1-1-2012

Truthful Mechanisms For Real-Time System Scheduling In Competitive Environments

Anwar Mohammadi
Wayne State University,

Recommended Citation

Mohammadi, Anwar, "Truthful Mechanisms For Real-Time System Scheduling In Competitive Environments" (2012). *Wayne State University Theses.* Paper 239.

TRUTHFUL MECHANISMS FOR REAL-TIME SYSTEM SCHEDULING IN COMPETITIVE ENVIRONMENTS

by

ANWAR MOHAMMADI

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2012

MAJOR: COMPUTER SCIENCE

Approved by:

Advisor Date

DEDICATION

To my father and mother with love.

ACKNOWLEDGEMENTS

First of all, I would like to express my gratefulness to my advisors, Dr. Nathan Fisher and Dr. Grosu for their advice, support and help. Their invaluable help of constructive comments and suggestions throughout my study and thesis works have contributed to the success of this research.

I am also very grateful to Dr. Weisong Shi and Dr. Zaki Malik for serving on my thesis committee. I also would like to thank all my colleagues in CoPaRTS group and LAST group. A special thanks to Masud Ahmad and Pradeep Hettiarachchi for their support and help during my study in CoPaRTS group. I also acknowledge the financial support of the National Science Foundation (Grant No. CNS-116787).

Last but not least, I deeply acknowledge the love and support of my father, my mother and all my brothers and sisters.

TABLE OF CONTENTS

Dedication ii

Acknowledgements iii

List of Tables vi

List of Figures vii

Chapter 1 INTRODUCTION 1

1.1 Related Work 3

1.2 Our Contributions 5

1.3 Organization 6

Chapter 2 MODEL 7

2.1 Implicit Deadline Sporadic Task Model 7

2.2 Competitive Allocations 8

2.2.1 Algorithm for Solving EDF-MAXVAL 9

2.3 Algorithm for Solving RM-MAXVAL 11

2.4 Motivating Examples 14

Chapter 3 MECHANISM DESIGN 17

3.1 Mechanism 17

3.2 Exact Mechanisms 18

3.3 Frugality 20

Chapter 4 APPROXIMATION MECHANISMS 25

4.1 Characterization of Truthful Approximation Mechanisms 25

4.2 Approximation Algorithms 28

4.3 Truthful Approximation Mechanisms 28

4.4 Frugality of Approximation Mechanisms 31

4.5 Approximation Mechanisms with Reserve Prices 31

Chapter 5 EXPERIMENTAL RESULTS 34
5.1 Non-Truthful Type Declaration 35
5.1.1 Results for EDF-based Mechanisms 36
5.1.2 Results for RM-based Mechanisms 37
5.2 Payments and Frugality Ratios 39
5.3 Execution Time 41
Chapter 6 CONCLUSION 43
Bibliography 44
ABSTRACT 48
AUTOBIOGRAPHICAL STATEMENT 49

LIST OF TABLES

Table 2.1: Agents' types example 15
Table 2.2: Upper Bound on Total Utilization for RM Feasibility 16
Table 3.1: Example illustrating tightness of upper bound on the frugality ratio for EDF-MAXVAL-VCG 24
Table 3.2: Example illustrating tightness of upper bound on the frugality ratio for RM-MAXVAL-VCG 24
Table 5.1: Agents' true parameters 35
Table 5.2: Different type declarations by Agent 5 37
Table 5.3: The utilities of Agent 4 in each case for EDF 37
Table 5.4: Different type declarations by Agent 4 for RM 39

LIST OF FIGURES

Figure 3.1: Computing payments 23
Figure 5.1: Utilities and Payments for EDF 37
Figure 5.2: Utilities and Payments for RM 38
Figure 5.3: Payments vs. Values, Frugality and Execution Times of EDF 40
Figure 5.4: Payments vs. Values, Frugality and Execution Times of RM 41

CHAPTER 1

INTRODUCTION

In a real-time system, several tasks with specific deadlines can co-execute on a hardware platform. Each task has some characteristics and generates infinite jobs in a very predictable manner. For example in a *sporadic* task system, different jobs are generated in a periodic manner and they have upper bounds upon their worst-case execution requirement and associated deadlines. In *implicit-deadline* sporadic tasks system, the deadline of jobs are equal to the time interval between to consecutive jobs of the task. These tasks should be scheduled such that the timing requirements of all tasks are satisfied. Scheduling implicit-deadline sporadic task systems using static and dynamic priority scheduling is well-understood for the traditional (non-competitive) setting. In this thesis, we consider scheduling implicit-deadline sporadic task systems upon a preemptive single processor platform in the presence of competition using the earliest-deadline-first (EDF) and the rate-monotonic (RM) scheduling algorithms. In the competitive setting, a value parameter is associated with each task that represents how much value is obtained by successful execution of all jobs of that task. Under the assumption that the processor cannot feasibly satisfy the temporal requirements of all the tasks, we are interested in selecting a subset of these tasks so that the resulting subset is feasible and the sum of the values of selected tasks is maximized.

For EDF scheduling, this problem is equivalent to the well known 0-1 knapsack problem in which the items' weights are utilizations of the tasks and the items' values are the values of the tasks. There are pseudo-polynomial time and also fully polynomial-time approximation scheme (FPTAS) algorithms that can solve this problem [31]. In the case of RM setting, the aforementioned 0-1 knapsack algorithms are not immediately applicable. We use the utilization-based feasibility test which is a sufficient condition and, we propose a pseudo-polynomial algorithm which optimally solves the problem. We also propose an approximation algorithm for the RM setting.

Using these algorithms to solve the problems are useful when we assume that the true characteristics of each task are known. This assumption is not valid in competitive environments such as cloud computing systems or shared real-time networks in which several agents are competing for processor time upon a shared computational resource. The agents may lie about their true task requirements and their value in order to maximize their own obtained benefits from execution.

We consider the scheduling of sporadic tasks in a competitive environment in which each task is owned by a separate agent. Each agent knows the characteristics of her own task and reports a utilization and a value to the processor owner. Since each agent is self-interested, she may report a utilization and value different from the true ones if she knows that by doing this, her task will be selected to run on the processor. By considering these self-interested agents, the problem is moved from the area of algorithm design to that of mechanism design [21].

Mechanism design is the art of designing rules in a competitive environment to achieve specific properties such as *truthfulness* and *efficiency*. The truthfulness property ensures that the agents will always tell the truth and the efficiency property will provide a maximized system-wide objective. Nisan and Ronen [27] were the first to consider the use of mechanism design in computational settings. In recent years, mechanism design has found many important applications in computer science such as network routing, load balancing, auctioning and internet advertisements. Mechanism design has had a spectacular commercial success. For example, Google and Yahoo! employ mechanism design for internet advertisement auctions and their revenues from these auctions in 2005 were over $6 billion and $2.6 billion respectively [13].

Given the significant impact of mechanism design in a large spectrum of computer science domains, it behooves us to understand the effects of competition on the design of real-time open environments in which several independently-developed real-time applications may share the same computational platform. Furthermore, future real-time and cyber-physical systems are likely to be open [1]. A lack of understanding of the effects of competition on the temporal correctness of open systems will ultimately lead to an inefficient allocation of resources.

Unfortunately, as we will see in the related work section, there is only one paper that addresses real-time scheduling under competition. Furthermore, this prior paper focuses on online scheduling of aperiodic jobs and not traditional hard-real-time recurring tasks. As an initial starting point for our exploration of competitive real-time systems, we consider the simple environment of implicit-deadline sporadic tasks to be scheduled based on the EDF and RM algorithms. A mechanism will take the task characteristics from each agent and decide which agents obtain the processor. The mechanism also determines the amount that should be paid by the agents who obtained the processor. We are interested in designing mechanisms that give incentives to the agents to report the true characteristics of their tasks and thus, guarantee an efficient processor allocation.

1.1 Related Work

Sporadic real-time task scheduling upon a uniprocessor in non-competitive settings has been studied extensively [20, 24]. For implicit-deadline sporadic tasks systems, Lehoczky et al. [19] presented an exact feasibility test for static-priority scheduling algorithms. Liu and Layland [20] presented a sufficient schedubility test for implicit-deadline sporadic task systems based on rate-monotonic scheduling algorithm.

In a hard real-time system, only tasks that meet their deadlines are considered to be successful. If it is not possible to guarantee the successful completion of all the tasks, the goal is typically to optimize a performance metric. A common metric is to associate a value with each task and quantify the "goodness" of an algorithm by the accumulated values of successful tasks [7, 6, 10]. Aydin et al. [5] studied reward-based scheduling for periodic tasks in which there is a reward associated with each task's execution. Each task is composed of a mandatory and an optional part. The mandatory part must meet the task's deadline, while a non-decreasing reward function is associated with the execution of the optional part. The goal is to find a schedule that maximizes the weighted average reward. All of these prior works assume that the task characteristics are publicly known, and none of them considers a competitive setting, in which the task's characteristics are private to the agents and the agents compete for resources.

Nisan and Ronen [27] introduced the technique of algorithmic mechanism design for computational problems in a competitive setting. They addressed the problem of minimization of the make-span of tasks on parallel machines by designing a truthful approximation mechanism for the problem. The field of mechanism design has been applied to several computer science problems such as routing [14] and multicast transmission [15]. Aggarwal et al. [3] studied knapsack auctions for selling advertisements on Internet search engines in which the size of objects are publicly known. Their work is related to our study, in both cases the underlying optimization problem is the knapsack problem.

There are only a few works we are aware of that apply the field of game theory and mechanism design to real-time systems. Sheikh et al. [29] used a game-theoretic computational technique to solve the problem of scheduling strictly periodic tasks in a non-competitive environment. Porter [28] studied the problem of online real-time scheduling of jobs on a single processor in a competitive environment. In this work, the private type of the agents consists of release time, job length, deadline, and value. However, none of these prior works on scheduling considers traditional recurring tasks (e.g., sporadic or periodic tasks) which are commonly found in real-time applications. The goal of our thesis is to investigate competitive scheduling for recurring tasks by introducing, developing, and analyzing techniques of mechanism design for scheduling sporadic tasks on a shared single processor platform.

1.2 Our Contributions

In this thesis, we employ the field of mechanism design for assigning a single processor to real-time sporadic tasks. These are our contributions in this work:

- We employ the field of mechanism design for scheduling implicit-deadline sporadic task systems upon a single processor in a competitive setting. To the best of our knowledge, this is the first work that considers scheduling recurring tasks in a competitive environment.

- We design truthful exact mechanisms based on the Vickrey-Clarke-Groves (VCG) mechanism [32, 11, 16] that allocates the single processor to a subset of participating agents based on EDF and RM scheduling algorithms. These mechanisms use dynamic programming algorithms to optimally select the agents who obtain the processor.

- We evaluate multiple definitions of *frugality* and determine the most suitable definition for real-time scheduling of sporadic tasks. The *frugality ratio* of a mechanism measures the amount of payment made by the agents compared to the agent's values.

- Since the allocation algorithms for the truthful exact mechanisms are computationally intractable, we provide truthful approximation mechanisms which use fully polynomial-time approximation scheme algorithms to find a near-optimal allocation and derive their frugality ratios.

- The total payments by the agents can be less than the cost of operating the system; therefore, we design truthful mechanisms with reserve prices which guarantee a minimum profit for the processor owner.

- We perform simulations to investigate the effects of non-truthful behavior of agents, comparing payments to the reported values, determining the frugality ratios of the mechanisms, and comparing the execution times of the mechanisms.

The results presented in this thesis were published in Proceedings of the 24th Euromicro Conference on Real-Time Systems (ECRTS12) [22]. An extended version of this paper is under review for IEEE Transactions on Computers [23].

