with a better upper bound is an extension of the Lenstra's algorithm which optimizes the rounding [4]. The results are presented in Table 7.3 and 7.2. Each entry is the ratio of the obtained

Table 7.1: Ratios (Algorithm Makespan/Optimal Makespan) obtained for identical agents

n	m	OTA(n)	MBA	pairwise-MBA	LPT
2	25	1.00	1.00	1.02	1.00
	50	1.00	1.00	1.01	1.00
	100	1.00	1.00	1.00	1.00
	200	1.00	1.00	1.00	1.00
	500	1.00	1.00	1.00	1.00
5	25	1.03	1.03	1.10	1.03
	50	1.01	1.01	1.01	1.00
	100	1.00	1.00	1.00	1.00
	200	1.00	1.00	1.00	1.00
	500	1.00	1.00	1.00	1.00
10	25	1.20	1.15	1.22	1.03
	50	1.06	1.02	1.15	1.02
	100	1.01	1.01	1.01	1.00
	200	1.00	1.00	1.00	1.00
	500	1.00	1.00	1.00	1.00
20	25	1.00	1.00	1.00	1.00
	50	1.28	1.09	1.31	1.05
	100	1.03	1.03	1.13	1.02
	200	1.01	1.01	1.01	1.00
	500	1.00	1.00	1.00	1.00

makespan and optimal makespan averaged over 5 random instances. The number of agents (n) considered are 2, 5, 10 and 20. The number of operations (m) considered are 25, 50, 100, 200 and 500. The processing times are integers between 1 and 50 generated using a uniform probability distribution. For the uniformly relate agents, the speeds of the agents lie between 1 and 2 and have been selected randomly using a uniform probability distribution. For non-identical/unrelated agents, the processing times are different for every agent. Columns indicated with (Pr) and (No Pr) represent Pruning and no Pruning, respectively. The important observations are listed below:

- MBA generates near optimal allocations for almost all the cases. This can be attributed to the fact that MBA always generates Pareto Optimal allocations.
- Compared to MBA, OTA(n) and pairwise-MBA have poorer performance, especially for the non-identical agents. This is primarily because these two algorithms may generate allocations that are far from the Pareto Optimal front.
- Both MBA and pairwise-MBA significantly outperform the Lenstra's algorithm even without pruning for both uniformly related and unrelated agents. Additionally, the Lenstra's algorithm is not guaranteed to satisfy any fair allocation criteria, unlike MBA and pairwise-MBA.
- OTA(n) especially displays poor performance with the average ratio reaching values as high as 1.69 for uniformly related agents and 1.59 for non-identical agents. Pairwise-MBA performs significantly better than OTA(n).
- Pruning leads to significant improvements when combined with OTA(n). By pruning out high operating times, Parametric Pruning helps in generating allocations closer to the Pareto Optimal front. However, OTA(n) combined with Pruning still does not perform as good as MBA and pairwise-MBA.
- Pruning when combined with MBA or pairwise-MBA does not lead to any improvement. For the case of MBA, this can be explained by the fact that we always obtain allocations that are Pareto Optimal. This

Table 7.2: Ratios (Algorithm Makespan/Optimal Makespan) obtained for uniformly related agents agents

