DISCRETE TIME/COST TRADE-OFF PROJECT SCHEDULING PROBLEM – AN APPLICATION TO THE MINISTRY OF HEALTH PROJECTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZLEM AKBUDAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

APRIL 2022

Approval of the thesis:

DISCRETE TIME/COST TRADE-OFF PROJECT SCHEDULING PROBLEM – AN APPLICATION TO THE MINISTRY OF HEALTH PROJECTS

submitted by **ÖZLEM AKBUDAK** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Industrial Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Esra Karasakal
Head of the Department, **Industrial Engineering** _____

Prof. Dr. Meral Azizoğlu
Supervisor, **Industrial Engineering, METU** _____

Assist. Prof. Dr. Gülşah Karakaya
Co-Supervisor, **Business Administration, METU** _____

Examining Committee Members:

Prof. Dr. Haldun Süral
Industrial Engineering, METU _____

Prof. Dr. Meral Azizoğlu
Industrial Engineering, METU _____

Assist. Prof. Dr. Gülşah Karakaya
Business Administration, METU _____

Assoc. Prof. Dr. M. Alp Ertem
Industrial Engineering, Çankaya University _____

Assist. Prof. Dr. Banu Yüksel Özkaya
Industrial Engineering, Hacettepe University _____

Date: 19.04.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Surname : Özlem Akbudak

Signature :

ABSTRACT

DISCRETE TIME/COST TRADE-OFF PROJECT SCHEDULING PROBLEM – AN APPLICATION TO THE MINISTRY OF HEALTH PROJECTS

Akbudak, Özlem
Master of Science, Industrial Engineering
Supervisor: Prof. Dr. Meral Azizoğlu
Co-Supervisor: Assist. Prof. Dr. Gülşah Karakaya

April 2022, 76 pages

The Discrete Time/Cost Trade-off (DTCT) problem is a widely studied and important research area in project scheduling literature. Decision-makers try to select the best schedule alternative when there are two or more conflicting criteria. So-called Time/Cost Trade-off problems represent the two conflicting criteria generalized as time and cost. Decreasing the processing time of a task requires more resources which demand additional cost.

This study focuses on the DTCT problems where some of the tasks have due dates. If a completion time of a task exceeds the due date of a task, we face tardiness. We aim to find the best solution in terms of cost, with the total tardiness not exceeding the maximum allowable total tardiness level.

We take our motivation from the two Information Technology projects that were most recently undertaken in the Ministry of Health.

We construct a mixed integer linear programming model for this well-defined problem. Additionally, we propose a heuristic approach, which makes use of the

optimal solutions of the Linear Programming Relaxation of the model. We tested the performance of the mathematical model and heuristic approach on several instances taken from the literature and on the two Ministry of Health projects, and report favorable results.

Besides, one of the well-known Evolutionary Algorithm is applied to generate all non-dominated objective vectors with respect to the total cost and total tardiness criteria. We illustrate the algorithm on the two Ministry of Health projects.

Keywords: Project Scheduling, Discrete Time/Cost Trade-off Problem, Tardiness, Mathematical Model, Heuristic Approach

ÖZ

KESİKLİ MALİYET/ZAMAN ÖDÜNLEŞİMLİ PROJE ÇİZELGELEME PROBLEMİ – T.C. SAĞLIK BAKANLIĞI PROJELERİ ÜZERİNDE UYGULAMA

Akbudak, Özlem
Yüksek Lisans, Endüstri Mühendisliği
Tez Yöneticisi: Prof. Dr. Meral Azizoğlu
Ortak Tez Yöneticisi: Dr. Ögr. Üyesi Gülşah Karakaya

Nisan 2022, 76 sayfa

Kesikli Zaman/Maliyet Ödünleşimli problemler proje çizelgeleme literatüründe kapsamlı olarak çalışılan ve önemli bir araştırma alanıdır. İki veya daha fazla zıt kriterin bulunduğu durumlarda karar vericiler proje için en iyi çizelge alternatifini seçmeye çalışırlar. Zaman maliyet ödünleşim problemleri birbirlerine zıt amaç fonksiyonlarını genel olarak zaman ve maliyet kategorileri altında değerlendirirler. Bir aktivitenin süresini kısaltmak daha fazla kaynak ihtiyacını doğurur ve maliyet artışına neden olur.

Bu çalışmada, teslim tarihli aktivitelerin de yer aldığı projeler için kesikli zaman maliyet ödünleşimli problemler üzerine odaklandık. Teslim tarihinden sonra tamamlanan aktiviteler gecikmeler ile karşılaşmamıza neden olur. Amacımız, kabul edilebilir toplam gecikme miktarı veya bu miktarın altında kalmak şartı ile optimal maliyet değerini veren proje çizelgesi üretmektir.

Yakın zamanda T.C. Sağlık Bakanlığı tarafından üstlenilen iki Bilgi Teknolojileri projesi bu çalışmanın motivasyon kaynağı oldu.

Bu iyi tanımlanmış problem için karmaşık tam sayılı doğrusal model geliştirdik. Ayrıca, geliştirdiğimiz modelin doğrusal programlama gevşemesinin optimal çözümünden yararlanan bir sezgisel yaklaşım algoritması oluşturduk. Matematiksel modelimizi ve sezgisel yaklaşımımızı literatürden aldığımız bazı problemler ve iki T.C. Sağlık Bakanlığı projesi üzerinde test edip, sonuçları raporladık.

Bunun yanında, toplam gecikme ve toplam maliyet kriterlerine uygun tüm domine edilmeyen amaç fonksiyonu vektörlerini elde etmek için iyi bilinen bir evrimsel algoritmanın uygulamasını yaptık. Bu algoritmayı iki T.C. Sağlık Bakanlığı projesi üzerinde gösterdik.

Anahtar Kelimeler: Proje Çizelgeleme, Kesikli Zaman/Maliyet Ödünleşimi, Gecikme, Matematiksel Model, Sezgizel Yaklaşım

To My Mother and Father

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor Prof. Dr. Meral Azizoğlu, and my co-supervisor Assist. Prof. Dr. Gülşah Karakaya. Their guidance and encouragements help me to complete my study. They offered me their deep knowledge and expertise with a lot of patience and support whenever I need. It was an honor for me to study with both of them.

I could not have imagined having a better team of advisors; Their positive vibes always motivated me thorough my thesis study.

I would like to express my sincere appreciation to Assoc. Prof. Dr. M. Alp Ertem. He always made time to guide and support me since my undergraduate education. I learned a lot from him and feel lucky to have him as a mentor.

Also, I would like to thank Prof. Dr. Haldun Süral for being a member of the examining committee, and for motivating me with his positive energy and support during my thesis defense.

I would like to thank Assist. Prof. Dr. Banu Yüksel Özkaya for being a member of the examining committee and sharing her valuable comments on this thesis besides her kind support.

Industrial Engineering Department and Informatics Institute of METU are great places to continue higher education. I am honored to be a part of METU.

Finally, I am more than thankful to my family and friends who always supported and encouraged me during my journey at METU.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Project management is defined as “The application of knowledge, skills, tools, and techniques to project activities to meet project requirements.” by the Project Management Institute (Project Management Institute Inc. 2021a). The right people, who have the required knowledge and experience, can lead projects into success by using the right tools and techniques. According to the Pulse of the Profession 2021 report (Project Management Institute Inc. 2021b), even though there has been an improvement over the years, still 55 percent of the projects are completed on time and 62 percent are completed within the original budget. Additionally, projections show that the global economy requires 25 million new project professionals by 2030 (Project Management Institute Inc. 2021c). Failing to meet this talent gap could come with a loss of up to $345.5 billion in global GDP by 2030 (Project Management Institute Inc. 2021c). Therefore, project management has gained importance over the years while the world has become more project-oriented.

Project Scheduling is an essential part of project management. It defines the start and completion times of the activities and a proper selection of processing modes for the activities. Meeting the project goals in a predefined budget and time requires realistic project schedules created with the help of tailored tools and techniques.

Project professionals in all sectors use the project scheduling tools and techniques at all phases of the project. Even though the construction and manufacturing sector highly benefit from the project management and scheduling tools and techniques from the very beginning, the Information and Technology (IT) sector has also become one of the top leading project-oriented sectors over the years with the driving role of digitalization.

Healthcare is one of the enormous sectors where digitalization plays a significant role and that benefits from the successful project scheduling tools and techniques. According to the OECD data, Turkey, Denmark, Germany, the UK, and the US spent 4.3, 10.6, 12.5, 12.8, and 16.8 percent of their GDP on healthcare projects (OECD 2022).

The Ministry of Health (MOH) is the main governmental entity responsible for the healthcare service of Turkey. Besides the policymaking, the implementation of the national health strategies through programs is also another duty of the MOH. The MOH conducts and participates in a considerable number of health projects most of which lie in the IT area.

This thesis considers the two IT projects that are most recently undertaken by the MOH. These projects have tasks with more than one processing alternative. Some of the tasks, in particular the ones that define the milestones, are required to meet predefined termination times, so-called the due dates. The termination times that exceed the due dates yield tardiness. The managers tolerate total tardiness, however to some extent. They define an upper bound on the maximum total tardiness value above which the solutions are not accepted. They aim to minimize the total cost of task time reductions. Based on the way that the managers of the MOH follow, we define a discrete time-cost trade-off problem. We generate project schedules, which have total tardiness values under the maximum tolerable amount while minimizing the total cost. We discuss the use of this tardiness-related constrained problem to generate all nondominated objective vectors for the total tardiness and total cost criteria. Our motivation is to help decision-makers to see the trade-offs between total tardiness and total cost criteria, and decide the best action with the most reliable information.

We develop a mathematical model to represent our constrained optimization problem. We observe that the model is not capable of solving large-sized problem instances. To tackle those instances, we design a heuristic algorithm that runs in two steps: the first step finds an initial solution using the optimal solutions of the Linear

Programming Relaxations of the mathematical model and the second step improves the initial solution by several pairwise interchange mechanisms. We tested the performances of the mathematical modal and the heuristic algorithm on the two real-life problems and on the instances gathered from the literature. We find that the model cannot handle all generated instances in our termination limit of two hours and the heuristic procedure finds high quality approximate solutions very quickly.

To generate all nondominated objective vectors with respect to the total tardiness and total cost criteria, we propose an ε-constraint approach that uses the optimal solutions of the constrained optimization model iteratively. We also apply a well-preferred Evolutionary Algorithm, NSGA-II, to find the approximate nondominated set of the objective vectors. We test the performances of the ε-constraint approach and Evolutionary Algorithm on the MOH projects.