1.3 Organization

The thesis is organized as follows. In Chapter 2, we discuss the problem of implicit-deadline sporadic task scheduling in both non-competitive and competitive environments and using dynamic and static-priority task scheduling algorithms. In Chapter 3, we review the basic concepts of mechanism design and introduce truthful exact mechanisms to solve the problems in competitive environments. We also discuss the frugality of the mechanisms. In Chapter 4, we present truthful approximation mechanisms and derive bounds on their frugality. In Chapter 5, we present and discuss experimental results.

CHAPTER 2

MODEL

In this chapter, we discuss the original problem of scheduling implicit-deadline sporadic task systems on a single processor and present previous results. We also define the competitive version of the problem, where there is a value associated with each task. We give examples that illustrate the need for mechanism design in competitive settings.

2.1 Implicit Deadline Sporadic Task Model

In a sporadic task system $S = \{T_i | i = 1, \ldots, n\}$, each task $T_i = (e_i, d_i, p_i)$ is characterized by three parameters: (i) the worst-case execution time of each job, e_i; (ii) the relative deadline, d_i; and (iii) the minimum separation between successive job releases of the task, p_i. The *utilization* u_i of task T_i, is defined as the ratio of the execution time of the task to its period, $u_i = e_i/p_i$. A sporadic task system is a finite collection of sporadic tasks. The utilization of a sporadic task system S is defined as $U(S) = \sum_{T_i \in S} u_i$.

We consider the case of implicit-deadline sporadic task systems in which the relative deadline of each job is equal to the minimum separation between successive jobs of the task, ($d_i = p_i$) for all tasks. We consider both dynamic-priority and static-priority task scheduling. In the dynamic-priority scheduling of sporadic tasks, all the jobs generated by a task may have different priorities, but in static priority-scheduling they the same priority.

For the dynamic-priority scheduling, it has been shown that the earliest deadline first scheduling algorithm (EDF) is an optimal algorithm for scheduling sporadic tasks in a preemptive environment [20, 24]. In other words, if it is possible to preemptively schedule a task system such that all the jobs meet their deadlines, then the EDF algorithm for this task system will meet all deadlines as well. A necessary and sufficient condition [24] for any implicit-deadline system S to be feasible upon a uniprocessor is $U(S) \leq 1$.

For implicit-deadline sporadic task systems, the *rate-monotonic* (RM) algorithm, is an optimal priority assignment in the static-priority scheduling. The RM algorithm assigns priorities to the tasks in inverse proportion to their period parameters, i.e., task T_i has higher priority than T_j's priority if $p_i < p_j$. It has been shown that ([20]), an implicit-deadline sporadic task system S is static-priority feasible if

$$U(S) \leq n(2^{\frac{1}{n}} - 1), \tag{2.1}$$

where $U(S)$ is its total utilization (the sum of utilizations of all tasks) and n is the number of tasks (i.e., $n = |S|$). This is a sufficient condition but not necessary condition for feasibility. In this thesis, we use this test to check the feasibility of a sporadic task system.

2.2 Competitive Allocations

Consider an environment where each task T_i is owned by Agent i and each agent competes for allocation of the processor to her task. Each agent declares a value she wishes to pay if her task is selected to run on the processor. Each Agent i is characterized by a *type* $\theta_i = (u_i, v_i)$, where u_i is the utilization required to execute the task and v_i is the value derived by the agent from executing the task. Let $N = \{1, 2, \ldots, n\}$ be the set of all agents. A set of agents is *feasible* if it is possible to schedule their tasks such that they meet their deadlines, i.e., for dynamic-priority scheduling, the sum of utilizations of selected agents should be less than or equal to 1 and for static-priority scheduling the sum of utilizations of selected agents is at most $k(2^{\frac{1}{k}} - 1)$, where k is the number of selected agents. Our objective is to allocate the processor to a feasible subset $\mathcal{O} \subseteq N$ so that the sum of the values of the agents in $\mathcal{O}$ is maximized. This is a common objective in economics and is also referred to as *social welfare*. We formulate two problems of maximizing the social welfare, one for EDF and one for RM as follows.

EDF-MAXVAL Problem: Given agent types $\theta_i = (u_i, v_i), i = 1, \ldots, n$

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i=1}^{n} v_i x_i \\ \text{subject to:} \quad & \textstyle\sum_{i=1}^{n} u_i x_i \leq 1 \\ & x_i \in \{0, 1\} \end{aligned} \tag{2.2}$$

RM-MAXVAL Problem: Given agent types $\theta_i = (u_i, v_i), i = 1, \ldots, n$

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i=1}^{n} v_i x_i \\ \text{subject to:} \quad & \textstyle\sum_{i=1}^{n} u_i x_i \leq (2^{1/\sum_{i=1}^{n} x_i} - 1) \sum_{i=1}^{n} x_i \\ & x_i \in \{0, 1\} \end{aligned} \tag{2.3}$$

where $x_i = 1$, when Agent i is selected, and $x_i = 0$, otherwise. The constraints ensure that the set of selected tasks is feasible. We consider that the values $(v_i, i = 1, .., n)$ are integers. The EDF-MAXVAL and RM-MAXVAL are natural formulations of the problem of determining how to allocate a shared processor and still guarantee that admitted tasks can meet their deadlines. Please note that EDF-MAXVAL and RM-MAXVAL use, in their formulation, only u_i and v_i for each task T_i. We may ignore, for now the particular values of e_i, p_i, and d_i. For any agent that is selected, we may successfully accommodate, via EDF or RM, any p_i and e_i such that $u_i = e_i/p_i$ (given the implicit-deadline assumption). In the following sections, we present two algorithms for solving the EDF-MAXVAL and RM-MAXVAL problems.

2.2.1 Algorithm for Solving EDF-MAXVAL

The EDF-MAXVAL problem is the standard 0-1 knapsack problem and is NP-hard; however, the problem does admit a pseudo-polynomial-time algorithm based on dynamic programming [31]. We restate the dynamic programming approach in the following. Let V be the maximum of the values of all tasks, i.e., $V = \max_{i=1}^{n} v_i$. It is trivial that an

Algorithm 1 EDF-MAXVAL-DP: Allocation Algorithm

```
1:  input: $u_1, \dots, u_n$ and $v_1, \dots, v_n$.
2:  $V = \max_i v_i$;
3:  for $j = 1$ to $nV$ do
4:     if $v_1 = j$ then
5:        $U(1, j) = u_1$
6:     else
7:        $U(1, j) = \infty$
8:     end if
9:  end for
10: for $i = 1$ to $n - 1$ do
11:    for $j = 1$ to $nV$ do
12:       if $v_{i+1} \leq j$ then
13:          $U(i+1, j) = \min\{U(i, j), u_{i+1} + U(i, j - v_{i+1})\}$
14:       else
15:          $U(i+1, j) = U(i, j)$
16:       end if
17:    end for
18: end for
19: $opt = \max\{v | U(n, v) \leq 1\}$
20: $\mathcal{O}$ = the set of selected agents by looking backward at $U(i, j)$.
21: output $(opt, \mathcal{O})$
```

upper bound on the maximum value that can be achieved by any solution is nV. For each $i \in \{1, 2, \dots, n\}$ and $v \in \{1, 2, \dots, nV\}$, let $S_{i,v}$ denote a subset of tasks $\{1, 2, \dots, i\}$ whose total value is exactly v and whose total utilization is minimized. Let $U(i, v)$ denote the utilization of the set $S_{i,v}$ (it is ∞ if no such set exists). Clearly, $U(1, v)$ is known for every $v \in \{1, \dots, nV\}$. The following recurrence computes all values $U(i, v)$ in $O(n^2V)$ time:

$$U(i+1, v) = \begin{cases} \min\{U(i, v), u_{i+1} + U(i, v - v_{i+1})\} & \text{if } v_{i+1} \leq v, \\ U(i, v) & \text{otherwise.} \end{cases} \tag{2.4}$$

The maximum value achievable by a set of tasks with total utilization bounded by 1 is $\max\{v | U(n, v) \leq 1\}$. The dynamic programming algorithm EDF-MAXVAL-DP is given in Algorithm 1. We are computing the optimum aggregate value in lines 2-19. In Line 20, we obtain the selected tasks. We can do this by just looking backward at $U(i,j)$ matrix. Let $opt = \max\{v | U(n,v) \leq 1\}$. If $U(n, opt) = U(n-1, opt)$ then we did not select the n-th item, so we just recursively work backwards from $U(n-1, opt)$. Otherwise, we select that item, output the n-th task and recursively work backwards from $U(n-1, opt - v_n)$. EDF-MAXVAL-DP determines the solution in $O(n^2 V)$ time, and thus, it is a pseudo-polynomial algorithm for EDF-MAXVAL.

2.3 Algorithm for Solving RM-MAXVAL

Now we propose a new dynamic programming algorithm for finding a feasible subset which has the maximum total aggregate value for the RM settings. We use the utilization bound feasibility test [20], i.e., a subset $S \subseteq N$ is feasible if

$$U(S) \leq |S|(2^{1/|S|} - 1)$$

The dynamic programming approach is as follow. Let V be the maximum of the values of all tasks, i.e., $V = \max_{i=1}^{n} v_i$. It is trivial that an upper bound on the maximum value that can be achieved by any solution is nV. For each $i \in \{1, 2, \ldots, n\}$ and $k \in \{0, 1, \ldots, n\}$ and $v \in \{1, 2, \ldots, nV\}$, let $S_{i,k,v}$ denote a subset of tasks $\{1, 2, \ldots, i\}$ whose total value is exactly v and contains exactly k tasks, and whose total utilization is minimized. Let $U(i,k,v)$ denote the utilization of the set $S_{i,k,v}$ (it is ∞ if no such set exists). Clearly, $U(1,k,v)$ is known for every $k \in \{0, \ldots, n\}$ and $v \in \{1, \ldots, nV\}$ and is

$$U(1,k,v) = \begin{cases} 0 & \text{if } k = 0, \\ u_1 & \text{if } v = v_1 \text{and } k = 1, \\ \infty & \text{otherwise.} \end{cases} \tag{2.5}$$

The following recurrence computes all values $U(i,k,v)$ in $O(n^3V)$ time:

$$U(i+1,k,v) = \begin{cases} \min\{U(i,k,v), u_{i+1} + U(i,k-1,v-v_{i+1})\} & \text{if } v_{i+1} \leq v, \\ U(i,k,v) & \text{otherwise.} \end{cases} \tag{2.6}$$

The maximum value achievable by a set of tasks is $\max\{v|U(n,k,v) \leq k(2^{1/k}-1), 1 \leq k \leq n\}$, where $S_{n,k,v}$ is the task set associated with $U(n,k,v)$. The dynamic programming algorithm RM-MAXVAL-RMDP is given in Algorithm 2. We are computing the optimum aggregate value in lines 2-24. In Line 25, we obtain the selected tasks. We can do this by just looking backward at $U(i,k,j)$ matrix as discussed in the previous section.

Now we prove that the algorithm is correct by using the following lemma:

Lemma 2.3.1 *Let $\mathcal{A}_i$ be the set of agents $\{1,2,...,i\}$. $U(i,k,v)$ is the minimum total utilization that can be obtained by selecting exactly k agents in $\mathcal{A}_i$ with a total value equal to v.*

Proof The proof is by induction on i. It is easy to check that the statement is correct for $i = 1$. For $k = 0$, $U(1,k,v) = 0$. If $v = v_1$, then $U(1,1,v) = u_1$ and $U(1,k,v) = \infty$, for $k > 1$. If $v \neq v_1$, then $U(1,k,v) = \infty$, for every $k > 0$.