		OTA(n)		MBA		pairwise-MBA		Lenstra
n	m	No Pr	Pr	No Pr	Pr	No Pr	Pr	
2	25	1.00	1.00	1.00	1.00	1.01	1.01	1.01
	50	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	100	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	200	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	500	1.00	1.00	1.00	1.00	1.00	1.00	1.00
5	25	1.08	1.03	1.04	1.03	1.07	1.05	1.21
	50	1.01	1.01	1.02	1.00	1.02	1.01	1.11
	100	1.01	1.00	1.01	1.00	1.01	1.00	1.05
	200	1.00	1.00	1.00	1.00	1.00	1.00	1.03
	500	1.00	1.00	1.00	1.00	1.00	1.00	1.01
10	25	1.24	1.38	1.14	1.11	1.19	1.20	1.68
	50	1.31	1.05	1.07	1.02	1.07	1.08	1.26
	100	1.33	1.02	1.02	1.01	1.01	1.01	1.16
	200	1.05	1.01	1.01	1.00	1.00	1.00	1.05
	500	1.02	1.00	1.00	1.00	1.00	1.00	1.03
20	25	1.49	1.18	1.22	1.00	1.33	1.18	1.62
	50	1.69	1.28	1.17	1.12	1.21	1.24	1.60
	100	1.45	1.17	1.08	1.05	1.11	1.12	1.36
	200	1.16	1.03	1.03	1.02	1.01	1.01	1.15
	500	1.23	1.02	1.01	1.00	1.00	1.00	1.08

is in contrary to the case of OTA(n) where pruning lead to solutions closer to the Pareto Optimal front. Hence, numerically we do not see any improvement. The explanation for pairwise-MBA is tougher to find and is left as an open problem. We suspect that the reason behind the observation is that in every pairwise interaction, the allocations generated for each pair are Pareto Optimal at all instances due to which any improvement cannot be observed numerically.

7.3 Numerical Evaluation of the DRGF Protocol

The results of the Dec-OTA(n) on uniformly-related agents are presented in Tables 7.4 and 7.5, and Figures 7.1a and 7.1b. We present the results only for uniformly related agents firstly because most practical applications have

Table 7.3: Ratios (Algorithm Makespan/Optimal Makespan) obtained for non-identical agents

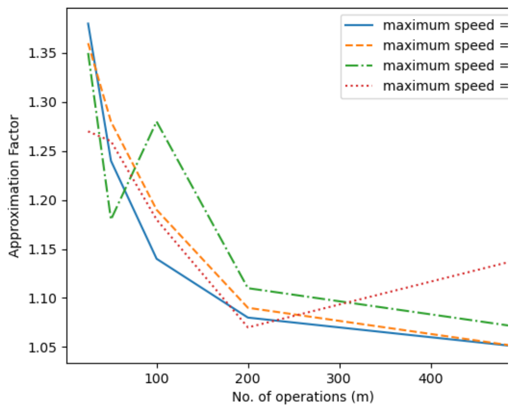
		OTA(n)		MBA		pairwise-MBA		Lenstra
n	m	No Pr	Pr	No Pr	Pr	No Pr	Pr	
2	25	1.25	1.08	1.04	1.01	1.01	1.01	1.03
	50	1.23	1.11	1.01	1.02	1.01	1.00	1.01
	100	1.31	1.26	1.01	1.01	1.01	1.00	1.01
	200	1.32	1.29	1.00	1.01	1.01	1.00	1.00
	500	1.28	1.24	1.00	1.00	1.00	1.00	1.00
5	25	1.34	1.15	1.08	1.08	1.07	1.08	1.19
	50	1.29	1.19	1.09	1.05	1.03	1.09	1.19
	100	1.29	1.20	1.03	1.04	1.04	1.02	1.05
	200	1.39	1.17	1.02	1.02	1.02	1.02	1.04
	500	1.45	1.31	1.01	1.01	1.01	1.01	1.01
10	25	1.38	1.16	1.13	1.07	1.26	1.17	1.50
	50	1.34	1.19	1.15	1.07	1.17	1.10	1.32
	100	1.37	1.27	1.07	1.07	1.07	1.09	1.19
	200	1.50	1.24	1.05	1.03	1.05	1.04	1.14
	500	1.41	1.26	1.02	1.02	1.02	1.02	1.04
20	25	1.34	1.12	1.02	1.00	1.08	1.00	1.45
	50	1.44	1.20	1.06	1.02	1.26	1.30	1.60
	100	1.59	1.28	1.15	1.11	1.20	1.19	1.38
	200	1.52	1.29	1.09	1.05	1.17	1.19	1.23
	500	1.42	1.25	1.03	1.04	1.07	1.08	1.13