The rest of the thesis is organized as follows. In Chapter 2, we define the projects in general, give the terminology on the project networks and project scheduling terms, and review the related literature. Chapter 3 provides general information on the MOH and the two real-life projects most recently undertaken by the MOH. We also explain the objectives, constraints, and current solutions/drawbacks in Chapter 3. In Chapter 4, we define our problem, present the mathematical model and the heuristic approach. Chapter 5 discusses the generation of all nondominated objective vectors and presents ε-constrained approach and the Evolutionary Algorithm. In Chapter 6, we report the results of our computational experiment for the constrained optimization problem. Chapter 7 concludes the study by summarizing the main results of our work and pointing out some future research directions.

CHAPTER 2

GENERAL INFORMATION ON PROJECTS AND LITERATURE REVIEW

In Chapter 2, we give general information about the projects and a general view of the project scheduling problems. We also review the literature on project scheduling, including single-mode and multi-mode project scheduling problems. Finally, we discuss the contribution of this study to the current literature.

2.1 Projects in General

Project Management Institute (PMI) is a worldwide accepted organization in the field of project management. It was established in 1969 to guide project management professionals and organizations through their goals with the help of project management principles and standards. PMI publishes a guide entitled ‘A Guide to the Project Management Body of Knowledge (PMBOK Guide)’ to support its aim of presence.

PMI defines a project as "A temporary endeavor undertaken to create a unique product, service, or result." in the PMBOK Guide (Project Management Institute Inc. 2021a). The guide also states that phases of the project tasks or the whole work between the project’s start and the finish times are defined by the project.

Project management is defined as "The application of knowledge, skills, tools, and techniques to project activities to meet project requirements." in the PMBOK Guide (Project Management Institute Inc. 2021a). The project manager, the responsible person for the project, makes decisions to complete the project successfully. Widely used or tailor-made tools or models help the project manager with these decisions.

2.1.1 Project Networks

The aforementioned tools use common representations and terms. The smallest and most meaningful workpiece is called activity. We use activity and task interchangeably in this thesis. The relation between the tasks is widely named precedence relations. Suppose that Task 1's output is required by Task 2 as an input. Then, starting time of Task 2 depends on the completion of Task 1. In this case, we say that there is a precedence relation between Task 1 and Task 2, and Task 1 is an immediate predecessor for Task 2. In other words, we can also say that Task 2 is an immediate successor for Task 1.

There are two different types of representations of the project networks in the literature: Activity on Node (AoN) and Activity on Arc (AoA). We next explain these representations.

2.1.1.1 Activity on Node (AoN) Representation

In AoN representation, tasks are represented by nodes, and arcs represent the precedence relations between the tasks. Arcs come out of one node and go into another node. The node that the arrow comes out is the immediate predecessor node, and the node that the arrow goes into is the immediate successor node.

An example project network is created to illustrate the AoN representation is given below.

Table 2.1 Precedence Relations of the Example Network

Task	*Immediate Predecessor*
0	-
1	0
2	0
3	1
4	1, 2
5	3, 4

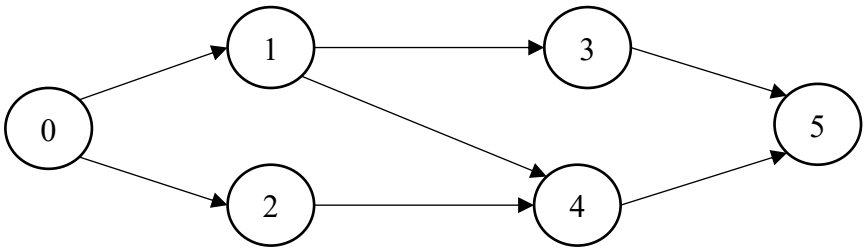

Figure 2.1 AoN Representation of Example Network

Figure 2.1 depicts that Task 1 is the immediate predecessor of Tasks 3 and 4, and the predecessor (but not immediate) of Task 5. Accordingly, Tasks 3 and 4 are immediate successors of Task 1, and Task 5 is the successor (but not immediate) of Task 1.

2.1.1.2 Activity on Arc (AoA) Representation

In AoA representation, tasks are represented by the arcs. For each arc there are two nodes named tail and head nodes. The activity comes out of the tail node and goes into the head node. Parallel arcs are not allowed in the project networks; hence, to represent some precedence relations, dummy activities may be needed.

AoA representation of the example network is given below.

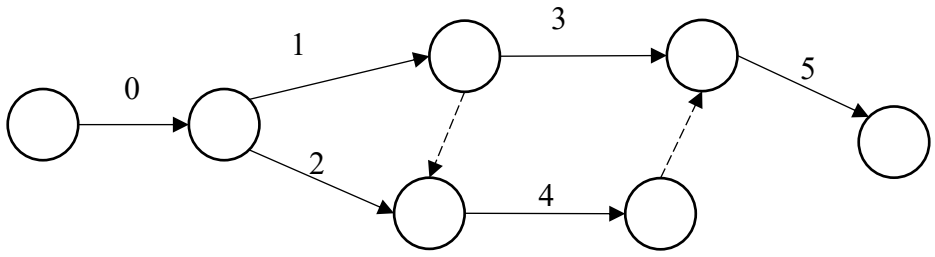

Figure 2.2 AoA Representation of the Example Project Network

The dashed lines in Figure 2.2 are dummy activities with zero time and cost.

2.2 Project Scheduling

Project scheduling determines the start time and finish time of the tasks while considering the precedence relations, durations, and consumed resources (usually indicated by cost) of these tasks. There are different tools in the literature to help schedule the project. Selecting the tools, algorithms, or models for scheduling activities also depends on the nature of the project. Project managers' aim may change with the constraints of the elements of the project. For example, some projects may have budget constraints while others have time limitations. In the literature, projects with tasks having only one processing alternative are defined as single-mode projects. Alternatively, in some projects, we may have an option to reduce the required processing time of a task to complete in exchange for an extra

cost. In the literature, project scheduling problems in which some tasks that have more than one processing alternative are called multi-mode project scheduling problems. We name the time–cost pairs as processing alternatives or modes.

2.2.1 Single-Mode Project Scheduling Problems

In single-mode project scheduling, tasks have fixed durations. The outputs of a single-mode project scheduling are the task start times and finish times obeying the precedence relations.

Critical Path Method (CPM) is the well-known and widely used method to construct a project schedule with project completion time. The definitions that are frequently used in the CPM are given below.

Critical Path: The longest path(s) in the project network is called the critical path.

Total Slack: The difference between the earliest start time and latest start time, or the earliest completion time and the latest completion time of the task, is called as total slack of the task. The positive total slack means that the task can be moved back or forward with the amount of total slack without causing any delay in the project completion time.

Critical Activity: The tasks with zero total slack are named as critical. Any delay of a critical activity causes a delay in the project completion time.

First, we select the tasks without any predecessor and set their start times as zero. A task's completion time equals the sum of the task's start time and duration. We select the maximum completion time to assign as the start time of the task with predecessors. If the task requires waiting some predefined time to start (lag), we add this lag to the maximum completion time of the predecessor tasks to find the start time of the successor task. We repeat this procedure until we find all the tasks' start and completion times. These start and completion times are also named as earliest start and completion times. The last task's completion time also equals the earliest

project completion time. Additionally, the process of finding the earliest times is called forward pass.

Afterwards, we select the tasks without any successor and use their earliest completion time found during a forward pass as the latest completion time. The latest start time of the task is the latest completion time of the task minus its duration. To find the latest completion time of the tasks with successors, we use the minimum latest start time of the immediate successor tasks. We complete the calculations in this direction until we find the latest start time of the tasks which do not have predecessors. This process is named backward pass.

After completing forward pass and backward pass, we detect the tasks with zero total slack. These tasks are marked as critical activities. The longest path consists of only critical activities, which we call the critical path. The project managers give extra attention to the critical path since any delay on this path directly affects the project's completion time.

We take the stepwise description of the CPM below from (Değirmenci 2008) who use the following notation:

t_i: Processing (task) time of activity i.
P_i: Set of immediate predecessors of activity i.
S_i: Set of immediate successors of activity i.
ES_i: Earliest start time of activity i.
LS_i: Latest start time of activity i.
EC_i: Earliest completion time of activity i.
LC_i: Latest completion time of activity i.
$Crit$: Set of critical activities.
$Slack_i$: Total slack of activity i.

Initialization:

$ES_i = 0$ $\quad i: P_i = \emptyset$

Main Body:

Repeat

$ES_i = \max_{j \in P_i}\{ES_j + t_j\}$ $\quad i: \forall j \in P_i\ ES_j\ is\ calculated$

Until $ES_i\ for\ i = 1,2, \dots, N\ are\ calculated$

$T = \max_i\{ES_i + t_i\}$

$LC_i = T$ $\quad i: S_i = \emptyset$

Repeat

$LC_i = \min_{j \in S_i}\{LC_j - t_j\}$ $\quad i: \forall j \in S_i\ LC_j\ is\ calculated$

Until

$LC_i\ for\ i = 1,2, \dots, N\ are\ calculated$

Finalization:

$Slack_i = LC_i - ES_i$ $\quad i = 1, 2, \dots, N$

$Crit = \{i = 1, 2, \dots, N\ | Slack_i = t_i\}$

2.2.2 Multi-Mode Project Scheduling Problems

In multi-mode project scheduling problems, one task or more than one task has at least two processing time (task time) options. The task time options require different amounts of resources represented by cost. These processing time–cost pairs are called as modes. Real-life practices show that reducing the duration of a task demands additional effort and/or resources, which finally comes with additional cost. Naturally, project managers are in favor of completing the projects with less time and money spent. Therefore, there is a trade-off between time and cost.

To illustrate the concept of multi-mode project scheduling, the example is given in Table 2.2, below.

Table 2.2 Multi-mode Project Scheduling Example Data

Task	*Normal Duration (day)*	*Cost (₺)*	*Reduced Duration (day)*	*Cost (₺)*
1	50	6000	30	12000
2	48	4500	42	5000
3	55	6200	45	8000
4	10	2000	8	1900
5	5	500	4	950

Note from the above table that there are two modes for all tasks and decreasing the time of the task increases its required cost. The project manager tries to select the modes for the desired project outcome. These kinds of problems are called Discrete Time/Cost Trade-off (DTCT) problems in the literature.

2.3 Literature Review on Time/Cost Trade-off Problems

In this section, we give the literature review for the linear and discrete time/cost trade-off (multi-mode) project scheduling problems. We discuss the related works in chronological order.

2.3.1 Linear Time/Cost Trade-off Project Scheduling

Choi and Chung (2014) and Choi and Park (2015) focus on linear project time/cost trade-off problems. Choi and Chung (2014) consider milestone activities whose tardiness values are penalized. They present mathematical models to minimize the total weighted number of tardy milestones. The first model minimizes the total

weighted number of tardy milestones within the total compression budget. The second model minimizes the total weighted number of tardy milestones plus the total cost of reducing the activity durations.