Now, assume that the statement holds for i, i.e., $U(i,k,v)$ is the minimum total utilization that can be obtained by a subset of k agents in $\mathcal{A}_i$ with total value equal to v. We prove that the statement also holds for $i+1$. In other words, we prove that $U(i+1,k,v)$ is the minimum total utilization that can be obtained by a subset of k agents in $\mathcal{A}_{i+1}$ with total value equal to v. There are two possible cases: first, the set with minimum utilization does not contain Agent $i+1$, and, second it contains Agent $i+1$. In the first case, the minimum utilization is equal to $U(i,k,v)$ and since this is the minimum utilization (based on induction hypothesis), thus $U(i+1,k,v)$ is the minimum utilization. In the second case, the minimum utilization is equal to the sum of

Algorithm 2 RM-MAXVAL-DP: Allocation Algorithm

```
 1: input: $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$.
 2: $V = \max_i v_i$;
 3: for $i = 1$ to $n$ do
 4:   for $j = 1$ to $nV$ do
 5:     $U(i, 0, j) = 0$
 6:   end for
 7: end for
 8: for $j = 1$ to $nV$ do
 9:   for $k = 1$ to $n$ do
10:     if $v_1 = j$ then
11:       $U(1, k, j) = u_1$
12:     else
13:       $U(1, k, j) = \infty$
14:     end if
15:   end for
16: end for
17: for $i = 1$ to $n - 1$ do
18:   for $k = 1$ to $n - 1$ do
19:     for $j = 1$ to $nV$ do
20:       if $v_{i+1} \leq j$ then
21:         $U(i + 1, k, j) =$
22:         $\min \{U(i, k, j), u_{i+1} + U(i, k - 1, j - v_{i+1})\}$
23:       else
24:         $U(i + 1, k, j) = U(i, k, j)$
25:       end if
26:     end for
27:   end for
28: end for
29: $opt = \max \{v | U(n, k, v) \leq k(2^{1/k} - 1), 1 \leq k \leq n\}$
30: $\mathcal{O}$ = the set of selected agents by looking backward at $U(i, k, j)$.
31: output $(opt, \mathcal{O})$
```

the utilization of agent $i+1$ and the minimum utilization that can be obtained by using $k-1$ agents of $\mathcal{A}_i$ with total value equal to $v - v_{i+1}$. Since $U(i, k-1, v-v_{i+1})$ is the minimum utilization of such subset (based on induction hypothesis), then $U(i+1, k, v)$ is the minimum utilization and the proof is complete.

Theorem 2.3.2 *Algorithm RM-MAXVAL-DP is correct.*

Proof Suppose that the value obtained by the algorithm is v^*, corresponding to $U(n, k^*, v^*)$. Thus, we have

$$v^* = \max\left\{v | U(n,k,v) \leq k(2^{1/k}-1), 1 \leq k \leq n\right\} \tag{2.7}$$

i.e., for each $v > v^*$, we have $U(n,k,v) > k(2^{1/k}-1)$. By using Lemma 2.3.1, $U(n,k,v)$ is the minimum total utilization that can be obtained by selecting k agents with total value v. Therefore, any subset with total value greater than v^* is infeasible, and v^* is the maximum achievable value.

The time complexity of RM-MAXVAL-DP algorithm is $O(n^3V)$ which is pseudo-polynomial time.

2.4 Motivating Examples

In order to compute the optimal solution for the EDF-MAXVAL and RM-MAXVAL problems, we rely upon the agents to report their true types. However, we now give examples to show how a lying agent can affect the outcome of the algorithms. We first provide an example for EDF-MAXVAL.

EDF Motivating Example Consider a competitive environment with five agents. The utilizations and values of the tasks owned by these agents are shown in Table 2.1. Since all these tasks cannot be scheduled to execute on a single processor using the EDF scheduling algorithm, we want to assign the processor to the agents such that we obtain the maximum social welfare. If each agent is truthful, EDF-MAXVAL-DP assigns the

Table 2.1: Agents' types example

Agent	1	2	3	4	5
u_i	0.1	0.2	0.4	0.6	0.7
v_i	2	7	8	9	11

processor to Agents 1, 2 and 5, which results in a welfare of 20. However, if Agent 4 lies and reports a valuation of 20 and everyone else report their true values, Agents 3 and 4 would be selected, giving a suboptimal social welfare of 17. Now, assume that all agents report their true types except Agent 5 who lies about her required utilization and declares 0.8 instead of her true utilization of 0.7. In this case, Agents 2 and 5 would be selected, resulting in a suboptimal social welfare of 18, and Agent 1 would not be selected anymore.

RM Motivating Example Consider the competitive environment with the same agents shown in Table 2.1. Again all the tasks can not be scheduled on a single processor based on the RM algorithm. According to the RM utilization feasibility test, a set of agents is feasible, if the sum of utilizations of these tasks is at most $U_{RM}(n) = n(2^{1/n} - 1)$, where n is the number of tasks. The values of $U_{RM}(n)$ for $n = 1, ..., 5$ are shown in Table 2.2. If each agent is truthful, RM-MAXVAL-DP assigns the processor to Agents 1, 2 and 3, which results in a welfare of 17 and a total utilization of 0.7 which is less than $U_{RM}(3) = 0.780$. As in the previous case, let assume that Agent 4 declares 20 as her value instead of her true value. In this case, Agents 2 and 4 would be selected and the suboptimal social welfare is 16. Now, assume that Agent 2 lies about her required utilization and declares 0.3. In this case, Agents 2 and 3 would be selected, resulting in a suboptimal social welfare of 16 and Agent 1 would not be selected anymore.

The allocations obtained above by non-truthful declarations, are inefficient and the processor is not allocated to the agents that value the execution the most. We are interested in ways to control the competition so that it is always in an agents' interest to

Table 2.2: Upper Bound on Total Utilization for RM Feasibility

n	1	2	3	4	5
$U_{RM}(n)$	1	0.828	0.780	0.757	0.743

declare their true types and achieve the optimal system welfare. In the following sections we will design such mechanisms that give incentives to the agents to be truthful.

CHAPTER 3

MECHANISM DESIGN

The field of mechanism design deals with algorithmic problems in a competitive environment. In this chapter, we present the basic concepts of mechanism design and introduce VCG-based mechanisms for the EDF-MAXVAL and RM-MAXVAL problems.

3.1 Mechanism

Mechanism. A mechanism is composed of an *allocation algorithm* A and a *payment scheme* π. The allocation algorithm determines which agents obtain the processor and the payment scheme calculates the payment of each agent.

We consider the problem of allocating processor time to a set of n agents. Each agent owns a sporadic task and declares a *type* which characterizes the utilization of her task and the value derived from running the task on the processor. Since the agent may strategically declare a different type from her true type, we denote Agent i's declared utilization and value by $\hat{u}_i$ and $\hat{v}_i$, respectively, and denote the true utilization and value by u_i and v_i. We denote the declared type of Agent i by $\hat{\theta}_i$ and the true type by θ_i.

A mechanism takes, as input, all declared types from agents and computes an allocation. The mechanism gives incentives to the agents to reveal their true types by charging them some payment. The allocation and payments depend on the agent declarations $\hat{\boldsymbol{\theta}} = (\hat{\theta}_1, \ldots, \hat{\theta}_n)$.

The allocation algorithm is given as input the vector $\hat{\boldsymbol{\theta}}$ of agents' types, and outputs a subset $A(\hat{\boldsymbol{\theta}}) \subseteq N$ of *winning agents*, where N is the set of participating agents. Thus, Agent i wins if $i \in A(\hat{\boldsymbol{\theta}})$. The *social welfare* obtained by the algorithm is given by $\sum_{i \in A(\hat{\boldsymbol{\theta}})} v_i$. The allocation algorithm attempts to maximize the social welfare.

The strategy of an agent is represented by her declared type and her goal is to maximize her utility. We define Agent i's utility as $\mu_i = v_i - \pi_i$, where π_i is the amount Agent i is required to pay for having the task executed on the processor. If Agent i is not selected to obtain the processor, then $\mu_i = 0$. Agent i may strategically prefer to declare a type different from her true type in order to increase her utility. We are interested in a truthful mechanism where it is always in each agent's best interest to declare her true type.

Truthful Mechanism. A mechanism (A, π) is called *truthful* (or *incentive compatible*) if for every declaration of the other agents $\hat{\boldsymbol{\theta}}_{-i}$ (i.e., $\hat{\boldsymbol{\theta}}_{-i} = (\hat{\theta}_1, \ldots, \hat{\theta}_{i-1}, \hat{\theta}_{i+1}, \ldots, \hat{\theta}_n)$), and every declaration $\hat{\theta}'_i$ of Agent i, we have: $\mu_i \geq \mu'_i$, where μ_i and μ'_i are the utilities obtained by Agent i when declaring θ_i and $\hat{\theta}'_i$, respectively. This means that truthful revelation is a dominant strategy; that is, agents maximize their utilities by reporting their true types.

In the rest of thesis, we assume that the agents always report a utilization equal to or greater than the actual utilization required by their tasks (i.e., $\hat{u}_i \geq u_i, i = 1, \ldots, n$). The reason is that if the agent reports a utilization less than the actual utilization of her task and wins the competition, her task cannot be executed on the processor, since it requires higher utilization and will potentially miss a deadline. We assume that the system employs a mechanism for temporally isolating tasks during execution and enforcing a winning agent to execute only her requested utilization. Such mechanism is described in [2]. In this mechanism, if a task needs more than its reported utilization, it may slow down if it jeopardizes the schedulability of the other tasks.

3.2 Exact Mechanisms

Nisan and Ronen [27] showed that the truthfulness of a mechanism can be guaranteed by standard Vickrey-Clarke-Groves (VCG)-based mechanisms [11, 16, 32], if the mechanism is able to compute the optimal solution.

VCG Mechanism. A mechanism composed of allocation algorithm A and payment algorithm π is called a VCG mechanism if

- $A(\hat{\boldsymbol{\theta}})$ is the allocation that maximizes the social welfare (i.e., $\sum_{i\in A(\hat{\boldsymbol{\theta}})} \hat{v}_i$), and
- $\pi_i = \sum_{j\in A(\hat{\boldsymbol{\theta}}_{-i})} \hat{v}_j - \sum_{j\in A(\hat{\boldsymbol{\theta}}), j\neq i} \hat{v}_j$

We define the VCG-based mechanism that solves the EDF-MAXVAL problem as follows.

EDF-MAXVAL-VCG Mechanism. The EDF-MAXVAL-VCG mechanism consists of the allocation algorithm EDF-MAXVAL-DP and the payment defined by:

$$\pi_i^{VCG} = \sum_{j\in A_{EDF}(\hat{\boldsymbol{\theta}}_{-i})} \hat{v}_j - \sum_{j\in A_{EDF}(\hat{\boldsymbol{\theta}}), j\neq i} \hat{v}_j \tag{3.1}$$

where A_{EDF} is the allocation algorithm EDF-MAXVAL-DP.

We also define the VCG-based mechanism that solves the RM-MAXVAL problem as follows.

RM-MAXVAL-VCG Mechanism. The RM-MAXVAL-VCG mechanism consists of the allocation algorithm RM-MAXVAL-DP and the payment defined by:

$$\pi_i^{VCG} = \sum_{j\in A_{RM}(\hat{\boldsymbol{\theta}}_{-i})} \hat{v}_j - \sum_{j\in A_{RM}(\hat{\boldsymbol{\theta}}), j\neq i} \hat{v}_j \tag{3.2}$$

where A_{RM} is the allocation algorithm RM-MAXVAL-DP.