Table 7.4: Ratios (Algorithm Makespan/Optimal Makespan) obtained for uniformly related agents using Dec-OTA(n)

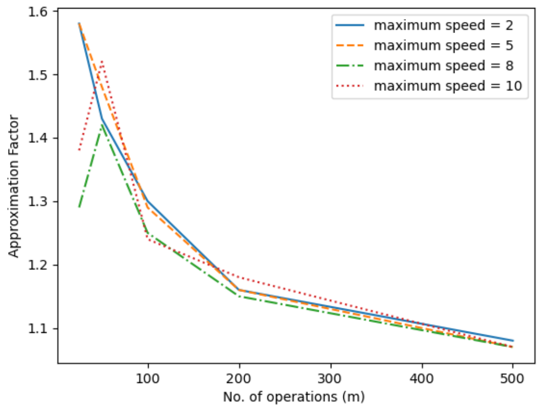
		con = 1.0			con = 0.8			con = 0.5		
n	m	UB	Itr	Time(s)	UB	Itr	Time(s)	UB	Itr	Time(s)
2	25	1.003	2	0.002	1.002	2	0.001	1.003	2	0.001
	50	1.0	2	0.003	1.0	2	0.001	1.0	2	0.001
	100	1.0	2	0.008	1.0	2	0.005	1.0	2	0.004
	200	1.0	2	0.009	1.0	2	0.014	1.0	2	0.009
	500	1.0	2	0.04	1.0	2	0.037	1.0	2	0.03
5	25	1.04	1.4	0.007	1.04	3	0.009	1.18	1.8	0.001
	50	1.02	1.8	0.01	1.05	2.2	0.013	1.22	2.4	0.007
	100	1.006	2	0.02	1.05	3.2	0.02	1.15	3.8	0.02
	200	1.002	2	0.04	1.001	2.8	0.035	1.16	4.6	0.02
	500	1.0	2.2	0.109	1.0	4.6	0.11	1.05	4.8	0.122
10	25	1.17	1.2	0.011	1.28	1.6	0.009	1.36	2.8	0.013
	50	1.12	1	0.011	1.29	1.6	0.009	1.50	3.8	0.021
	100	1.14	1.6	0.02	1.22	2	0.024	1.412	2.8	0.024
	200	1.04	1.8	0.049	1.042	3.4	0.076	1.086	2.4	0.038
	500	1.015	2	0.201	1.006	5	0.17	1.092	6.6	0.154
20	25	1.54	1	0.02	1.77	1.4	0.017	1.82	1	0.008
	50	1.3	1.2	0.034	1.32	1.6	0.02	1.54	2.0	0.022
	100	1.22	1.0	0.013	1.19	1.8	0.011	1.24	2	0.03
	200	1.314	1	0.056	1.09	2.2	0.078	1.233	2.4	0.055
	500	1.11	1.6	0.169	1.025	3.8	0.225	1.14	3.6	0.186

Table 7.5: Ratios (Algorithm Makespan/Optimal Makespan) obtained for uniformly related agents using Dec-OTA(n)

		con = 0.3			Line Graph		
n	m	UB	Itr	Time(s)	UB	Itr	Time(s)
2	25	1.002	2	0.001	1.003	2	0.002
	50	1.0	2	0.003	1.0	2	0.002
	100	1.0	2	0.004	1.0	2	0.004
	200	1.0	2	0.01	1.0	2	0.009
	500	1.0	2	0.034	1.0	2	0.036
5	25	1.18	2.8	0.006	1.3	3	0.0013
	50	1.103	2.8	0.019	1.28	2.4	0.0036
	100	1.57	3.0	0.013	1.22	4.4	0.015
	200	1.26	3.6	0.02	1.18	4	0.033
	500	1.16	3.2	0.07	1.15	3.4	0.043
10	25	1.63	1.6	0.006	1.42	2.4	0.007
	50	1.31	1.8	0.012	1.49	2.8	0.007
	100	1.17	3	0.021	1.38	3.1	0.018
	200	1.255	3.4	0.032	1.54	4	0.027
	500	1.167	3.2	0.1	1.6	3.1	0.046
20	25	1.74	1.2	0.005	1.97	1.6	0.0027
	50	1.53	2	0.017	1.93	2	0.01
	100	1.31	2.4	0.03	1.8	2.6	0.024
	200	1.211	3.4	0.056	1.79	3.5	0.029
	500	1.137	4.4	0.161	1.66	5.1	0.046