Choi and Park (2015) assume some activities with deadlines. Their objective is to minimize the total penalty cost plus the total cost of reducing the activity durations. They show that the problem is NP-hard in the ordinary sense and propose an optimization algorithm that runs in pseudo-polynomial time.

2.3.2 Discrete Time/Cost Trade-off Project Scheduling

We review the discrete time/cost trade-off (DTCT) problems in three categories: the deadline problem, the budget problem, and the time/cost curve problem. The deadline problem minimizes the total cost of mode assignments without exceeding the prespecified project deadline. The budget problem minimizes the project completion time subject to the constraint that the total cost of mode assignments does not exceed the available budget. The time/cost curve problem generates all non-dominated vectors with respect to the total cost and the project completion time objectives. De et al. (1997) show that all three DTCT problems are NP-hard in the strong sense.

2.3.2.1 Deadline Problem

Crowston and Thompson (1967) propose a Decision Critical Path Method (DCPM) that considers alternative methods to complete a job in the project. Alternative methods of completing a job may represent various cost and time options or technological dependencies. They formulate a mixed-integer programming model and propose a heuristic approach for large sized problem instances.

Crowston (1970) suggests a reduced version of the DCPM network that includes only the decision nodes and the maximum distances between these nodes, and ignores precedence relations.

Hindelang and Muth (1979) also focus on the DCPM and propose a dynamic programming approach.

De et al. (1995) show that Hindelang and Muth (1979)'s approach cannot handle two immediate predecessors without causing a multi-dimensionality problem. They propose a new dynamic programming model for the deadline problem that corrects the approach of Hindelang and Muth (1979). Besides, they indicate that their proposed model can be used to generate a time/cost trade-off curve. Moreover, De et al. (1995) introduce a network decomposition approach and discuss its application to the DCPM.

Demeulemeester et al. (1996) present a dynamic programming approach to solve DTCT problems. They propose two procedures, one of which finds the minimal number of node reductions to transform a general network to a series-parallel network. The other method aims to minimize the difficulty of enumerating alternative modes.

Skutella (1998) works on a discrete time/cost problem with a fixed budget. Skutella (1998) proposes the first approximation algorithm with a performance guarantee of $O(\log l)$, where l is the ratio of the maximum time to the minimum nonzero time for both deadline and the budget problems. He also discusses the bi-criterion approximation algorithms for the budget and deadline problems.

Vanhoucke et al. (2002) study the deadline problem with a special constraint that forces a specified starting time for some activities. They propose a branch-and-bound algorithm and a heuristic procedure.

Grigoriev and Woeginger (2004) generalize the DTCT problem with irregular costs that depend on activities' starting and completion times. They prove the NP-hardness

of the deadline problem and show that the problem is polynomially solvable when the precedence constraints are series-parallel.

Vanhoucke and Debels (2005) introduce three different extensions of the DTCT problem's deadline version: time/switch constraints, work continuity constraints, and net present value maximization. They develop a new meta-heuristic approach for powerful approximate solutions.

Akkan et al. (2005) discuss the hardness of the deadline problems in tackling large-sized instances and provide lower bounds on their total cost values.

He and Xu (2008) work on the deadline problem and consider bonus/penalty incentives on the project's deadline. They focus on the payment schedule and try to decide the timing and magnitude of the payments. They propose a simulated annealing heuristic search for their mixed-integer non-linear model.

Hafızoğlu and Azizoğlu (2010) consider a deadline model and use of the deadline model in generating the time/cost curve. They propose a branch-and-bound algorithm that benefits from the optimal solutions of the linear programming relaxations.

Hazir et al. (2011) work on a deadline version of DTCT problem where the activity times are known but activity costs have interval uncertainty. They propose an exact method based on Benders decomposition and a tabu search algorithm for the approximate method.

Said and Haouari (2015) focus on a deadline version of the stochastic discrete time/cost trade-off problem. They try to select modes to the activities while minimizing the activity costs and tardiness costs. They also aim for the stability of the schedule against uncontrollable factors. They propose a two-stage solution approach that first uses simulation optimization to find a cost-effective schedule and then mixed-integer programming to stabilize the resulting schedule.

Aminbakhsh and Sonmez (2016) propose a swarm optimization method for an effective solution of the cost optimization problem. The proposed method solves instances with up to 630 activities in reasonable times and with small deviations.

2.3.2.2 Budget Problem

Robinsonf (1975) works on the time/cost trade-off budget problem, which has activities with arbitrary cost-time functions. He features a dynamic programming approach to allocate the resources while trying to minimize the duration of the project.

Hazir et al. (2010a) focus on a multi-mode DTCT problem with budget constraints. They develop a branch-and-cut procedure that is based on decomposition idea and report that their algorithm can solve instances with up to 136 activities in reasonable times.

Hazir et al. (2010b) discuss a stochastic budget problem where they aim to build schedules that are less sensitive to uncontrollable parameters. They propose a two-step methodology First, they find the minimum required budget and then inflate the budget slightly using a specified amplification factor.

Değirmenci and Azizoğlu (2013) study the budget problem and propose a branch-and-bound algorithm to solve medium-sized problem instances to optimality and a heuristic algorithm based on the linear programming relaxation to solve large-sized problems approximately.

2.3.2.3 Curve Problem

Fulkerson (1961) introduces a network flow method to create the complete project cost curve (time/cost curve). The author describes the project network, which has jobs with normal and crashed completion times. The cost of completing these jobs associates within the normal and crashed completion time intervals. He indicates that

the desired solution lies on the time/cost curve of the project while proving that this curve is piecewise linear and convex between normal and crash completion times.

Feng et al. (1997) consider time/cost trade-off curve problem. They mention that the existing methods are not capable of dealing the real-life construction projects. They propose a genetic algorithm that searches only a small part of the search space.

Demeulemeester et al. (1998) generate a time/cost curve for a DTCT problem where the task times are functions of a single resource. They use a horizon-varying approach-based procedure to minimize the total resource used.

Chassiakos et al. (2005) argue that the existing literature does not consider the generalized precedence relations, external time constraints, activity planning constraints, and incentives for early or delayed project completions. They propose mathematical programming models and provide an approximation procedure by considering the characteristics aforementioned above to provide a project time/cost curve.

Szmerekovsky and Venkateshan (2012) give three integer programming formulations for the time/cost trade-off problem where the costs are irregular. Their models could solve problems with up to 90 activities.

Li et al. (2018) provide two bi-objective heuristic algorithms to solve the large scale DTCT problems. The aim of their study is to create the efficient frontier set for the problem. One of them is based on the NSGA-II algorithm, and the other one is based on the steepest descent heuristic algorithm. They give a performance analysis for large-sized problems.

Li et al. (2020) indicate a need to create a schedule with time and resource buffer to make these schedules durable against the uncertain factors and focus on time/cost/robustness trade-offs. They construct a multi-objective optimization model with the makespan, cost, and robustness minimization objectives. The multi-objective model is decomposed into a series of robustness-maximization models with various makespan and cost constraints to be able to deal with the NP-hardness

of the problem. Three propositions are proposed to navigate trade-off among objectives. Additionally, they provide an ε-constraint method-based genetic algorithm to generate the efficient frontier for the problem.

Eynde and Vanhoucke (2022) propose a network based decomposition algorithm to create the exact set of nondominated objective vectors. Even though their method does not outperform the best reported branch-and-bound procedure, it shows a promise of the decomposition idea for performance improvements.

2.4 Our Contribution to the Literature

In this thesis, we study the DTCT problem with task due dates. We minimize the total cost subject to the constraint that the total tardiness is below a specified value. We formulate the problem as a mixed-integer linear program and propose a linear programming based heuristic procedure for its solution. We also discuss the use of that model in finding the time/cost curve with respect to the total tardiness and total cost criteria. To the best of our knowledge, there is no reported discrete time/cost trade-off study with tardiness penalties.

CHAPTER 3

GENERAL INFORMATION ABOUT MINISTRY OF HEALTH AND ITS PROJECTS

In this chapter, we first give a general information about Ministry of Health and its projects. We then discuss the details of the two most recent projects that motivated this study.

3.1 General Information About Ministry of Health

Ministry of Health Turkey (MOH), established in 1920, is the central government body responsible for health sector policymaking, implementation of national health strategies through programs, and the direct provision of health services. MOH is the major provider of primary, secondary, and tertiary health care services in Turkey. It has fourteen general directorates and departments to fulfill its duties with its powers given with the Decree-Law 663 (Ministry of Health 2022). The directorates are:

- General Directorate of Legal Services
- Health Institution of Turkey
- Department of Strategy Development
- General Directorate of Emergency Health Services
- General Directorate of Health Promotion
- General Directorate of Administrative Services
- General Directorate of Health Investments
- General Directorate of Health Information Systems
- General Directorate of EU and Foreign Affairs

- General Directorate of Health Services
- General Directorate of Borders and Coastal Health of Turkey
- Turkish Medicines and Medical Devices Institution
- General Directorate of Turkish Public Hospitals
- General Directorate of Public Health

The General Directorate of Health Information Systems (GDHIS) is responsible for digital maturity of information systems of the MOH's healthcare services. One of the essential duties and power of GDHIS given by the Presidential Decree No. 1 on the Presidency Organization is "to make and have all kinds of information systems and projects related to the information systems, including personal health data, country-level health status, and data and information flow related to health services" (Sağlık Bilgi Sistemleri Genel Müdürlüğü 2020). Hence, GDHIS develops national and international projects to support digital health care services.

European Union (EU) has created European Union Research Framework Programmes since 1984. With the help of these multi-annual research framework programmes, EU aims to drive economic growth and create jobs. Through this instrument of funding research, EU invests in their future for smart, sustainable, and inclusive growth and jobs (Horizon 2020 | European Commission 2022). Recent framework programs of EU are Horizon 2020 (2014-2020) and Horizon Europe (2021-2027).

These framework programs support the United Nation's Sustainable Development Goals and elevate the EU's competitiveness and growth (Horizon Europe | European Commission 2022). EU opens calls within these framework programs for EU member countries and associate countries to develop projects, prepare proposals, and carry out these projects successfully (if funded) by forming consortiums.

Numerous governmental organizations and private sector players from Turkey participate in these projects. MOH is one of these participant governmental

organizations. GDHIS participates and conducts projects on the behalf of MOH, supporting digitalization in the health sector and improving organizational culture around digital health while creating employment.

3.2 Project Descriptions

EU opens calls within the framework programs for specific themes. Organizations that are willing to be part of the project search for appropriate potential partners to form a consortium with the help of brokerage events or using their network. Potential partners offer their professions to the consortium to support the project idea. Afterward, the consortium is established, consortium members write a proposal, which will be presented to the European Commission for a grant. The proposal comprises a Gantt Chart at a Work Package (WP) level. Additionally, it has WP's descriptions and high-level tasks belonging to these WPs. Sub-tasks and the dependencies within these tasks can be deduced by analyzing these descriptions. EU only needs to know (related to the project's schedule) the due dates of the deliverables, after it funds the project. The consortium conducts the project and presents the associated works by submitting the related deliverables.