The first term in Equations 3.1 and 3.2, represents the optimal welfare obtained when Agent i is excluded from the competition, and the second term represents the sum of all values in the optimal set except Agent i's value. EDF-MAXVAL-DP and RM-MAXVAL-DP compute the optimum social welfare but they are not polynomially computable. EDF-MAXVAL determines the winning agents in $O(n^2V)$ time and for each wining agent it computes the payment using Equation 3.1 by solving an EDF-MAXVAL problem with $n-1$ agents. Hence, computing the payments needs at most $nO((n-1)^2V) = O(n^3V)$. Using the same analysis, the RM-MAXVAL-VCG mechanism computes the allocations and payments in $O(n^4V)$. Thus, the EDF-MAXVAL-DP and RM-MAXVAL-DP are pseudo-polynomial time mechanisms.

3.3 Frugality

In a truthful mechanism, the payment by an agent is less than her declared value. Agents may have multiple choices and would like to pay lower amounts for obtaining the processor; thus, from the agents' perspective, lower payments are desirable. We measure the total payment made by agents by the *frugality ratio*. In the following, we identify an appropriate definition of frugality ratio and investigate the frugality of the EDF-MAXVAL-VCG and RM-MAXVAL-VCG mechanisms.

The study of frugality in the context of mechanism design was initiated by Archer and Tardos [4]. They investigated the frugality of path auctions in weighted directed graphs and showed that the total payment of any truthful mechanism for path auctions can be a linear factor of the second optimal disjoint path. However, there are other different definitions for frugality in the literature. Talwar [30] defined the frugality ratio of VCG mechanisms for set system problems. They defined the frugality ratio as the worst possible ratio of the payment to the cost of the best rival solution. Karlin et al. [17] argued that a natural choice for the frugality ratio is the overpayment of a mechanism compared to the minimum payment by a non-truthful mechanism; hence, the frugality ratio characterizes the cost of truthfulness. They proposed the Nash Equilibrium [26] as the lower bound for the payments. They proved that the VCG mechanism has a frugality ratio of 1 for *monopoly-free matroid* systems.

The question is: how should we measure the frugality in our competitive real-time setting? A trivial way to define the frugality is to compare the total payments to the mechanism to the sum of the winning agents' declared values. We now argue that this definition results in unstable behavior of the frugality ratio and it is not suitable for characterizing frugality in our setting. In the EDF-MAXVAL-VCG mechanism, if a winning agent raises her value, her payment will not change, thus, a good definition for the frugality ratio should not depend on the declared values of the winning agents. For example, let us assume that the frugality ratio is defined as the ratio of total payments to the sum of the declared values of the winning agents. Consider the problem instance given in Table 2.1. The EDF-MAXVAL-VCG mechanism allocates the processor to agents 1, 2, and 5. The payments of the winning Agents 1, 2, and 5 are 1, 2, and 9, respectively. Thus, the frugality ratio is $(1 + 2 + 9)/(4 + 5 + 11) = 0.6$. Now, if we assume that the declared value of Agent 5 is 100, the payments are still the same, and the frugality ratio is $12/109$ which is less than 0.6. If Agent 5 declares a high value, the frugality ratio will be close to zero. Thus, employing this definition, the frugality ratio can be changed easily since it depends on the values of the winning agents despite the payments remaining unchanged.

From the processor owner's perspective, a drawback of a truthful mechanism is that the payments can be even lower than the total value of the second optimal disjoint set, which is the optimal set of agents obtained from solving the problem while excluding the winning agents from the original problem. Thus, comparing the total payment to the second disjoint optimum is a reasonable way to evaluate the frugality of the mechanism. Let OPT'_{dis} be the sum of the values of the agents in the second disjoint optimal set. Recall that we assume that the set of tasks cannot be feasibly scheduled; thus OPT'_{dis} is well-defined. We use the definition of Talwar [30] who defined the frugality ratio as the total payments divided by the second disjoint optimum value, i.e., the frugality ratio is

$$\mathfrak{F} = \sum_{i=1}^{n} \frac{\pi_i}{OPT'_{dis}} \tag{3.3}$$

A frugality ratio less than one indicates that the total payments to the mechanism are less than the social welfare the mechanism could get by selecting the second disjoint optimal set of agents. A frugality ratio greater than one indicates that the mechanism receives more payment than the total value of the resource according to the second disjoint optimal set of agents.

We now compute an upper bound on the frugality ratio of the VCG mechanism for a special class of set systems we call *inclusive set systems.* This class of set systems has the property that every subset of a feasible set is also feasible. A formal definition of inclusive set systems is as follows.

Inclusive Set System. Consider the set system (E, F) where E is the list of elements and $F \subseteq 2^E$ is the set of all feasible subsets of E. (E, F) is *inclusive* if for each feasible set S, all its subsets are also feasible, i.e., for all $S \in F$, $S' \in F$ for all $S' \subseteq S$.

The implicit-deadline sporadic task system for EDF is inclusive, because if a set S is feasible ($U(S) \leq 1$), then each subset $S' \subseteq S$ is also feasible (because $U(S') \leq 1$). The sporadic task system for RM is also inclusive. Suppose that set system S is feasible, i.e., $U(S) \leq |S|(2^{1/|S|} - 1)$. Since $U_{RM}(n) = n(2^{1/n} - 1)$ is a decreasing function, thus for any $S' \subseteq S$ we have $U_{RM}(|S'|) \geq U_{RM}(S)$. This with the fact that $U(S') \leq U(S)$, implies that S' is feasible. Therefore, the implicit deadline sporadic task system for RM is also inclusive.

We prove that the maximum frugality ratio of any VCG mechanism for inclusive set systems is equal to the number of winning agents. Thus, the frugality ratios of EDF-MAXVAL-VCG and RM-MAXVAL-VCG mechanisms are bounded.

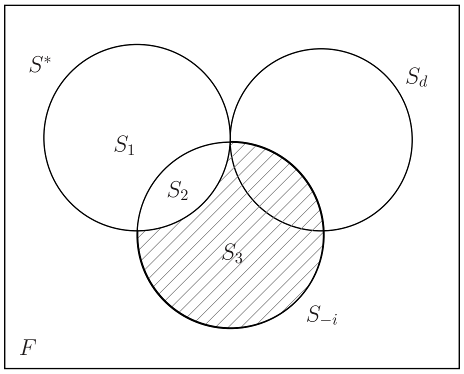


Figure 3.1: Computing payments

Theorem 3.3.1 *The maximum frugality ratio of the VCG mechanism for inclusive set systems is k, where k is the number of winning agents.*

Proof Suppose N is the set of agents and $F \subseteq 2^N$ is the set of feasible sets. Assume that $S^* \subset F$ is a set of winning agents with cardinality k and S_d is the second disjoint optimum set. We denote by $V(S)$, the sum of the values of all agents in set S, i.e., $V(S) = \sum_{i \in S} v_i$.

Now we compute the VCG-payment by Agent $i \in S^*$ and show that it is not greater than $V(S_d)$. Let S_{-i} be the optimum set by excluding Agent i, $S_1 = S^* \setminus S_{-i}$, $S_2 = S^* \cap S_{-i}$ and $S_3 = S_{-i} \setminus S^*$ (Figure 3.1). From Equations 3.1 and 3.2, Agent i's VCG payment is $\pi_i^{VCG} = V(S_{-i}) - V(S^*) + v_i$. We get,

$$\begin{aligned} \pi_i^{VCG} &= V(S_2) + V(S_3) - (V(S_1) + V(S_2)) + v_i \\ &= V(S_3) - (V(S_1) - v_i) \end{aligned} \tag{3.4}$$

Since $i \in S_1$, we have $V(S_1) \geq v_i$, this along with (3.4) implies that $\pi_i^{VCG} \leq V(S_3)$. S_3 is feasible, because it is a subset of feasible set S_{-i}. Since S_3 is disjoint from S^* and S_d is the optimum disjoint feasible set, then we have $V(S_3) \leq V(S_d)$, $\pi_i^{VCG} \leq V(S_d)$, for all $i, 1 \leq i \leq k$. Thus, $\sum_{1 \leq i \leq k} \pi_i^{VCG} \leq kV(S_d)$. Then, $\mathfrak{F} \leq k$.

Now, we show that this bound is tight for EDF-MAXVAL-VCG and RM-MAXVAL-VCG mechanisms by giving examples of a set of agents that achieves this bound. Consider an environment of n agents as displayed in Table 3.1 which should be scheduled based on EDF algorithm. The total payment is $(n-1)V'$ and the sum of the values of the second disjoint optimum set is V'. Hence, $\mathfrak{F} = (n-1)V'/V' = n-1$. Thus, a larger value of n results in a larger frugality ratio. For RM-MAXVAL-VCG, the example shown in Table 3.2, illustrates the tightness of the bound. The total payment is again $(n-1)V'$ and the sum of the values of the second disjoint optimum set is V'. Hence, $\mathfrak{F} = (n-1)V'/V' = n-1$.

Table 3.1: Example illustrating tightness of upper bound on the frugality ratio for EDF-MAXVAL-VCG

Agent	a_1	a_2	$\dots$	a_{n-1}	a_n
$\hat{u}_i$	$\frac{1}{n-1}$	$\frac{1}{n-1}$	$\dots$	$\frac{1}{n-1}$	$\frac{1}{n-1}$
$\hat{v}_i$	V	V	$\dots$	V	$V' < V$
Winner?	Yes	Yes	$\dots$	Yes	No
π_i	V'	V'	$\dots$	V'	0

Table 3.2: Example illustrating tightness of upper bound on the frugality ratio for RM-MAXVAL-VCG

Agent	a_1	a_2	$\dots$	a_{n-1}	a_n
$\hat{u}_i$	$2^{\frac{1}{n-1}}-1$	$2^{\frac{1}{n-1}}-1$	$\dots$	$2^{\frac{1}{n-1}}-1$	1
$\hat{v}_i$	V	V	$\dots$	V	$V' < V$
Winner?	Yes	Yes	$\dots$	Yes	No
π_i	V'	V'	$\dots$	V'	0

The minimum frugality ratio is zero and it is obtained when all payments are zero. If for each winning agent the optimum set by excluding that agent is a subset of the set of winning agents, all payments will be zero.

CHAPTER 4

APPROXIMATION MECHANISMS

As mentioned in the previous chapters, the running times of EDF-MAXVAL-VCG and RM-MAXVAL-VCG are not polynomial in the system input size. In this chapter, we explore techniques for reducing the computational complexity by providing approximate mechanisms instead of exact mechanisms. That is, the mechanisms are not guaranteed to obtain the optimal welfare, but near-optimal welfare. In Section 4.1, we present the mechanism design concepts that will be employed in the design of our approximation mechanisms for solving the EDF-MAXVAL and RM-MAXVAL problems. In Section 4.2, we present approximation algorithms that solve the problems in non-competitive environments. In Section 4.3, we discuss why we cannot use these approximation algorithms for non-competitive environments as building blocks of truthful mechanisms, and design monotonic approximation algorithms suitable for the competitive setting. In Section 4.4, we give a new bound on the frugality ratio of these mechanisms as applied to our scheduling problems.

4.1 Characterization of Truthful Approximation Mechanisms

Monotonicity of the allocation algorithm is a necessary condition for a mechanism to be truthful [18]. An allocation algorithm is monotone when for any winning agent, she also wins by increasing the value or decreasing the utilization while all other agents' types are fixed. Before giving the formal definition of a monotone algorithm, we define the comparison operator for agents' types.