(a) $n = 10$



(b) $n = 20$

Figure 7.1: Approximation factors for agents with higher speeds

uniformly related agents and secondly because all possible allocations are Pareto Optimal for such agents. This is not true for non-identical agents. If two agents cannot communicate and the allocations are not guaranteed to be Pareto Optimal, the allocation generated by Dec-OTA(n) can be arbitrarily bad. We leave this problem for future work. The number of agents (n) considered are 2, 5, 10 and 20. The number of operations (m) considered are 25, 50, 100, 200 and 500. The processing times are integers between 10 and 50 generated using a uniform probability distribution. The speeds of the agents in Tables 7.4 and 7.5 lie between 1 and 2 and have been selected randomly using a uniform probability distribution. The variable *con* in the table represents the probability that any 2 agents are connected given that the overall graph is connected. *con* thus controls the connectivity in the graph. The performance of Dec-OTA(n) is also evaluated on the line graph which represents the worst case scenario under connected- set of agents. For each level of connectivity, the table lists the Upper Bound (UB) (which is the ratio of obtained makespan and optimal makespan), number of iterations to converge (itr) and the time to converge, all averaged over 5 random instances. In Figure 5.1, we present the approximation factors obtained numerically considering a line graph (worst case scenario) when the speeds of the agents lie between a larger range. The important observations from Tables 7.4 and 7.5, and Figure 7.1 are listed below:

- Dec-OTA(n) generates near optimal allocations for most cases, even under severe communication failures, i.e., $con = 0.3$.

- The average performance of Dec-OTA(n) does not exceed an approximation factor of 2 even in the most extreme case, i.e., the line graph.

- The numerical performance of Dec-OTA(n) is just as strong when the maximum possible normalized speed of an agent is 10 as it is when the maximum possible normalized speed is 2 as considered in the Table. It does not vary significantly when the range of the speeds of the agents increases. This can be observed from Figure 5.1.

- The performance of Dec-OTA(n) improves as the number of operations increases compared to the number of agents. This can be explained using Equation (3.18). As the number of operations increases, the value of k increases as well, resulting in an improved approximation factor ratio. The algorithm is thus, a good choice for large scale practical problems.

- The algorithm can start from any random initialization and generates an allocation in under 0.6 seconds even for large scale problems. Thus, the algorithm can be used for quick re-planning in case of machine failures, pop-up jobs, changes in operating times, etc.

It can be concluded that Dec-OTA(n) is robust to communication and machine failures due to the DRGF protocol. Additionally, the algorithm is fast and can be used in a reactive manner to tackle constant changes in the environment. The algorithm is also parameter-less, thus not requiring any form of tuning.