GDHIS participated in various projects on the behalf of the MOH, two most recent of those projects are explained in Sections 3.2.1 and 3.2.2

3.2.1 HSMonitor

Pre-commercial Procurement of innovative ICT- enabled monitoring to improve health status and optimize hypertension care (HSMonitor) is a project funded under the Pre-Commercial Procurement Action within the Horizon 2020 Framework Program by EU. It invests in Research and Development (R&D) services towards innovative ICT-enabled monitoring solutions to improve health status and optimize hypertension care (HSMonitor - Health Status Monitor - ICT-enabled monitoring 2022). There are five healthcare providers and two private sector organizations as

partners of this project. Partners are from six countries: Turkey, Italy, Sweden, Spain, Germany, and Croatia.

Output of this project will be the software, which aims to support end-users to monitor their overall health status and assist them to manage their hypertension care with the help of using technology, easily share their results with their doctors and get feedback, and manage required processes to stay healthy. ICT- enabled solutions in non-communicable diseases aim to decrease hospital visitations and better patient monitoring practices by empowering patients on their healthcare management. As a result of using ICT-enabled products in healthcare, both patients with non-communicable diseases who conduct avoidable visitations to the hospitals and patients who require to visit hospitals unavoidably will benefit from these products and healthcare professionals in an optimally managed way.

3.2.2 STAMINA

Demonstration of intelligent decision support for pandemic crisis prediction and management within and across European borders (STAMINA) is a project funded under the Innovation Action within the Horizon 2020 Framework Programme by EU. Early detection and management of infectious diseases remain a serious challenge due to the number of people involved, the different legal, administrative, professional and political cultures, and the lack of transboundary crisis management infrastructures. STAMINA project aims to create an ICT-based solution to overcome these challenges by providing improved decision-making technology to pandemic crisis management practitioners at regional, national and international levels (The STAMINA Project - Stamina 2022). The project has 37 partners from Turkey, Greece, Austria, Netherlands, Luxembourg, Belgium, England, Germany, Italy, Slovenia, Czech Republic, Tunisia, Romania, France, Spain, and Lithuania.

The project's output is a set of guidelines and best practices to improve the response and an integrated set of software tools to support early detection and efficient

management of infectious diseases at national and international levels. STAMINA tools target both strategic level services and operational level services. There are Preparedness Pandemic Training Tool, Predictive Models, and Crisis Management Tool for strategic level end-users. For the use of operational level end-users, there are test devices, wearable devices, web and social media analytic tools, and early warning system tools.

3.3 Decisions, Parameters, Constraints, and Objectives

In this section, the common characteristics of HSMonitor and STAMINA projects are described.

The project consists of WPs each of which has its own purpose. WPs are broken down into tasks that have precedence relations within each other. Some tasks have lag and some have a due date to be met. The tasks with due dates are referred to as deliverables.

The time of a task may vary based on the number of people who work on the task, person's expertise level, required quality level, over time permit, and other reasons. Therefore, in our projects, some tasks have various time options. One of the options, normal task time, is the longest possible time that can be used with no additional cost. Other time options are relatively smaller than the normal task time and are used to reduce tardiness at an expense of increasing cost.

The following figures illustrates the characteristics of our projects.

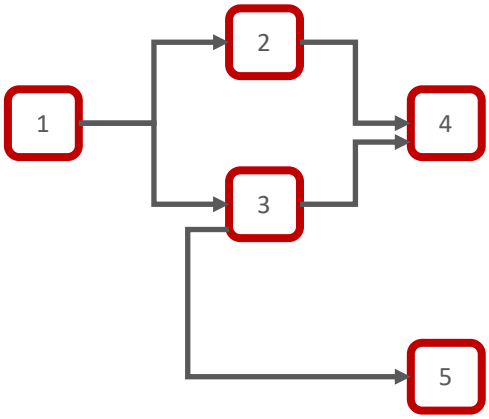

Figure 3.1 Illustration of Precedence Relations

Figure 3.1 illustrates the Finish-to-Start (FTS) and Start-to-Start (STS) precedence relations within tasks. The relation between tasks 3 and 5 is an example of STS precedence relations, while the remaining precedence relations illustrated in Figure 3.1 are examples of the FTS relations.

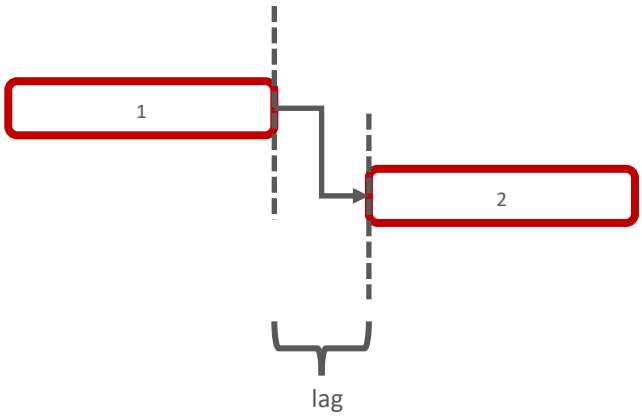

Figure 3.2 Illustration of Lag Characteristic

Figure 3.2 illustrates the lag characteristic of our projects. Task 2 requires waiting a predefined amount of time after Task 1 is completed.

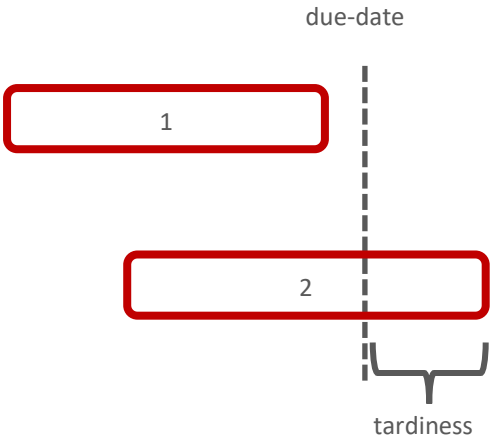

Figure 3.3 Illustration of Due-date and Tardiness Characteristics

Figure 3.3 illustrates the due-date and tardiness characteristics. Some of the tasks have due-dates called deliverables. Tardiness occurs if the task's completion time exceeds the due date of the task.

3.3.1 Decisions

On the planning of the project, EU requires the consortium to divide planned work into WPs, assign the responsibilities within the consortium, and set out a schedule for main deliverables and milestones (Get prepared - H2020 Online Manual 2022).

Even though a detailed project plan, which includes the start and finish times of the tasks, is not a direct requirement by EU, it is essential to have a detailed schedule for successful management of the project. Therefore, the project coordinator or responsible partner(s) should create a project plan that includes the start and finish times of the tasks with precedence relations to monitor the process properly. Moreover, it will be beneficial when a consortium faces any tardiness for one or

several tasks to respond with appropriate risk assessment and mitigation plans for future tasks.

To create a schedule, precedence relations and their lags, and time of the tasks are required. The critical decision for an optimal schedule is to select the best task time options that balance the total tardiness and the total cost (cost in terms of the amount of total time reduced compared to the normal task time).

3.3.2 Parameters

The parameters of the projects are listed below.

- Task time options (modes): Task times and the number of task time options vary within the tasks. Every task has at least one time option, that is, the normal task time.
- Precedence relations: Precedence relations are known and two types of them, FTS and STS, are considered.
- Lags: Some tasks may have lags.
- Due dates: Target completion times of the tasks are known beforehand. Not all, but several tasks have due dates.

3.3.3 Constraints

Precedence relations should be considered since one task is an input for one or several other tasks. In our projects, the following two types of precedence relations constraints are considered:

- Finish-to-Start with Lag: A task can start after its immediate predecessor task(s) is(are) finished and waited for its lag. If the lag is zero then a task can start after its immediate predecessor task(s) is(are) finished.

- Start-to-Start with Lag: A task can start after its immediate predecessor task(s) is(are) started and waited for its lag. If the lag is zero then a task can start after its immediate predecessor task(s) is(are) started.

3.3.4 Objectives

The projects have several deliverables, i.e., tasks with due dates. The consortium makes an effort to meet these due dates. As mentioned before, tardiness occurs when the completion time of a task is greater than the due date and its value is the difference between these two. The sum of the tardiness values for all tasks indicates the total tardiness value of the project with the schedule taken into consideration. The best practice is to complete the project with minimum total tardiness value or to complete the project without exceeding a predefined total tardiness value.

Decreasing the tardiness of a task requires either pulling back its start time or decreasing its time. In order to pull back the start time of the task, one or several predecessor tasks' times should be shortened. Eventually, efforts to decrease the tardiness come with its cost. Decreasing a task's time requires additional resources or overtime with the existing resources. Both of these options increase the overall cost of the project. Therefore, the consortium makes an effort to avoid increasing the total cost of the project. The cost of decreasing the task time can be expressed with the amount of decreased time. Reduction of the tasks indicates the difference between the normal task time and the task time selected to use in the current schedule. Therefore, total reduction signifies the sum of the reduction amounts for all tasks. Since increasing the cost of the project is not preferable by the consortium, minimization of the total cost is a valuable practice.

3.4 Current Solutions and Its Drawbacks

MOH handles several EU Framework Program projects simultaneously. Even though the application areas change for these projects, the characteristics are very

similar. EU requires the description of work on the high-level tasks, but neither a detailed breakdown of the work nor a project schedule. At the beginning of the project, it pursues the due dates for the deliverables promised in the proposal.

To manage the project successfully, the responsible partner /organization requires to keep track of the works completed and set start and completion time for the works for preparing the aforementioned deliverables. As a current solution, the upcoming steps are tried to be foreseen by the project manager based on previous experiences on similar projects with similar/same partners. Accordingly, appropriate start and completion times for the works are set for the responsible partners. Since EU requires only high-level tasks on the proposal, a complete breakdown structure of the project may not be available. Therefore, he may treat several tasks (due to uncompleted breakdown structure) as one united task. Time management problems may occur if different partners are responsible for these tasks.

Current practice has several drawbacks that are listed below:

- Due date setting for deliverables at the beginning does not base on a methodical approach. It requires specific expertise in the field and is biased in favor of the project manager's foresight.
- Deliverables have preceding tasks. An increase in complexity of the precedence relations may affect the successful management of the tasks resulting in undesirable tardiness for the deliverables.
- There may be less costly alternatives to meet the due dates of the deliverables.

In this thesis, we offer a systematic approach that can be used to create a project schedule for the MOH projects. We test the performance of the approach on the HSMonitor and STAMINA projects and obtain very favorable results.