Agent-Type Partial Ordering. Type $\hat{\theta}_i = (\hat{u}_i, \hat{v}_i)$ is greater than type $\hat{\theta}'_i = (\hat{u}'_i, \hat{v}'_i)$ if $\hat{u}_i < \hat{u}'_i$ and $\hat{v}_i > \hat{v}'_i$. It is smaller if $\hat{u}_i > \hat{u}'_i$ and $\hat{v}_i < \hat{v}'_i$. They are equal if $\hat{u}_i = \hat{u}'_i$ and

$\hat{v}_i = \hat{v}'_i$. They are not comparable in any other situation. We denote greater, less and equal operators by $\succ$, $\prec$ and $=$, respectively. We similarly define 'greater than or equal' and 'less than or equal' comparison operators.

Monotonicity. (Mu'alem and Nisan [25]) An allocation algorithm A is *monotone* if, for every Agent i and every $\hat{\boldsymbol{\theta}}_{-i}$, if $\hat{\theta}_i$ is a winning declaration, then every higher declaration $\hat{\theta}'_i \succeq \hat{\theta}_i$ is also winning. In other words, if Agent i wins by declaring $\hat{u}_i$ and $\hat{v}_i$, she also wins by declaring $\hat{u}'_i \leq \hat{u}_i$ and $\hat{v}'_i \geq \hat{v}_i$.

Lemma 4.1.1 (Critical Value) *(Mu'alem and Nisan [25]) Let A be a monotone allocation algorithm, then, for every $\hat{\boldsymbol{\theta}}_{-i}$ there exists a unique value v_i^c such that $\forall \hat{\theta}_i \succeq (\hat{u}_i, v_i^c)$, $\hat{\theta}_i$ is a winning declaration, and $\forall \hat{\theta}_i \prec (\hat{u}_i, v_i^c)$, $\hat{\theta}_i$ is a losing declaration. We refer to this single value as the* critical value *of Agent i.*

Payment. The payment scheme π^A associated with the monotone allocation algorithm A that is based on the critical value is defined as follow:

$$\pi_i^A = \begin{cases} v_i^c & \text{if } i \text{ wins,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

where v_i^c is the critical value of Agent i.

Theorem 4.1.2 (Truthfulness) *(Mu'alem and Nisan [25]) An individually rational mechanism (i.e., a mechanism where agents are guaranteed non-negative utility if they report their true types) is* truthful *if and only if its allocation algorithm is monotone and its payment scheme is based on the critical value.*

Now, we present the definition of *bitonicity* which will be used later to design truthful approximation mechanisms. Simply, an allocation algorithm is *bitonic*, if for any Agent j, the social welfare does not increase with $\hat{\theta}_j$ when Agent j loses, and increase with $\hat{\theta}_j$ when Agent j wins.

Bitonicity. (Mu'alem and Nisan [25]) An allocation algorithm is *bitonic* if it is monotone, and for every agent j and any $\hat{\theta}_{-j}$, the social welfare function is a non-increasing function of $\hat{\theta}_j$ for $v_j < v_j^c$ (Agent j is losing), and a non-decreasing function of $\hat{\theta}_j$ for $v_j \geq v_j^c$ (Agent j is winning).

The bitonicity property helps us to obtain monotone allocation algorithms by combining monotone allocation algorithms. The combination operators can be MAX and the If-then-Else operators. We later use these operators to design approximate truthful mechanisms. One of the most useful operators to combine two allocation algorithms is the MAX operator which try both algorithms and picks the first one.

MAX Operator. (Mu'alem and Nisan [25]) Given two allocation algorithms A_1 and A_2, the algorithm MAX(A_1, A_2)

- Run both algorithms A_1 and A_2.
- if $\omega_{A_1}(\hat{\boldsymbol{\theta}}) \geq \omega_{A_2}(\hat{\boldsymbol{\theta}})$ return $A_1(\hat{\boldsymbol{\theta}})$, else return $A_2(\hat{\boldsymbol{\theta}})$, where ω is the social welfare function.

General this algorithm is not guaranteed to be monotone even if both algorithms A_1 and A_2 be monotone algorithms. the following theorem shows that if A_1 and A_2 be bitonic algorithms, then MAX(A_1, A_2) is bitonic and monotone.

Theorem 4.1.3 *(Mu'alem and Nisan [25]) If A_1 and A_2 be two bitonic algorithms, then $MAX(A_1, A_2)$ is also a bitonic algorithm.*

In the next sections, we design truthful approximation mechanisms for EDF-MAXVAL and RM-MAXVAL problems.

4.2 Approximation Algorithms

Algorithms EDF-MAXVAL-DP and RM-MAXVAL-DP are pseudo-polynomial algorithms. In this section, we present approximation algorithms that solve the EDF-MAXVAL and RM-MAXVAL problems. These algorithms are based on rounding the values of tasks.

The EDF-MAXVAL problem can be approximately solved by using Algorithm 3 [31] which rounds the 'v_i's to admit only a polynomial number of different valuations and then solve optimally by using the EDF-MAXVAL-DP algorithm. The running time of the algorithm is $O(n^2\lfloor V\alpha \rfloor) = O(n^2\lfloor n/\epsilon \rfloor)$ [31], which is polynomial in n and $1/\epsilon$. Thus, the proposed algorithm is an FPTAS algorithm.

Algorithm 3 FPTAS Allocation Algorithm for EDF-MAXVAL

```
1: input: $\hat{u}_1, \dots, \hat{u}_n$ and $\hat{v}_1, \dots, \hat{v}_n$.
2: $\alpha := \frac{n}{\epsilon \hat{v}_{\max}}$;
3: for all $i$ set $v'_i = \lfloor \alpha \cdot \hat{v}_i \rfloor$;
4: return EDF-MAXVAL-DP $(u_1, \dots, u_n; v'_1, \dots, v'_n)$
```

We use the same technique for solving the RM-MAXVAL problem by using the RM-MAXVAL-DP algorithm. The algorithm is shown in Algorithm 4. We can prove that this algorithm is FPTAS similar to the proof provided in [31]. The running time of the algorithm is $O(n^3\lfloor V\alpha \rfloor) = O(n^3\lfloor n/\epsilon \rfloor)$, which is polynomial in n and $1/\epsilon$.

Algorithm 4 FPTAS Allocation Algorithm for RM-MAXVAL

```
1: input: $\hat{u}_1, \dots, \hat{u}_n$ and $\hat{v}_1, \dots, \hat{v}_n$.
2: $\alpha := \frac{n}{\epsilon \hat{v}_{\max}}$;
3: for all $i$ set $v'_i = \lfloor \alpha \cdot \hat{v}_i \rfloor$;
4: return RM-MAXVAL-DP $(u_1, \dots, u_n; v'_1, \dots, v'_n)$
```

4.3 Truthful Approximation Mechanisms

The FPTAS algorithms given in Algorithms 3 and 4 do not satisfy the required monotonicity property, and, thus, cannot be used as allocation algorithms in truthful

Algorithm 5 EDF-MAXVAL-AA: Monotone FPTAS for EDF-MAXVAL

```
1: input: $\hat{u}_1, \ldots, \hat{u}_n$ and $\hat{v}_1, \ldots, \hat{v}_n$.
2: $V := \max_i \hat{v}_i$, $opt = 0$, $\mathcal{O} = \emptyset$
3: for $j = 0$ to $\log((1-\epsilon)^{-1}n) + 1$ do
4:    $k := \lceil \log(V) \rceil - j$;
5:    $\alpha_k := \frac{n}{\epsilon \cdot 2^k}$;
6:    for $i = 1$ to $n$ do
7:       $v'_i := \min\{\hat{v}_i, 2^{k+1}\}$;
8:       $v''_i := \lfloor \alpha_k \cdot v'_i \rfloor$;
9:    end for
10:   $(opt', \mathcal{O}') =$ EDF-MAXVAL-DP $(u_1, \ldots, u_n; v''_1, \ldots, v''_n)$
11:   if $opt' > opt$ then
12:      $\mathcal{O} = \mathcal{O}'$ ;
13:      $opt = opt'$;
14:   end if
15: end for
16: output $(opt, \mathcal{O})$;
```

mechanisms. They are not monotone because the rounding depends on the highest valuation. Briest et al. [9] proposed general approximation techniques for utilitarian mechanism design. A utilitarian mechanism aims to select an output that maximizes the total welfare. They used the concept of *bitonicity* first introduced in [25]. Given a monotone algorithm A, the property of bitonicity requires that the welfare does not increase with v_i when v_i loses ($v_i < v_i^c$), and it does increase with v_i when v_i wins ($v_i > v_i^c$). Briest et al. [9] showed that the algorithm that finds the maximum welfare over the outputs of a set of bitonic algorithms is monotone. Algorithm 5 directly applies the utilitarian mechanism design technique of Briest et al. [9] to obtain a solution for EDF-MAXVAL problem by finding the maximum over the outputs of a bitonic algorithm (Lines 5-10), and thus, by using Theorem 4.1.3 it is monotone. The bitonicity of Lines 5-10 can be proved by a similar argument provided in [9]. It is an FPTAS for EDF-MAXVAL and hence it can be used as the allocation algorithm for a truthful approximation mechanism.

A similar algorithm can be provided for RM-MAXVAL problem. The only change needed is in Line 10 which refers to RM-MAXVAL-DP. We refer to this algorithm (shown

Algorithm 6 RM-MAXVAL-AA: Monotone FPTAS for RM-MAXVAL

```
1: input: $\hat{u}_1, \ldots, \hat{u}_n$ and $\hat{v}_1, \ldots, \hat{v}_n$.
2: Same as Lines 2-9 in Algorithm 5
3: $(opt', \mathcal{O}')$ = RM-MAXVAL-DP $(u_1, \ldots, u_n; v''_1, \ldots, v''_n)$
4: Same as Lines 11-16 in Algorithm 5
```

Algorithm 7 PAY: Payment for winning Agent i

```
1: $a = 0; b = v_i$;
2: while $b - a > 1$ do
3:     $v_i^c = (a + b)/2$
4:     if Agent $i$ is winning by declaring $v_i^c$ then
5:         $b = v_i^c$
6:     else
7:         $a = v_i^c$
8:     end if
9: end while
10: $v_i^c = b$;
11: return $v_i^c$
```

in Algorithm 6) as RM-MAXVAL-AA.

The payments are based on the critical types of the winning agents. The payment of winning Agent i is v_i^c, where v_i^c is the critical value of Agent i, if i wins and zero if i loses. Finding the critical value is done by a binary search over values less than the declared value (Algorithm 7).

In the following, we design two truthful approximation mechanisms for solving the EDF-MAXVAL and RM-MAXVAL problems as follows.

EDF-MAXVAL-APROX Mechanism. The EDF-MAXVAL-APROX mechanism consists of the allocation algorithm EDF-MAXVAL-AA and the payment algorithm PAY.

RM-MAXVAL-APROX Mechanism. The RM-MAXVAL-APROX mechanism consists of the allocation algorithm RM-MAXVAL-AA and the payment algorithm PAY.

4.4 Frugality of Approximation Mechanisms

In this section, we provide a bound on the frugality ratios of proposed approximation mechanisms. Briest et al. [9] showed that, if A is a truthful $(1+\epsilon)-$approximation mechanism it holds that

$$\frac{\epsilon}{1+\epsilon}(n+2) \leq \frac{\pi^A}{\pi^{VCG}} \leq 1+\epsilon(n+2) \tag{4.2}$$

where π^A is the FPTAS mechanism total payment and π^{VCG} is the VCG total payment. Using Theorem 3.3.1, we can derive a formula for the upper bound on the frugality ratio for any truthful FPTAS mechanism for any inclusive set system (which includes the set systems associated with the implicit-deadline sporadic tasks based on EDF and RM scheduling algorithms).