The codes can be found at: Github Repository

Chapter 8

Conclusion and Future Work

In this thesis, we present the Operation Trading algorithm to generate DEQx allocations and a Market Based Algorithm to generate partial-DEQ1 allocations. We then show both theoretically and numerically that these algorithms can be effectively used to generate allocations with near optimal makespans. OTA(n) guarantees an upper bound of $2 - \frac{1}{n+1}$ for n identical agents and an upper bound of 1.618 for 2 uniformly related agents. The algorithm also guarantees an upper bound of $\frac{1+\sqrt{4n-3}}{2}$ for n uniformly related agents in general. Further, we have shown that under certain cases which occur often in practical scenarios, OTA(n) guarantees acceptable upper bounds for uniformly related agents. MBA guarantees an upper bound of 1.618 for 2 non-identical agents. This shows the similarities in the mathematical structures between the fair allocation problem and makespan minimization problem. Incorporating Parametric Pruning was further shown to improve the performances both theoretically and numerically. The thesis also presents the idea of performing allocation in a pairwise manner (OTA(n) and pairwise-MBA) which has the advantages of being decentralizable, robust and reactive. The DRGF protocol is presented to implement the algorithms in a decentralized and distributed manner. It has been shown numercially that the DRGF protocol has fast convergence and generates near-optimal allocations even under severe communication failures.

There are primarily two ways to extend this research. The first way is to broaden the scope of the work and try to incorporate precedence constraints, travel times, spatially distributed tasks, geometric constraints (for multi-robot systems), and alike. The second way to extend the paper is to dive deeper and try to develop combinatorial (and preferably decentralizable) algorithms that guarantee an upper bound for any number of non-identical agents. The problem of generating DEQ1+PO allocations still remains an open problem. An interesting approach would be to look for modifications of the market based algorithm to generate DEQ1 allocations. Further, the presented algorithms do not guarantee a constant factor approximation even for identical agents if the agents are not fully connected. In order to investigate the possibility of guaranteeing a constant factor approximation for the makespan minimization problem of identical and uniformly related agents even when the agents are not fully connected would require characterizing the hardness of the makespan minimization problem in terms of decentralization and distribution. Principled methods to achieve this does not exist currently in the literature and is of great interest.

References

[1] J. K. Lenstra, D. B. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," in 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), 1987, pp. 217–224.

[2] C. Potts, "Analysis of a linear programming heuristic for scheduling unrelated parallel machines," Discrete Applied Mathematics, vol. 10, no. 2, pp. 155 – 164, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0166218X85900095

[3] D. S. Hochbaum and D. B. Shmoys, "A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach," SIAM Journal on Computing, vol. 17, no. 3, pp. 539–551, 1988. [Online]. Available: https://doi.org/10.1137/0217033

[4] E. V. Shchepin and N. Vakhania, "An optimal rounding gives a better approximation for scheduling unrelated machines," Operations Research Letters, vol. 33, no. 2, pp. 127 – 133, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167637704000690

[5] R. Freeman, S. Sikdar, R. Vaish, and L. Xia, "Equitable allocations of indivisible chores." Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[6] T. Ebenlendr, M. Krčál, and J. Sgall, "Graph balancing: A special case of scheduling unrelated parallel machines," ser. SODA '08. USA: Society for Industrial and Applied Mathematics, 2008, p. 483–490.

[7] M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 3rd ed. Springer Publishing Company, Incorporated, 2008.

[8] T. Gonzalez, O. Ibarra, and S. Sahni, "Bounds for lpt schedules on uniform processors," SIAM J. Comput., vol. 6, pp. 155–166, 03 1977.

[9] D. Friesen and M. Langston, "Bounds for multifit scheduling on uniform processors," SIAM J. Comput., vol. 12, pp. 60–70, 02 1983.

[10] B. P. Gerkey and M. J. Mataric, "Sold!: auction methods for multirobot coordination," IEEE Transactions on Robotics and Automation, vol. 18, no. 5, pp. 758–768, 2002.

[11] M. B. Dias and A. T. Stentz, "A free market architecture for distributed control of a multirobot system," in Proceedings of 6th International Conference on Intelligent Autonomous Systems (IAS-6), July 2000, pp. 115–122.

[12] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," Proceedings of the IEEE, vol. 94, no. 7, pp. 1257–1270, 2006.

[13] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," The International Journal of Robotics Research, vol. 23, no. 9, pp. 939–954, 2004. [Online]. Available: https://doi.org/10.1177/0278364904045564

[14] M. Alighanbari and J. How, "Decentralized task assignment for unmanned aerial vehicles," 01 2006, pp. 5668 – 5673.