CHAPTER 4

PROBLEM DEFINITION, MODEL, AND A HEURISTIC PROCEDURE

In this chapter we define our problem, give its mathematical model, discuss its complexity, and present a heuristic procedure.

4.1 Problem Definition

We consider a project scheduling problem with *n* tasks (activities). We define two dummy tasks, task 0 where all tasks with no immediate predecessors are connected, and task *n*+1 where all tasks with no immediate successors are connected. Accordingly, task 0 and task *n*+1 represent the start and completion of the entire project, respectively.

Task *i* has m_i processing alternatives, each alternative is defined by its task time and cost. We refer to these alternatives as modes. We let (p_{ik} , c_{ik}) be the task time and cost values of the k^{th} mode of task i. We assume that the modes are nondominated, i.e., $p_{ik_1} > p_{ik_2}$ implies $c_{ik_1} < c_{ik_2}$ for any modes k_1 and k_2 of any task i.

The time/cost pairs are indexed according to the increasing order of the times (thereby decreasing order of the cost). Accordingly, $p_{i1} < p_{i2} < \cdots < p_{im_i}$ and $c_{i1} > c_{i2} > \ldots > c_{im_i}$.

Some tasks have due dates such that their completions beyond those dates are penalized. We let d_i denote the due date of task i and penalize the task if its completion time exceeds d_i. Moreover, we let *D* be the total tardiness allowed over all tasks.

There exist two types of precedence relations between some predefined task pairs. Those relations are explained below.

- Finish-to-Start with lags

Task j cannot start before task i completes. There may be lags between the start time of one activity and the completion time of another one. We let lag_j be the time units between the start time of task j and completion time of task i. When lag_j is zero, task j can start immediately after the completion of task i. We let *FS* be the set of all Finish-to-Start relations.

- Start-to-Start with lags

Task j cannot start before task i starts. There may be lags between the start times of the two activities. We let lag_j be the time units between the start of task j and the completion of task i. When lag_j is zero, task j can start immediately after the start of task i. We let *SS* be the set of Start-to-Start precedence relations.

Our decisions are explained by the following two types of decision variable sets:

- Mode Assignment Decision Variables

$$x_{ik} = \begin{cases} 1 & \text{if mode } k \text{ is selected for task } i, i = 1,..,n \quad k = 1,..,m_i \\ 0 & \text{otherwise} \end{cases}$$

- Time Related Decision Variables

$S_i = \text{start time of the task } i, i = 1,..,n$

$T_i = \text{tardiness of the task } i, i = 1,..,n$

$$T_i = Max\{0, S_i + \sum_{k=1}^{m_i} p_{ik}x_{ik} - d_i\}$$

There are $\sum_{i=1}^{n} m_i$ binary x_{ik} variables and $2n$ (n for S_i + n for T_i) continuous variables.

We have five constraint sets each of which is explained below.

1. Mode Assignment Constraints

Each task should be assigned to one of its defined modes.

$$\sum_{k=1}^{m_i} x_{ik} = 1 \quad i = 1,..,n \quad (1)$$

2. Precedence Constraints

Finish-to-Start Relations

Task j can start after lag_j units of completion of task i for $(i,j) \in FS$.

$$S_i + \sum_{k=1}^{m_i} p_{ik} x_{ik} + \ lag_j \leq \ S_j \ for \ (i,j) \in FS \quad (2)$$

Start-to-Start Relations

Task j can start after lag_j units of the start of task i for $(i,j) \in SS$

$$S_i + \ lag_j \ \leq \ S_j \ for \ (i,j) \in SS \quad (3)$$

3. Project Start Time

Project starts at time zero, i.e., the start time of dummy task 0 is zero.

$$S_0 = 0 \quad (4)$$

4. Tardiness Related Constraints

The total tardiness of the tasks cannot exceed D.

$$T_i \geq S_i + \sum_{k=1}^{m_i} p_{ik}x_{ik} - d_i \; for\; all\; i \quad (5)$$

$$T_i \geq 0 \quad for\; all\; i \quad (6)$$

$$\sum_{i=1}^{n} T_i \leq D \quad (7)$$

5. Binary Constraints

$$0 \leq x_{ik} \leq 1 \; and \; integer \; i = 1,..,n \quad k = 1,..m_i \quad (8)$$

Our objective is to minimize total cost over all activities and stated below.

$$Min \sum_{i=1}^{n}\sum_{k=1}^{m_i} c_{ik}x_{ik} \quad (O1)$$

where $c_{ik} = p_{i0} - p_{ik}$

We hereafter refer to our problem as P_1, where P_1 is stated as

$Min\; O1$ s.t. Constraint Sets (1) through (8).

P_1 is a Discrete Time/Cost Trade-off Problem (DTCTP) with total tardiness constraint.

4.2 A Heuristic Algorithm – Two-Step Heuristic

(De *et al.* 1997) show that all versions of the Discrete Time/Cost Trade-off Problem (DTCTP) are NP Hard in the strong sense so is our DTCTP with total tardiness constraint. This justifies the use of heuristic algorithm to find high quality solutions in reasonable solution times. In this study, we develop a heuristic algorithm that uses the optimal solution of the Linear Programming Relaxation (LPR) of our model, P_1. The name of our heuristic algorithm is Two-Step Heuristic.

The optimal LPR solution is found by relaxing the integrality constraints on the binary variables, hence using the following constraint and solving the resulting model optimally.

$$0 \leq x_{ik} \leq 1 \;\; i = 1,..,n \;\; k = 1,..,m_i$$

Our heuristic procedure consists of two phases: construction and improvement. In the construction phase, we find a feasible solution, and in the improvement phase, we try to decrease the total cost of the solution while maintaining its feasibility.

We illustrate the general schema of the Two-Step Heuristic in Figure 4.1, below.

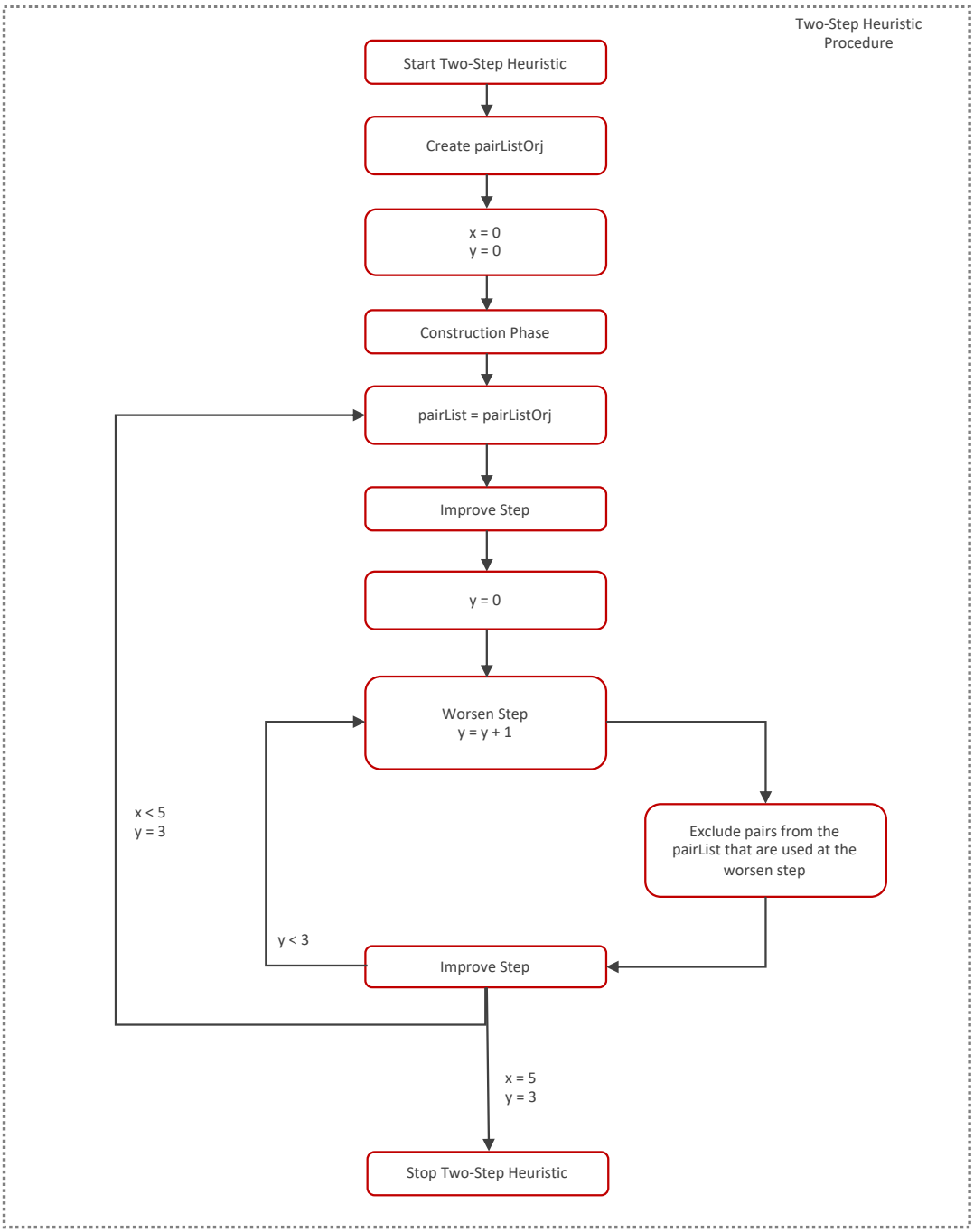

Figure 4.1 A General Scheme of Two-Step Heuristic Procedure

4.2.1 Construction Phase

We find the solution by repairing the optimal solution of the LPR of P_1. The LP relaxed model uses continuous decision variables, hence may produce partial mode assignments. We keep the integer mode assignments and move the partial mode assignments to their next smaller time mode. In doing so, we keep feasibility (as the times are decreased) and increase the total cost as low as possible (as the next smaller times are taken).

Below is the formal description of our construction phase.

Let

$$x_{ik}^{LP} = \text{optimal LP relaxed assignment variable}$$

$$p_i^{LP} = \sum_{k=1}^{m_i} p_{ik}\, x_{ik}^{LP}$$

$$c_i^{LP} = \sum_{k=1}^{m_i} c_{ik}\, x_{ik}^{LP}$$

If x_{ik}^{LP} is fractional for any k then we set its task to mode r_i such that

$$p_{ir_i} \leq p_i^{LP} < p_{ir_{i+1}}$$

Equivalently

$$c_{ir_i} \geq c_i^{LP} > c_{ir_{i+1}}$$

We illustrate the construction phase of the Two-Step Heuristic in Figure 4.2.