Theorem 4.4.1 *The upper bound on frugality ratio for any truthful* $(1+\epsilon)$*-approximation mechanism* A *for inclusive set systems is* $(1+\epsilon(n+2))k$ *where* k *is the number of winning agents.*

Proof Using Equation 4.2, we get $\pi^A < (1+\epsilon(n+2))\pi^{VCG}$. By Theorem 3.3.1, the maximum frugality ratio is equal to k for inclusive set systems, so $\pi^{VCG}/V(S_d)$ is at most k, where $V(S_d)$ is the sum of the values in the second disjoint optimum set. Thus, we have $\pi^{VCG} \leq kV(S_d)$. So $\pi^A < (1+\epsilon(n+2))kV(S_d)$ and $\mathfrak{F} = \pi^A/V(S_d) \leq (1+\epsilon(n+2))k$.

4.5 Approximation Mechanisms with Reserve Prices

As we discussed, the payments calculated by the VCG and the approximate mechanisms, are less than the agents' declared values and sometimes they can be zero. A processing resource owner may introduce reserve prices to ensure that the costs of operating the system are recovered (e.g., energy costs to run the processor) and that a certain minimum profit margin is achieved. In order to guarantee a minimum profit, we can define a reserve price per utility that is the lower bound on the sale price of the processor for 100 percent of utilization. Let the reserve price for using the full utilization of the processor be C. The reported value of Agent i should be at least $\hat{u}_i C$, i.e., $\hat{v}_i \geq \hat{u}_i C$, where $(\hat{u}_i, \hat{v}_i)$ is the declared type of Agent i. We define the reserve price mechanisms for EDF-MAXVAL and RM-MAXVAL problems as follows.

EDF-MAXVAL-APROX-R. The EDF-MAXVAL-APROX-R mechanism consists of the allocation algorithm EDF-MAXVAL-AA and the payment defined by:

$$\pi_i^R = \max\{v_i^c, u_i * C\} \qquad i = 1, \ldots, n \tag{4.3}$$

RM-MAXVAL-APROX-R. The RM-MAXVAL-APROX-R mechanism consists of the allocation algorithm RM-MAXVAL-AA and the payment defined by:

$$\pi_i^R = \max\{v_i^c, u_i * C\} \qquad i = 1, \ldots, n \tag{4.4}$$

where v_i^c is the critical value computed by PAY (Algorithm 4) and C is the reserve price.

We prove that the mechanisms with reserve prices are truthful.

Theorem 4.5.1 *The EDF-MAXVAL-APROX-R and RM-MAXVAL-APROX-R mechanisms are truthful.*

Proof Consider that Agent i is declaring a non-truthful type $\hat{\theta}_i = (\hat{u}_i, \hat{v}_i) \neq \theta_i = (u_i, v_i)$. Let the utilities of Agent i by truthful and non-truthful type declarations be μ_i and $\hat{\mu}_i$. We consider the following possible cases and show that $\hat{\mu}_i \leq \mu_i$ in all cases.

1. Agent i wins by both declaring $\hat{\theta}_i$ and θ_i. Since the payments are independent from agent's declaration, the utilities in both cases are equal.

2. Agent i loses by both declaring $\hat{\theta}_i$ and θ_i. The utilities are zero in both cases.

3. Agent i loses by declaring $\hat{\theta}_i$ and wins by declaring θ_i. By the monotonicity property

of the allocation algorithm, we have $\hat{v}_i < v_i^c \leq v_i$. This with the fact that $v_i \geq u_i * C$ implies that $\pi_i^R = \max\{v_i^c, u_i * C\} \leq v_i$ and hence $\mu_i = v_i - \pi_i^R \geq 0 = \hat{\mu}_i$.

4. Agent i wins by declaring $\hat{\theta}_i$ and loses by declaring θ_i. By the monotonicity property of the allocation algorithm, we have $\hat{v}_i \geq v_i^c > v_i$. This with $\pi_i^R \geq v_i^c$ implies that $\pi_i^R > v_i$, thus, $\hat{\mu}_i = v_i - \pi_i^R < 0 = \mu_i$.

We showed that truthful declaration is a dominant strategy and hence, the mechanism is truthful.

Since each agent is at least paying C times her declared utilization, the minimum total payment by all winning agents is C times the sum of the utilizations. Hence, the lower bound on the frugality ratio of the approximation mechanism with reserve prices is $C \cdot \min_{i \in \{1..n\}} \hat{u}_i$. The upper bound on the frugality ratio is the same as that of the mechanism without reserve price.

CHAPTER 5

EXPERIMENTAL RESULTS

We perform a set of experiments to investigate the effects of non-truthful type declarations, comparing payments to the reported values, determining the frugality ratios of the mechanisms and also evaluating the execution time of the mechanisms. In Section 5.1, we investigate the effect of non-truthful value declaration by an agent and show how this affects the utility of her and other agents. In Section 5.2, we generate a set of problem instances and investigate the frugality ratios and payments. In Section 5.3, we evaluate the execution times of the exact and approximate mechanisms.

In order to generate the utilizations of the agents we used the UUniFast-Discard method described in [12] with a discard limit equal to half the number of agents. UUniFast-Discard (Algorithm 8) uses the UUniFast method [8] which takes as input the number of agents and parameter U which is the target sum of the utilizations. UUniFast was designed for generating a set of tasks with total utilization of at most 1. Since UUniFast may generate utilizations that are greater then 1 for $U > 1$, UUniFast-Discard was designed to ensure that all generated individualized task utilizations are less than 1. UUniFast-Discard takes as input the number of agents, total utilization and a limit for discarding the number of generated utilizations. When a utilization generated by UUniFast is greater than 1, UUniFast-Discard discards all the generated utilizations and starts from the beginning. If the number of discarded values exceeds the limit, the method reports failing. We set the target utilization U to 5 and the discard limit to half of the number of agents. For the value generation we use a random uniform number generator to generate a vector of integers within $[1, 1000]$. Then, each value is computed by multiplying the corresponding entry in this vector with its associated utilization. Using this approach the values are correlated with the utilizations. For the approximation

Algorithm 8 UUniFast-Discard Algorithm

```
1: input: $n, U$.
2: discarded=0, success = $true$, sumU = $U$
3: for $i = 1$ to $n - 1$ do
4:    nextSumU = sumU $\times$ (rand)$^{\frac{1}{n-i}}$
5:    vectU($i$) = sumU $-$ nextSumU;
6:    if vectU($i$) $> 1$ then
7:       discarded=discarded+1
8:       $i = 1$;
9:       if discarded$> n/2$ then
10:         success= $false$
11:      end if
12:   end if
13:   sumU = nextSumU;
14: end for
15: vectU($n$) = sumU
16: output (vectU, success)
```

Table 5.1: Agents' true parameters

Agent	1	2	3	4	5	6	7	8	9	10
u_i	0.10	0.15	0.20	0.24	0.30	0.12	0.13	0.17	0.22	0.25
v_i	120	400	300	550	600	270	350	125	340	410

mechanism EDF-MAXVAL-APPROX we use $\epsilon = 0.1$. We use the Microsoft Visual C++ environment on an 8-core Intel Core i7 (1.73GHz) machine to generate the problem instances and implement the algorithms.

5.1 Non-Truthful Type Declaration

In this set of experiments, we investigate the effect of reporting *non-truthful* utilizations and values by an agent in EDF-MAXVAL-APROX and RM-MAXVAL-APROX mechanisms. We show how this affects the utility of a lying agent and also those of the other agents. We consider an environment composed of ten agents. The actual utilization and values of these agents are shown in Table 5.1. In the following, we discuss the results for both EDF-based and RM-based mechanism.

5.1.1 Results for EDF-based Mechanisms

We first investigate how a lying agent can affect the utilities and payments of other agents for the EDF-based mechanisms. If all agents declare their true types, Agents 2, 4, 5, 6 and 7 win the competition and their payments will be 300, 420, 530, 150 and 150, respectively. As shown in Table 5.1, the true type of Agent 5 is $\theta_5 = (0.30, 600)$. We consider that Agent 5 is misreporting her type. This leads to six cases as shown in Table 5.2. In Case I, Agent 5 is reporting her true value. In Case II, she is declaring a value greater than her actual value while she is winning the competition. In Case III and Case IV, a non-true value is reported by Agent 5 with the difference that she is winning in Case III but losing in Case IV. In Case V and VI, she is reporting her actual value but reporting utilizations greater than her actual utilization. In Case V, she is winning but in Case VI she is losing.

In Figure 5.1(a), we show the utilities of Agent 5 in all these cases. By reporting the true type, Agent 5 wins the competition and her utility is 70. The utility of Agent 5 is less than or equal to 70 in all the other cases. This is expected, because our mechanism is truthful and the maximum utility is obtained by truth telling. Since Agent 5 loses in Cases IV and VI her utilities in these cases are zero. In all other cases, she wins and obtains the same utility as in Case I, in which she reports her true type.

Now, we investigate how reporting non-true types affects the utilities of the other agents. In Figure 5.1(b), we show the utilities of the Agents 1, 2, 4, 6, 7 and 10 in each of the cases (Since Agents 3 and 9 are losing in all cases, they are not shown in the figure). As we can see, utilities of other agents are changing in most cases. These results show that lying by one agent has a significant effect on the outcome and utility of the other agents. The reason is that, by non-true declaration of Agent 5, the allocations may change and also the agents may have different critical values and hence different payments. For example utilities of Agent 4 in each case is shown in Table 5.3. As we can

Table 5.2: Different type declarations by Agent 5

Case No	$\hat{\theta}_5$	Remark
I	$(0.30, 600)$	True type
II	$(0.30, 700)$	$\hat{v}_5 > v_5, \hat{u}_5 = u_5$
III	$(0.30, 550)$	$\hat{v}_5 < v_5, \hat{u}_5 = u_5$
IV	$(0.30, 400)$	$\hat{v}_5 < v_5, \hat{u}_5 = u_5$
V	$(0.35, 600)$	$\hat{v}_5 = v_5, \hat{u}_5 > u_5$
VI	$(0.40, 600)$	$\hat{v}_5 = v_5, \hat{u}_5 > u_5$

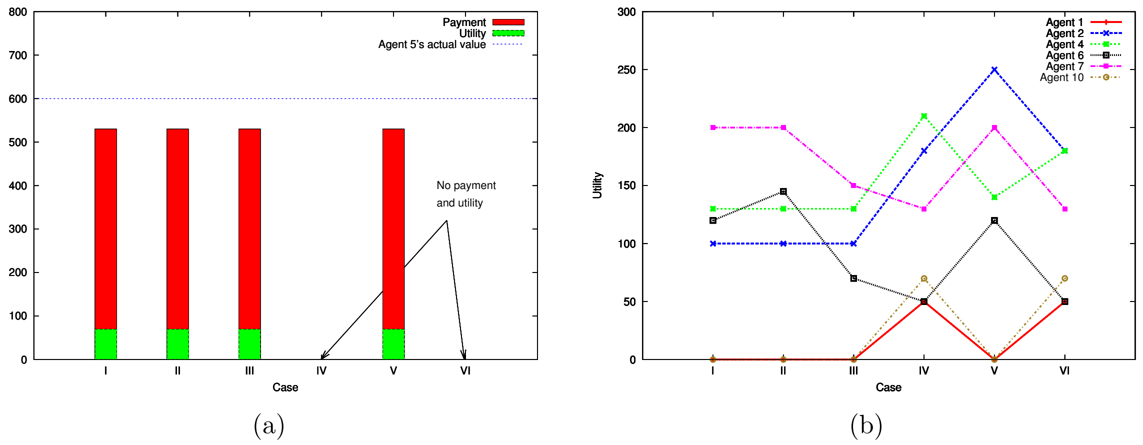


Figure 5.1: Results for EDF-based mechanisms: (a) The utility and payment of Agent 5 in different cases; (b) The utilities of agents in different cases (The agents that are losing in all cases are not shown in the figure).

see the utility of Agent 4, in Case IV in which Agent 5 is declaring a utilization greater than her actual utilization is 210 while in Case I her utility is 130.