[15] Z. Li, Z. Duan, G. Chen, and L. Huang, "Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 1, pp. 213–224, 2010.

[16] H. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," IEEE Transactions on Robotics, vol. 25, no. 4, pp. 912–926, 2009.

[17] M. Franceschelli, A. Giua, and C. Seatzu, "Fast discrete consensus based on gossip for makespan minimization in networked systems," Automatica, vol. 56, pp. 60–69, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109815001028

[18] M. Franceschelli, A. Giua, and C. Seatzu, "Gossip based asynchronous and randomized distributed task assignment with guaranteed performance on heterogeneous networks," Nonlinear Analysis: Hybrid Systems, vol. 26, pp. 292–306, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1751570X17300535

[19] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," Journal of Intelligent Manufacturing, vol. 29, no. 3, pp. 603–615, 2018. [Online]. Available: https://EconPapers.repec.org/RePEc:spr:joinma:v:29:y:2018:i:3:d:10.1007_s10845-015-1039-3

[20] F. Y. Nedwed, I. Zinnikus, M. Nukhayev, M. Klusch, and L. Mazzola, "shopst: Flexible job-shop scheduling with agent-based simulated trading," in Multiagent System Technologies, J. O. Berndt, P. Petta, and R. Unland, Eds. Cham: Springer International Publishing, 2017, pp. 121–137.

[21] L. Asadzadeh and K. Zamanifar, "An agent-based parallel approach for the job shop scheduling problem with genetic algorithms," Mathematical and Computer Modelling, vol. 52, no. 11, pp. 1957–1965, 2010, the BIC-TA 2009 Special Issue. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0895717710002098

[22] E. Budish, "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes," Journal of Political Economy, vol. 119, no. 6, pp. 1061–1103, 2011. [Online]. Available: https://doi.org/10.1086/664613

[23] A. D. Procaccia and J. Wang, "Fair enough: Guaranteeing approximate maximin shares," in Proceedings of the Fifteenth ACM Conference on Economics and Computation, ser. EC '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2600057.2602835 p. 675–692.

[24] B. Plaut and T. Roughgarden, "Almost envy-freeness with general valuations," in Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018, A. Czumaj, Ed. SIAM, 2018. [Online]. Available: https://doi.org/10.1137/1.9781611975031.165 pp. 2584–2603.

[25] D. Foley, "Resource allocation and the public sector," Yale Economic Essays, vol. 7, pp. 45–98, 1967. [Online]. Available: https://ci.nii.ac.jp/naid/10009821677/en/

[26] L. E. Dubins and E. H. Spanier, "How to cut a cake fairly," The American Mathematical Monthly, vol. 68, no. 1, pp. 1–17, 1961. [Online]. Available: http://www.jstor.org/stable/2311357

[27] L. Gourvès, J. Monnot, and L. Tlilane, "Near fairness in matroids," ser. ECAI'14. NLD: IOS Press, 2014, p. 393–398.

[28] R. Freeman, S. Sikdar, R. Vaish, and L. Xia, "Equitable allocations of indivisible goods," CoRR, vol. abs/1905.10656, 2019. [Online]. Available: http://arxiv.org/abs/1905.10656

[29] A. Mas-Colell, M. Whinston, and J. Green, Microeconomic Theory. Oxford University Press, 1995. [Online]. Available: https://EconPapers.repec.org/RePEc:oxp:obooks:9780195102680

[30] C.-Y. Lee and J. David Massey, "Multiprocessor scheduling: combining lpt and multifit," Discrete Applied Mathematics, vol. 20, no. 3, pp. 233–242, 1988. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0166218X88900790

[31] J. N. D. Gupta and A. J. Ruiz-Torres, "A listfit heuristic for minimizing makespan on identical parallel machines," Production Planning & Control, vol. 12, no. 1, pp. 28–36, 2001. [Online]. Available: https://doi.org/10.1080/09537280150203951