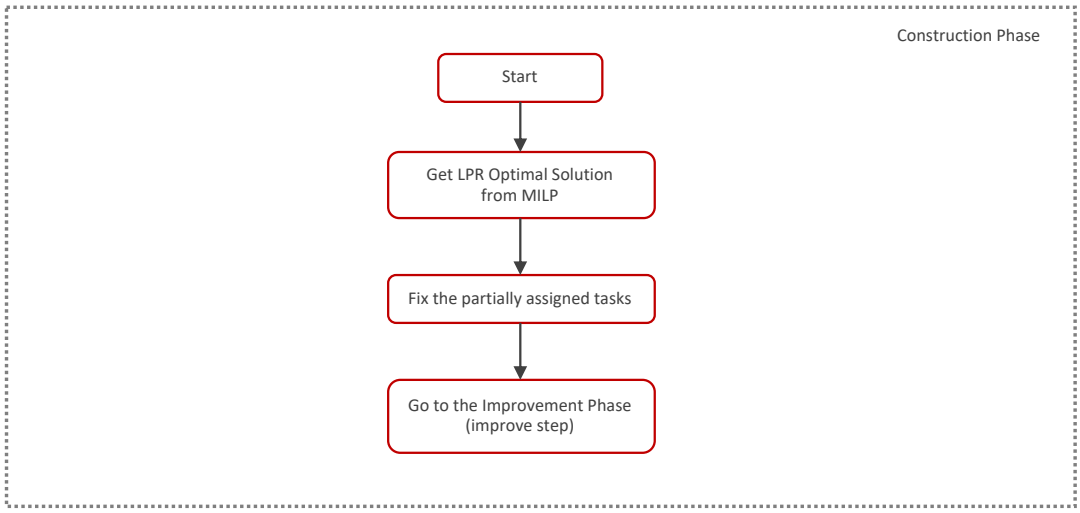

Figure 4.2 Two-Step Heuristic Construction Phase

4.2.2 Improvement Phase

In the improvement phase, we improve the schedule found by the construction phase. The improvement phase consists of two sub-steps. We improve the schedule, i.e., reduce its total cost value, while keeping the feasibility, at the first step. When further improvements are not possible, we let the schedule worsen within the feasible region with the hope of reducing the total cost value in the subsequent iterations. We keep the incumbent solution at hand throughout the process.

While realizing all interchanges, we check the feasibility using the Critical Path Method (CPM). Recall that the CPM produces the earliest possible completion times for the tasks for a given set of mode assignments. We use the method to check the feasibility of the total tardiness constraint. For any other schedule, one cannot find smaller task completion times, thereby smaller tardiness for any task, thereby smaller total tardiness over all tasks.

We use a list of task pairs, which consist of two tasks on the same path, to improve the schedule found by the construction phase. The main idea of the improvement phase is that while increasing one of the task's task time, we can decrease the other task's task time simultaneously to stay in the feasible region and move into a better

spot. The main goal is to increase any task's task time (which means decreasing the cost) and maintain feasibility (which means staying below the preset total tardiness value).

We need to select the task pairs during the improvement step, which supports our goal and brings improvement to our schedule at hand. For this purpose, we calculate the benefit of a pair as follows.

$$Dr_t = \max\left(Dr_{t_1}, Dr_{t_2}\right)$$

where t indicates the pair, Dr_{t_1} indicates the benefit if we increase the task time (decrease the cost) of the first task and the Dr_{t_2} indicates the benefit if we increase the task time (decrease the cost) of the second task of the pair, respectively.

$$Dr_{t_1} = sum\ of\ the\ current\ costs\ of\ two\ tasks\ in\ the\ pair\ t - (next\ smaller\ cost\ of\ first\ task + next\ larger\ cost\ of\ the\ second\ task)$$

$$Dr_{t_2} = sum\ of\ the\ current\ costs\ of\ two\ tasks\ in\ the\ pair\ t - (next\ larger\ cost\ of\ first\ task + next\ smaller\ cost\ of\ the\ second\ task)$$

If the task is already at its maximum or minimum task time option, we cannot increase or decrease its task time and let $Dr_{t_1} = 0$ or $Dr_{t_2} = 0$, respectively.

The pairs with positive values are rewarding since they help us improve our schedule.

We use several Dr_t modes to calculate values which are explained below.

1. current task time, next second larger task time
2. current task time, next larger task time
3. next smaller task time, next larger task time
4. next second smaller task time, next larger task time
5. next smaller task time, second next larger task time

At the first step of the improvement phase, we evaluate the pairs without any specific order (the first one in the pair list comes first). First, we use mode 2 for our pair at hand, which means using the current task time of the one task and the next larger task time of the other task of the pair. (Note that deciding which task to increase its task time depends on the Dr_{t_1}, Dr_{t_2} values explained above.). If the Dr_t value is positive, using the CPM, we check the feasibility of the schedule. If the schedule is feasible, we accept the task time changes. If the schedule is not feasible, we try other modes to calculate Dr_t values for the same pair, hoping that we can acquire a positive Dr_t value to work with.

After we try mode 2 and either cannot find a feasible schedule or positive Dr_t, we give a chance to the same pair by using mode 3. The same procedure applies to mode 3. Subsequently, we try mode 4 and mode 5 to look for interchanges, which also yields a feasible schedule. If the pair does not improve at the end of mode 5, we pass the pair and try the next one. Remember that we start with mode 2 and continue in the order of mode 3, 4, and 5 when the modes give positive Dr_t values but infeasible schedules. If the schedule is also feasible, we accept the changes and try the next pair to check for additional improvement.

On the other hand, we have a second branch at mode 3. If the Dr_t value is negative for the pair t when we apply mode 3, we try mode 5 directly. The aforementioned procedure continues until every pair in the pair list is checked for improvement. Lastly, we try mode 1 for every pair starting over the top of the pair list with the hope that there may be a pair that have a task already at its minimum task time option (therefore we couldn't improve), but the other task may have not its maximum (longest) task time option. We repeat this procedure for a predefined number of iterations. Also, to avoid unnecessary iterations during the first step of the improvement phase, we control if there is an improvement compared to the previous iteration's cost objective. If we cannot see any improvement, we stop the first step and continue with the second step of the improvement phase. This procedure completes the first step of the improvement phase.

We illustrate the improve step of the improvement phase in Figure 4.3.

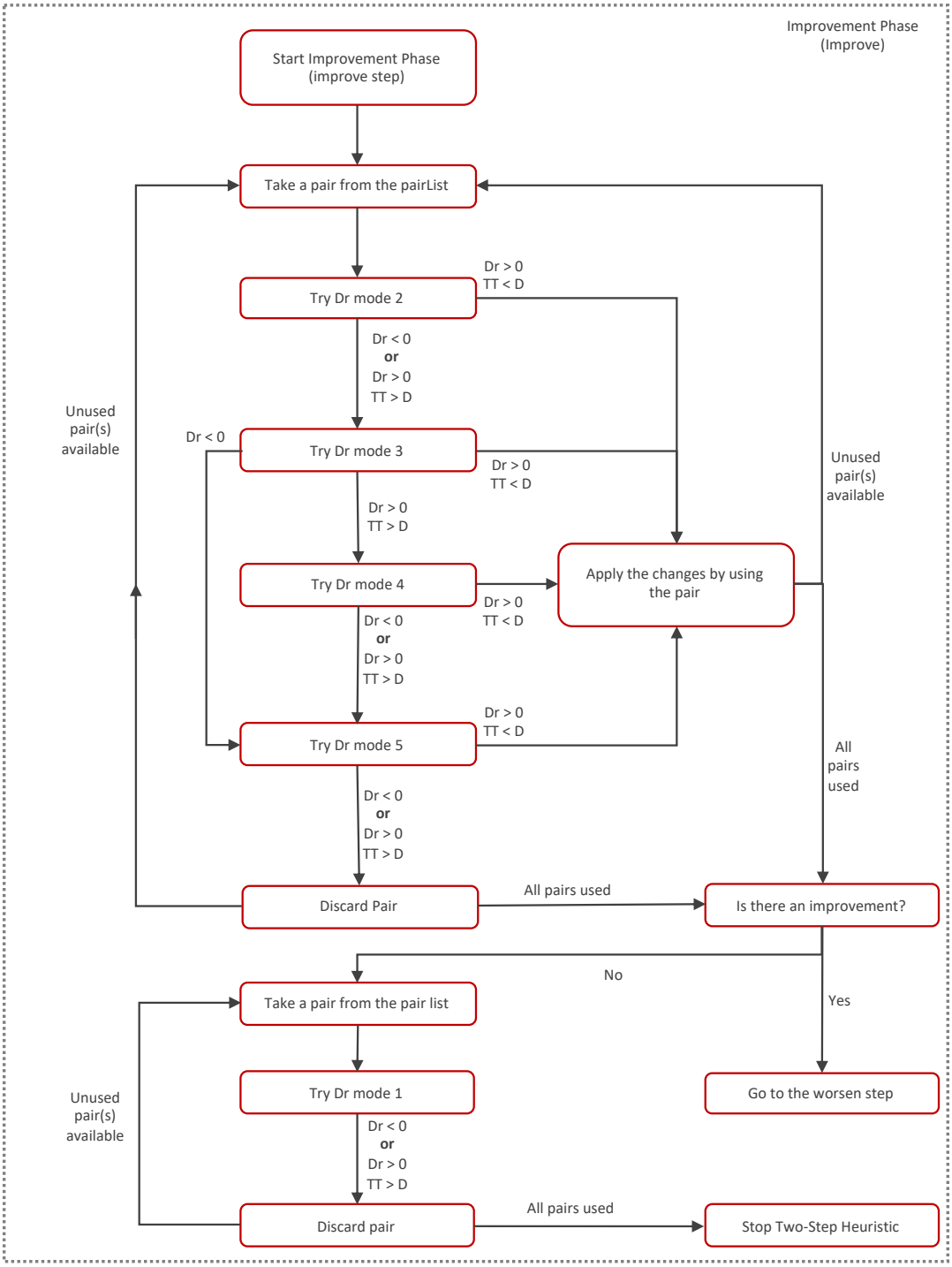

Figure 4.3 Two-Step Heuristic Improvement Phase (Improve Step)

We let the schedule worsen at the second step of the improvement phase by using 2 pairs. At this stage, we select the two pairs with the help of Dr_t values. Unlike the first step of the improvement phase explained above, we calculate Dr_t value for all pairs at once (without accepting any changes). Then, we select the pair with the maximum Dr_t values among negative ones. We apply the interchanges and accept the changes if we are still in the feasible region. If not, we exclude the pair and select another one. This procedure continues until we find two acceptable pairs, which worsen the schedule at hand within the feasible region. The reason for selecting the number of pairs to worsen the schedule as two is not the lead schedule to the non-promising area while escaping the local optimal values.