Table 5.3: The utilities of Agent 4 in each case for EDF

Case	I	II	III	IV	V	VI
Utility	130	130	130	210	140	180

5.1.2 Results for RM-based Mechanisms

Now, we investigate the effects of non-truthful type declaration on the mechanism for the RM-based mechanisms. If all agents report their true types, Agents 1, 2, 4, 6 and 7 win the competition and their payments will be 70, 290, 480, 220 and 220, respectively. As shown in Table 5.4, the true type of Agent 4 is $\theta_4 = (0.24, 550)$. Consider that

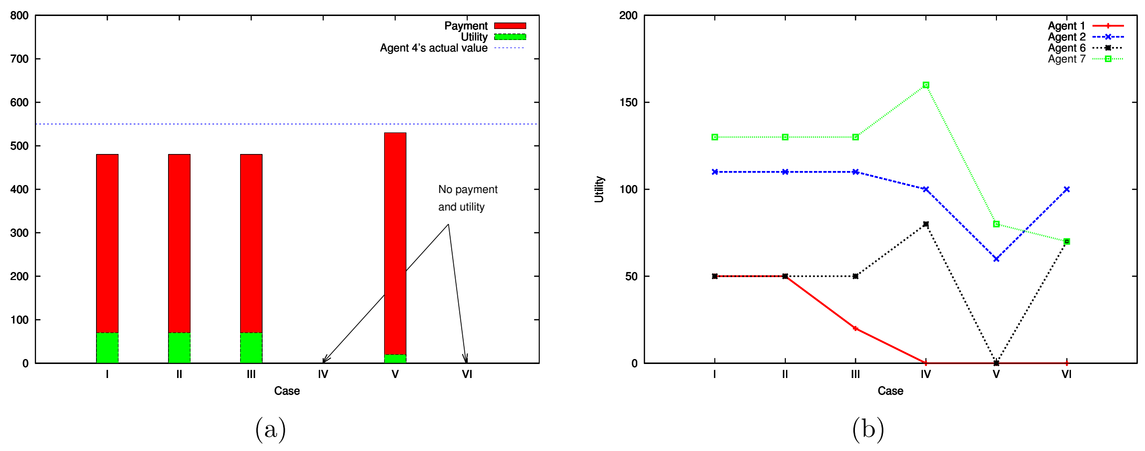


Figure 5.2: Results for RM-based mechanisms: (a) The utility and payment of Agent 4 in different cases; (b) The utilities of agents in different cases (The agents that are losing in all cases are not shown in the figure).

Agent 4 is misreporting her type in different cases which are shown in Table 5.4. In Case I, Agent 4 is reporting her true value. In Case II, she is declaring a value greater than her actual value while she is winning the competition. In Case III and Case IV, a non-true value is reported by Agent 4 with the difference that she is winning in Case III but losing in Case IV. In Case V and VI, she is reporting her actual value but reporting utilizations greater than her actual utilization. In Case V, she is winning but in Case VI she is losing.

In Figure 5.2(a), we show the utilities of Agent 4 in all these cases. By reporting the true type, Agent 4 wins the competition and her utility is 70. As we expected in the EDF example, the utility of Agent 4 is less than or equal to 70 in all the other cases. In Case V, the utility of Agent 4 is 20, which is less than her utility in Case I, that is 70. In this case, Agents 2, 4, 7 and 9 are winning. Since the maximum utilization of a system of 4 tasks to be schedulable by RM is less than the maximum utilization of a task system of 5 tasks (Case I), thus we expect more payments by agents. Therefore we expect that in Case V with 4 winning agents, the utility of Agent 4 is higher than the Case I with 5 winning agents.

Table 5.4: Different type declarations by Agent 4 for RM

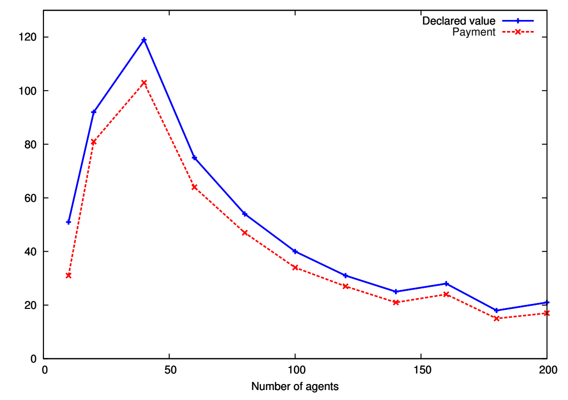
Case No	$\hat{\theta}_4$	Remark
I	$(0.24, 550)$	True type
II	$(0.24, 600)$	$\hat{v}_4 > v_4, \hat{u}_4 = u_4$
III	$(0.24, 500)$	$\hat{v}_4 < v_4, \hat{u}_4 = u_4$
IV	$(0.30, 450)$	$\hat{v}_4 < v_4, \hat{u}_4 = u_4$
V	$(0.25, 550)$	$\hat{v}_4 = v_4, \hat{u}_4 > u_4$
VI	$(0.30, 550)$	$\hat{v}_4 = v_4, \hat{u}_4 > u_4$

Now, we investigate how reporting non-true types affects the utilities of the other agents. In Figure 5.2(b), we show the utilities of the Agents 1, 2, 6 and 7 in each of the cases (Since Agents 3, 5, 8, 9 and 10 are losing in all cases, they are not shown in the figure). As we can see, utilities of other agents are changing in most cases. These results show that lying by one agent has a significant effect on the outcome and utility of the other agents. The reason is that, by non-true declaration of Agent 4, the allocations may change and also the agents may have different critical values and hence different payments. As we discussed in the previous paragraph, the utilities of all agents decreases in Case V.

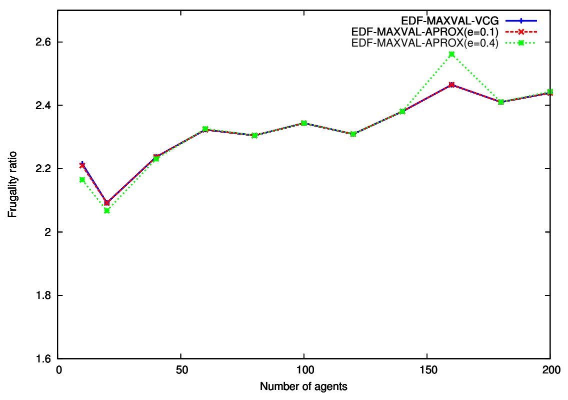
5.2 Payments and Frugality Ratios

Now we perform a set of experiments to illustrate the frugality ratio for both EDF-MAXVAL-VCG and EDF-MAXVAL-APROX mechanisms. We calculate the frugality ratio for problem instances with different number of agents ranging from 10 to 200. For each problem size we generate 100 problem instances and calculate the frugality ratio for each of them.

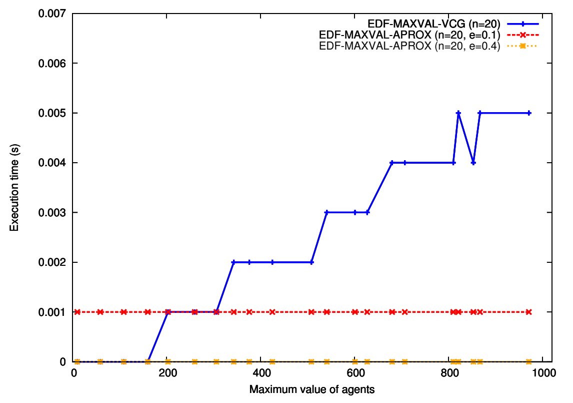
In Figure 5.3(a), we show the average payment of agents comparing to the average declared values for the EDF-MAXVAL-APROX mechanism. The payments are small compared to the declared values for cases with a small numbers of agents, which results in lower revenue of the processor owner. We can see that as the number of agents is increasing, the payments are being closer to the sum of the values and the processor



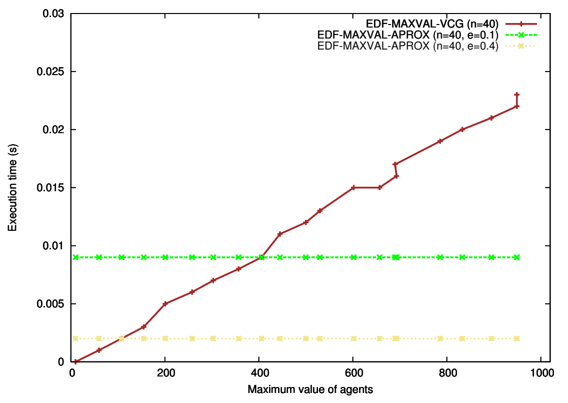
(a)



(b)



(c)



(d)

Figure 5.3: Results for EDF-based mechanisms: (a) Payments vs. Values; (b) Frugality ratio as a function of number of agents; (c) The execution times of EDF-MAXVAL-VCG and EDF-MAXVAL-APROX for $n = 20$; (d) The execution times of EDF-MAXVAL-VCG and EDF-MAXVAL-APROX for $n = 40$.

owner's revenue increases.

In Figure 5.3(b), we show the frugality ratio of EDF-MAXVAL-VCG and EDF-MAXVAL-APROX mechanisms as a function of the number of agents. The figure shows that the frugality ratio of both mechanisms are very close. It also shows that in general the frugality ratio grows with the number of agents. Participation of more agents in the competition leads to higher frugality ratios and higher payments by agents compare to their declared values. The average frugality over all problem instances is 1.1. Although theoretically the frugality ratio can be as large as the number of the agents and also can be as small as zero, we can see that in most cases it is between 1 and 1.2.