After we successfully worsen the schedule by using two pairs, we again apply the first step of the improvement phase.

We illustrate the worsen step of the improvement phase in Figure 4.4 below.

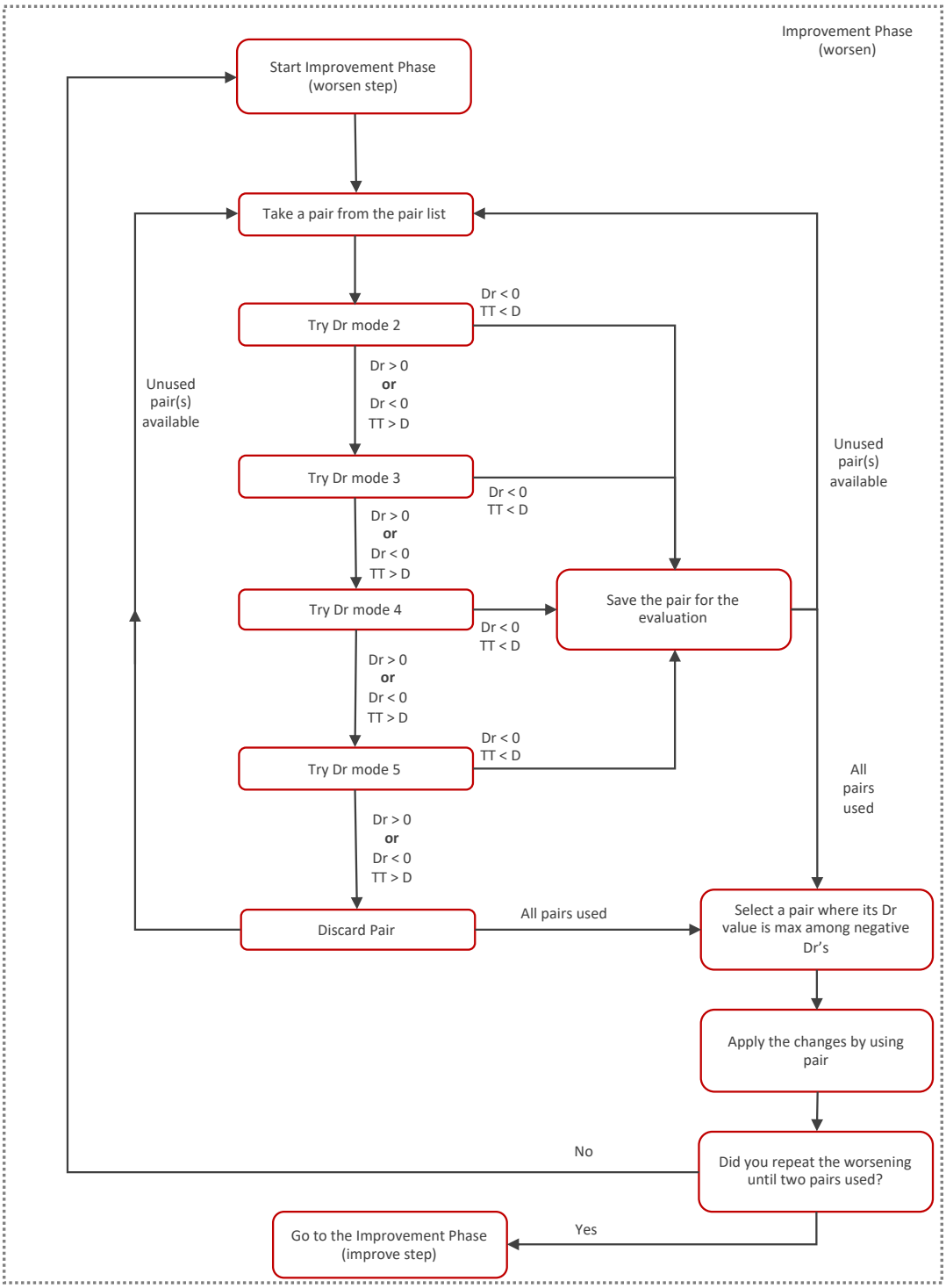

Figure 4.4 Two-Step Heuristic Improvement Phase (Worsen Step)

We apply the improvement phase 5 times and visit the worsen step 3 times for every improvement phase applied unless we stop the algorithm since we do not see further improvement. These stopping criteria are defined based on the preliminary experiments.

CHAPTER 5

GENERATION OF NONDOMINATED OBJECTIVE VECTORS

In this section, we generate nondominated objective vectors considering two conflicting objectives to minimize: total cost of reduction, Z^C, and total tardiness, Z^T. The corresponding model is as follows:

Min $\{Z^C, Z^T\}$ s.t. $x \in X$

This model is formed by relaxing Constraint (7), the tardiness constraint of Model P_1, and defining it as objective. The existence of the second objective leads to many nondominated points. Dominance and efficiency terms are counterparts of each other in the objective and decision spaces, respectively. An efficient solution is a solution such that there is no other solution that is at least as good as it in both objectives and strictly better than it in at least one objective. That is, a decision vector $s \in X$ is efficient if there does not exist $s' \in X$ satisfying $Z_s^C \leq Z_{s'}^C$ and $Z_s^T \leq Z_{s'}^T$ with a strict inequality in at least one. Otherwise, s is inefficient. If s is efficient (inefficient), then the corresponding objective vector (Z_s^C, Z_s^T) is nondominated (dominated). The set of all nondominated vectors is referred to as Pareto front.

We generate the Pareto front exactly and approximately in Sections 5.1 and 5.2, respectively. In Section 5.3, we illustrate the approaches to MOH projects.

5.1 Model-based Procedure

We apply an augmented version of the ε-constraint method by Haimes et al. (1971) to find all nondominated objective vectors. In the bi-objective ε-constraint method, one of the objectives is optimized and the other is bounded as a constraint. All nondominated objective vectors are generated by changing the bound in the constraint systematically. In our implementation, without loss of generality, we

optimize Z^C and put a bound on Z^T. Thus, we replace the objective function of Model P_1 with Min $Z^C + \mu Z^T$ and solve it iteratively by changing the right-hand-side value of Constraint (7). We use the augmentation in the objective function, μZ^T, to break the ties in favor of minimizing total tardiness to find a nondominated point. That is, when there are alternative objective vectors having the same Z^C as the optimal, the augmented part enables to end up with the one having minimum Z^T among the alternatives.

We note that a sufficiently small μ value should be used to guarantee that the augmentation does not cause any trade-off with Z^C and it only has the effect of breaking ties.

Let

$(Z^T)_{max}$ = maximum possible Z^T value = total tardiness value returned by the CPM with maximum task time options

$(Z^T)_{min}$ = minimum possible Z^T value = total tardiness value returned by the CPM with minimum task time options

Using $(Z^T)_{min}$ and $(Z^T)_{max}$ we set the following relation for a sufficiently small value of μ:

$Z^C + \mu(Z^T)_{max} \leq Z^C + 1 + \mu(Z^T)_{min}$ (as the Z^C value should not increase even one unit for the maximum reduction of the Z^T value).

This follows,

$\mu\ ((Z^T)_{max} - (Z^T)_{min}) \leq 1$

$\mu \leq \frac{1}{((Z^T)_{max} - (Z^T)_{min})}$

We set μ to $\frac{1}{((Z^T)_{max} - (Z^T)_{min}) + 1}$.

For the sake of completeness, we next provide the steps of the procedure.

Let (Z_*^C, Z_*^T) be the optimal objective vector of the corresponding model and P denote the set of nondominated points. Recall that D is the bound on Z^T.

Algorithm 1: Finding all nondominated objective vectors

Step 0 (Initialization)

Set $P \leftarrow \emptyset$

Set $\mu \leftarrow \frac{1}{((Z^T)_{max} - (Z^T)_{min}) + 1}$

Solve Min $Z^C + \mu Z^T$ s.t. $x \in X$

Set $P \leftarrow P \cup (Z_*^C, Z_*^T)$ and $D \leftarrow Z_*^T - 1$

Step 1 (Finding nondominated objective vector and a corresponding efficient solution)

Solve Min $Z^C + \mu Z^T$ s.t. $Z^T \leq D,\ x \in X$

If the model is infeasible, go to Step 2; otherwise set $P \leftarrow P \cup (Z_*^C, Z_*^T)$, $D \leftarrow Z_*^T - 1$ and repeat Step 1

Step 2 (Termination)

Stop, all nondominated objective vectors (in P) and an efficient solution representing each objective vector are found.

We solve Min $Z^C + \mu Z^T$ s.t. $Z^T \leq D,\ x \in X$ to find a new nondominated objective vector in Step 1. We keep updating D and solving the model until it becomes infeasible. Whenever the model is infeasible, we stop searching as there is no more nondominated point available. We utilize the integrality property of objective vectors and decrement the optimal total tardiness value, Z_*^T, by 1 while updating D.

.

5.2 Evolutionary Algorithm

Evolutionary Algorithm (EA) is a metaheuristic approach based on the genetic processes of biological organisms. EA tries to mimic the survival of the fittest of nature and applies it to the field of solving problems. One well-known multi-objective metaheuristic is the NSGA-II algorithm proposed by Deb et al. (2002).

NSGA-II has been developed for multi-objective problems. In this algorithm, each solution is compared with every other solution in the population, and then solutions are placed to the corresponding frontiers according to their dominance/nondominance relation among each other. That is, all nondominated objective vectors are placed to the first front. Then, all nondominated objective vectors of the resulting population (excluding the ones on the first) are placed to the second front, and so forth. In addition, NSGA-II uses a crowded-comparison approach to ensure diversity and density evaluation. As for diversity preservation, crowding distance comparison guides the selection process toward a uniformly spread-out Pareto front.

NSGA-II maintains a parent population and generates offspring population using selection, recombination, and mutation operators. Then, parent and offspring populations are combined to form a global population. In the global population the frontiers are formed and the crowding distances of the solutions are calculated. Then, solutions in each frontier are sorted in descending order by considering the crowding distance value. Finally, solutions are selected starting from the first frontier until the initial population size is reached. These steps are repeated for a predefined number of generations. This approach aims to find better solutions in each iteration and pass these good solutions to the next generations.

NSGA-II algorithm is adapted and modified for our problem. The chromosomes that are used in the algorithm and details of the related algorithm are explained in the following sections.

5.2.1 Chromosome Representation Approach

Recall that some tasks have more than one task time option in our projects. Shorter task times can be used to crush some tasks, if required, to meet the due dates. Therefore, it is proper to represent the chromosomes according to the task time options, i.e., 0, 1, and 2.