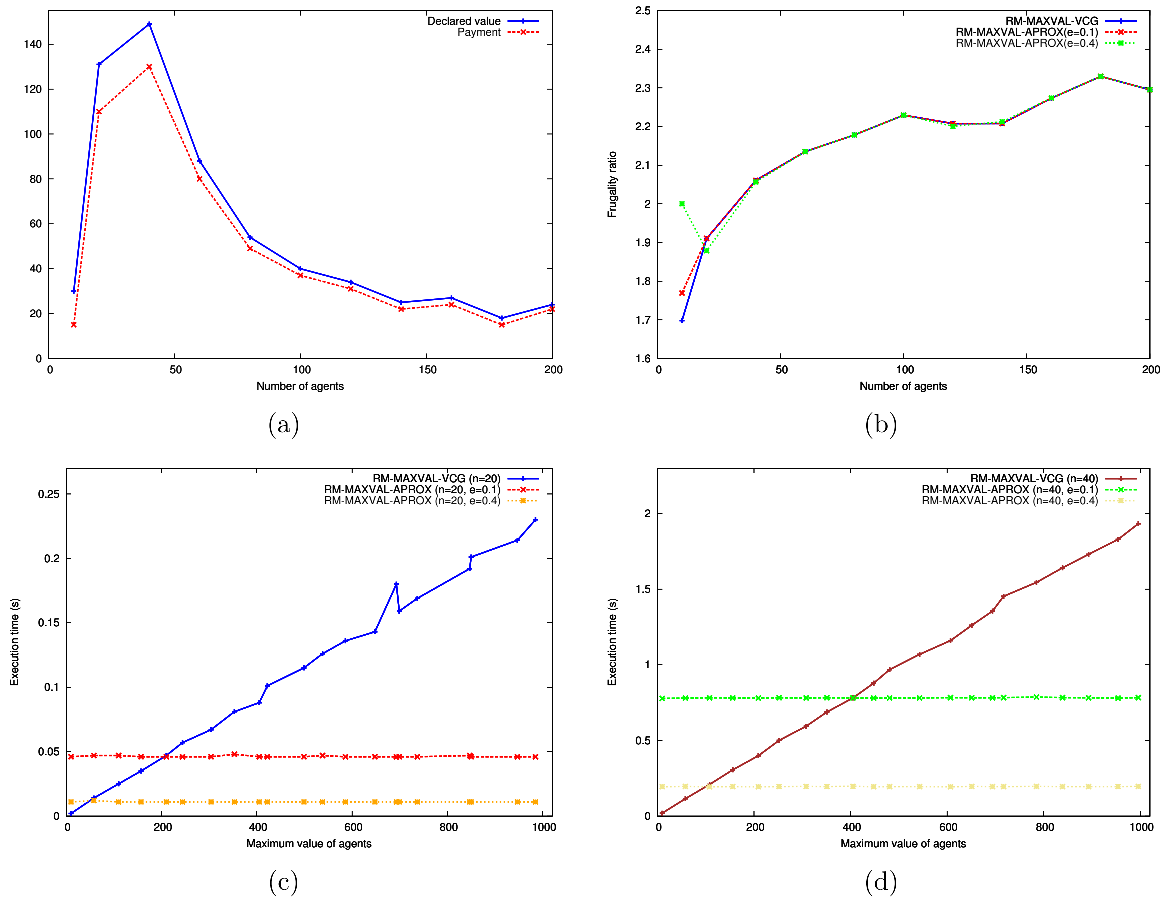


Figure 5.4: Results for RM-based mechanisms: (a) Payments vs. Values; (b) Frugality ratio as a function of number of agents; (c) The execution times of RM-MAXVAL-VCG and RM-MAXVAL-APROX for $n = 20$; (d) The execution times of RM-MAXVAL-VCG and RM-MAXVAL-APROX for $n = 40$;

The results for RM-MAXVAL problem are shown in Figure 5.4(a) and Figure 5.4(b).

5.3 Execution Time

We now perform simulations to compare the execution time of the EDF-MAXVAL-VCG and EDF-MAXVAL-APROX mechanisms. As we mentioned, the time complexity of the EDF-MAXVAL-DP is $O(n^2V)$, where V is the maximum of the agents' declared values. Thus, the execution time of the EDF-MAXVAL-VCG mechanism is highly dependent on V. In this set of experiments, we run the mechanisms for different values of V. We fix the number of agents and for each $V \in \{10, 60, \ldots, 1010\}$, we generate ten problem instances and plot the average execution time. We perform the experiments for $n = 20$ and $n = 40$. The utilization of the agents are generated using the same method we discussed in Section 5.2. We use a uniform random number generator to generate values in $[1, V]$. The execution times of both the EDF-MAXVAL-VCG and EDF-MAXVAL-APROX mechanisms are displayed in Figure 5.3(c). The figure reveals that the execution time of the approximation mechanism is lower than the execution time of the exact VCG mechanism for $V > n/\epsilon$. As the value of V increases, the performance of the approximation mechanism improves compared to the exact VCG mechanism. For small values of V, the performance of the exact mechanism is better. The reason is that in the approximation algorithm EDF-MAXVAL-AA, we multiply each value by $n/(\epsilon \cdot V) = n/(0.1 \cdot V) = 10n/V$. So for $V < 10n$, EDF-MAXVAL-AA takes more time to complete. However, in Figure 5.3(c), the approximation mechanism has better performance for $V > 200$, when $n = 20$, and for $V > 400$ when $n = 40$.

We obtained similar results by comparing the RM-MAXVAL-VCG and RM-MAXVAL-APROX mechanisms. The time complexity of the RM-MAXVAL-VCG is $O(n^3V)$, where V is the maximum of the agents' declared values. We performed the same set of experiments as in the case of EDF-based mechanisms and the results are shown in Figure 5.4(c).

CHAPTER 6
CONCLUSION

Our main objective was to introduce the concept of designing real-time systems with competition in mind. In this thesis, we explored the scheduling of implicit-deadline sporadic task systems in a competitive environment in which each task is owned by a selfish agent. Since each agent is self-interested and tries to maximize her own goals, we used the mechanism design theory to design mechanisms to incentivize honest behavior on the agents' part. Since VCG Mechanism is the only exact mechanism that satisfies the truthfulness property, we designed VCG-based mechanisms for solving the problems. The VCG mechanisms are computationally intractable; thus, we designed truthful approximation mechanisms, which use fully-polynomial time approximation algorithms to optimally allocate the processor to the agents. In this thesis we considered only single deviations by the agents. As a future work, we will study the effect of collusion among agents to lower their payments.

Our larger research goal is a comprehensive exploration of how competition affects real-time resource allocation. Thus, for future work, we also plan to extend this initial result to more complex real-time settings. As a next step, we would like to extend the setting to exact mechanisms for uniprocessor fixed-priority scheduling that is non-trivial and requires fundamentally new results not currently present in the mechanism design literature. Our hope is that a thorough exploration of competition in real-time systems will inform the design of future open and distributed real-time, embedded, and cyber-physical systems.

BIBLIOGRAPHY

[1] Tarek Abdelzaher. Interdisciplinary foundations for open cyber-physical systems. In *Proc. of the 8th Intl. Conf. on Formal Modeling and Analysis of Timed Systems*, 2010.

[2] Luca Abeni, Giorgio Buttazzo, Scuola Superiore, and S. Anna. Integrating multimedia applications in hard real-time systems. In *Proc. of the 19th IEEE Real-time Systems Symp.*, pages 4–13, 1998.

[3] Gagan Aggarwal and Jason D. Hartline. Knapsack auctions. In *Proc. of the 17th ACM-SIAM Symp. on Discrete Algorithms*, pages 1083–1092, 2006.

[4] Aaron Archer and Éva Tardos. Frugal path mechanisms. *ACM Trans. Algorithms*, 3:3:1–3:22, February 2007.

[5] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Optimal reward-based scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, 50(2):111 –130, February 2001.

[6] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang. On the competitiveness of on-line real-time task scheduling. In *Proc. of the 12th IEEE Real-Time Systems Symp.*, pages 106 –115, December 1991.

[7] S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, and D. Shasha. On-line scheduling in the presence of overload. In *Proc. of the 32nd Symp. on Foundations of Computer Science*, pages 100–110, 1991.

[8] Enrico Bini and Giorgio C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Syst.*, 30:129–154, May 2005.

[9] Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *Proc. of the 37th ACM Symp. on Theory of Computing*, pages 39–48, 2005.

[10] Giorgio C. Buttazzo. *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series).* Springer-Verlag TELOS, Santa Clara, CA, USA, 2004.

[11] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[12] R.I. Davis and A. Burns. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In *Proc. of the 30th IEEE Real-Time Systems Symp.*, pages 398 –409, December 2009.

[13] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Econ. Rev.*, 97(1):242–259, 2007.

[14] Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, and Scott Shenker. A bgp-based mechanism for lowest-cost routing. In *Proc. of the 21st ACM Symp. on Principles of Distributed Computing*, pages 173–182, 2002.

[15] Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21 – 41, 2001.

[16] Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[17] A.R. Karlin and D. Kempe. Beyond VCG: frugality of truthful mechanisms. In *Proc. of the 46th IEEE Symp. on Foundations of Computer Science*, pages 615 – 624, October 2005.

[18] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49:577–602, September 2002.

[19] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Proc. of Real Time Systems Symposium*, pages 166 –171, 1989.

[20] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20:46–61, January 1973.

[21] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, June 1995.

[22] A. Mohammadi, N. Fisher, and D. Grosu. Real-time competitive environments: Truthful mechanisms for allocating a single processor to sporadic tasks. In *Proc. of 24th Euromicro Conference on Real-Time Systems*, pages 199 –208, july 2012.

[23] A. Mohammadi, N. Fisher, and D. Grosu. Truthful mechanisms for allocating a single processor to sporadic tasks in competitive real-time environments. In *IEEE Transaction on Computers (submitted)*, 2012.

[24] A. K. Mok. *Fundamental Design Problems of Distributed Systems for Hard Real-Time Environments*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1983.

[25] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract. In *Proc. of the 18th National Conference on Artificial intelligence*, pages 379–384, 2002.

[26] John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[27] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[28] Ryan Porter. Mechanism design for online real-time scheduling. In *Proc. of the 5th ACM Conference on Electronic Commerce*, pages 61–70, 2004.

[29] A. Al Sheikh, O. Brun, P. E. Hladik, and B. J. Prabhu. A best-response algorithm for multiprocessor periodic scheduling. In *Proc. of the 23rd Euromicro Conference on Real-Time Systems*, pages 228–237, 2011.

[30] Kunal Talwar. The price of truth: Frugality in truthful mechanisms. In *Proc. of the 20th Symp. on Theoretical Aspects of Computer Science*, 2003.

[31] V.V. Vazirani. *Approximation algorithms.* Springer, 2001.

[32] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):pp. 8–37, 1961.

ABSTRACT

TRUTHFUL MECHANISMS FOR REAL-TIME SYSTEM SCHEDULING IN COMPETITIVE ENVIRONMENTS

by

ANWAR MOHAMMADI

December 2012

Advisor: Dr. Nathan Fisher

Major: Computer Science

Degree: Master of Science

In a non-competitive environment, sporadic real-time task scheduling on a single processor is well understood. In this thesis, we consider a competitive environment comprising several real-time tasks vying for execution upon a shared single processor. Each task obtains a value if the processor successfully schedules all its jobs. Our objective is to select a feasible subset of these tasks to maximize the sum of values of selected tasks. We consider both dynamic-priority and static-priority scheduling algorithms. There are algorithms for solving these problems in non-competitive settings. However, we consider these problems in an economic setting in which each task is owned by a selfish agent. Each agent reports the characteristics of her own task to the processor owner. The processor owner uses a mechanism to allocate the processor to a subset of agents and to determine the payment of each agent. Since agents are selfish, they may try to manipulate the mechanism to obtain the processor. We are interested in truthful mechanisms in which it is always in agents' best interest to report the true characteristics of their tasks. We design exact and approximate truthful mechanisms for this competitive environment and study their performance.

AUTOBIOGRAPHICAL STATEMENT

Anwar Mohammadi received his MS degree in Computer Science from Wayne State University, Detroit, Michigan. He also received a MS degree in Computer Science from Sharif University of Technology, Tehran, Iran and a B.Sc. degree in Computer Engineering from Amirkabir University of Technology, Tehran, Iran. At present he is working as Graduate Research Assistant in the Compositional and Parallel Real-Time Systems (CoPaRTS) Laboratory in the Department of Computer Science at Wayne State University under the supervision of Dr. Nathan Fisher and Dr. Daniel Grosu. His areas of research interests include real-time systems, mechanism design and algorithmic game theory.

Before joining to the MS program at Wayne State Universiy, Anwar worked in Polfilm Company, Tehran, Iran as IT manager and software developer. He developed a software system for controlling the production process of the factory. He has also several software developing experience in different software companies in Tehran, Iran.

Anwar is a student member of IEEE society. He can be reached at amohammadi @wayne.edu.