Tasks that have more than one task time option are included in the chromosomes. The selected representation schema does not include tasks having only one task time option. Hence, feasibility is guaranteed without additional mechanisms. To exemplify, suppose that a gene representing a task is assigned to the second task time although the task does not have more than one task time option. This situation yields an infeasible solution. The representation schema used in our EA avoids these kinds of infeasibilities and shortens the chromosome length.

A gene represents time option of the task of the project.

A chromosome represents a solution for the project scheduling problem.

A solution of the project shows which task time option is used for the tasks if they have any task time reduction option.

The chromosome representation is illustrated in the following instance where 0 is the original task time of the tasks.

1 – first level task time reduction option

2 – second level task time reduction option

Example:

Table 5.1 Example Chromosome Structure

Task No	2	4	5	8
Chromosome 1	0	2	1	0
Chromosome 2	1	0	1	2

- Suppose that tasks 2, 4, 5, and 8 have more than one time option.
- Tasks 2 and 5 have 2 different time options meaning that the genes standing for tasks 2 and 5 can take the value of either 0 or 1.
- Tasks 4 and 8 have 3 different time options meaning that the genes standing for tasks 4 and 8 can take the value of 0, 1, and 2.

5.2.2 Evolutionary Algorithm Operators and Parameter Settings

Representation schemes, parameter settings, and the selection of suitable operators play a significant role on the success of the EAs. We next explain the selection of crossover and mutation operators.

Three most common crossover operators are 1-point crossover, 2-point crossover, and uniform crossover (Lim *et al.* 2017, Pinho and Saraiva 2020). In our study, we use 1-point crossover as it is more suitable for our chromosome representation. 2-point crossover is preferred when the chromosome is viewed as a loop rather than a string. We prefer 1-point crossover to uniform crossover as building blocks are less disrupted by 1- or 2- point crossover.

In the application of the 1-point crossover, we select two individuals from the population randomly. Then, the cut point at which the crossover is performed is selected randomly. That is, the parents are divided into two parts with respect to the

selected cut point. Lastly, we combine the first part of the first parent and the second part of the second parent to create the first child, and the second part of the first parent and the first part of the second parent to create the second child. We repeat this operation for half of the population size times. The following example illustrates the generation of offspring from the parents.

Parent 1 = 0 2 | 1 0
Parent 2 = 1 1 | 1 2

Child 1 = 0 2 | 1 2
Child 2 = 1 1 | 1 0

The mechanism of the mutation operator is explained as follows. First, we define the number of individuals to be used for the mutation. In our study, it is set to 0.2 of the population size based on the preliminary experiments. We randomly select the individuals for mutation. For each selected individual, we choose three genes randomly. During the mutation of a gene, we assign a randomly selected task time for the gene different than the current one within the task times of the corresponding task.

Additionally, we included the extreme solutions, which are the solutions with maximum total tardiness and maximum total cost, in the initial population as seeds.

5.3 Illustration on MOH Projects

In our bi-objective problem setting, the objectives are total tardiness and total cost, and Pareto fronts of MOH projects are generated exactly and approximately. In each project, the exact Pareto front is achieved by Algorithm 1 of Section 5.1 and approximation is obtained by the EA explained in Section 5.2.

Population sizes 30, 40, 50, and 60 are illustrated for both of our real-life projects. We run our experiments for the 100, 150, 200, and 250 generations to observe the convergence of the generated efficient frontiers. For all of the experiments illustrated below, we selected (population size * 0.2) the number of individuals for the mutation and mutated three genes of every selected individual. We produced two individuals from every randomly selected two parents from the population and repeated the crossover process for (population size / 2) times.

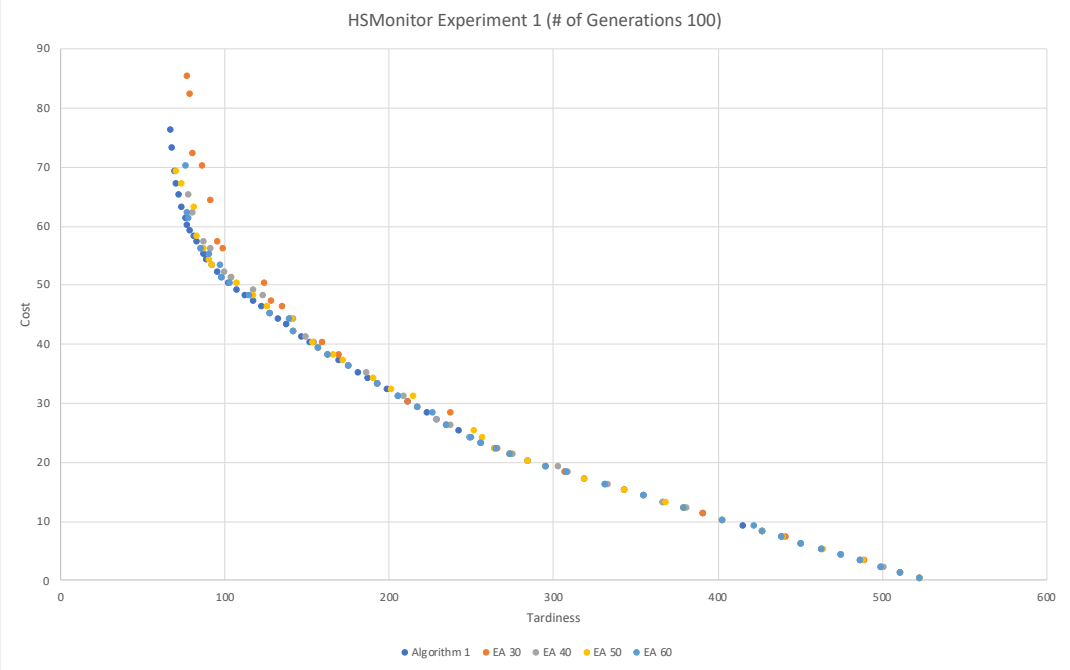

Figure 5.1 HSMonitor Experiment 1 (# of Generations: 100)

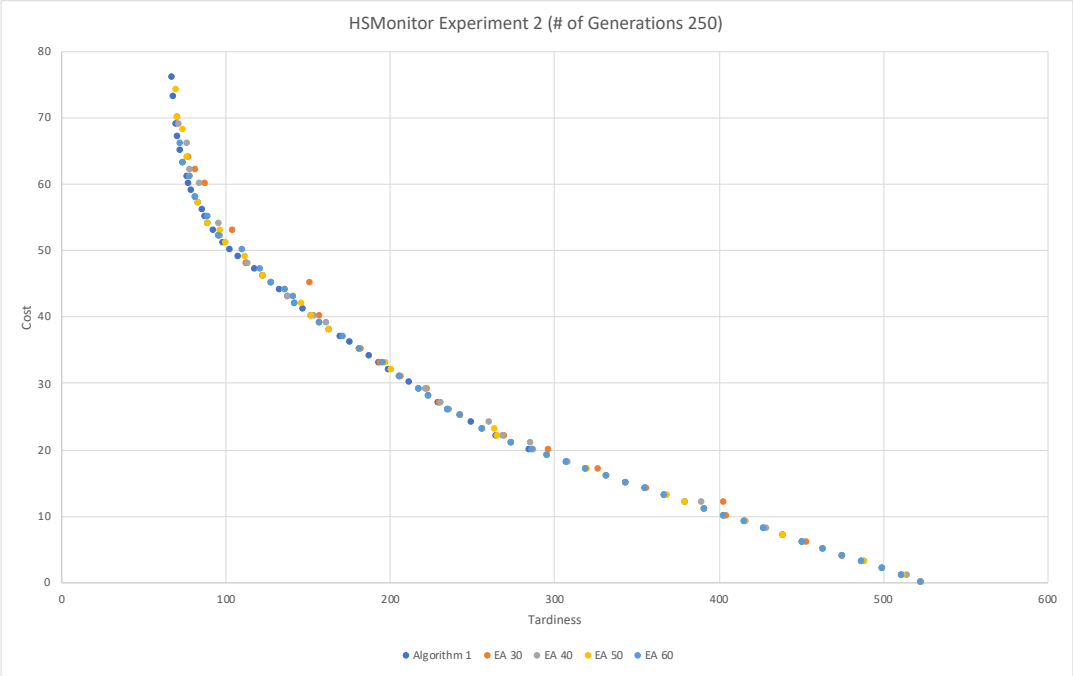

Figure 5.2 HSMonitor Experiment 2 (# of Generations: 250)

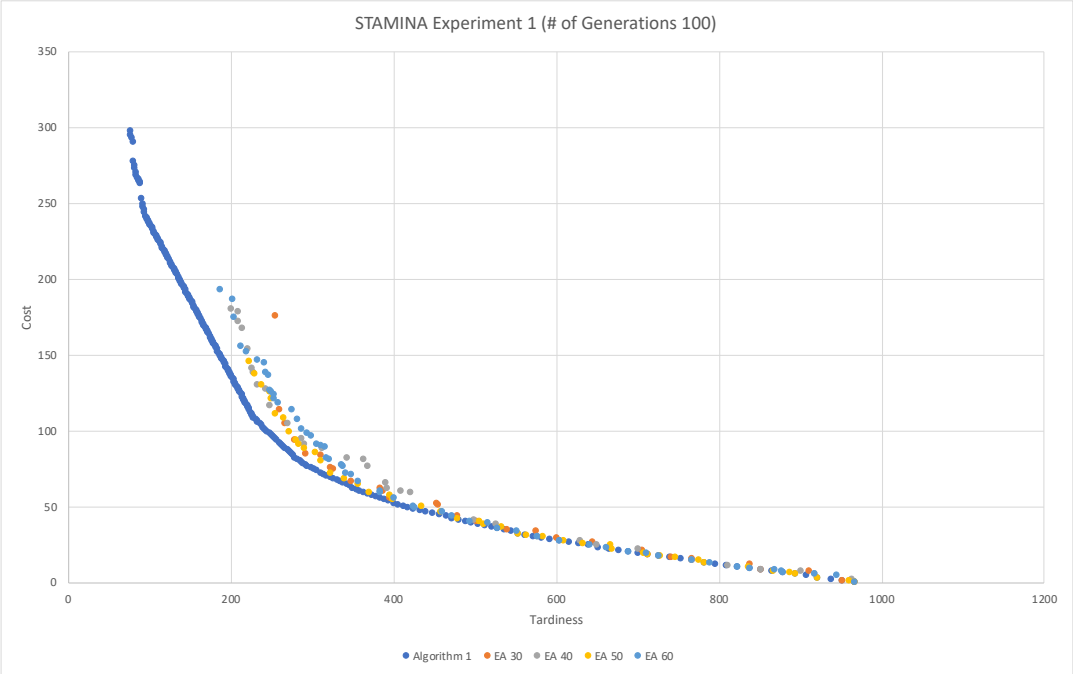

Figure 5.3 STAMINA Experiment 1 (# of Generations: 100)

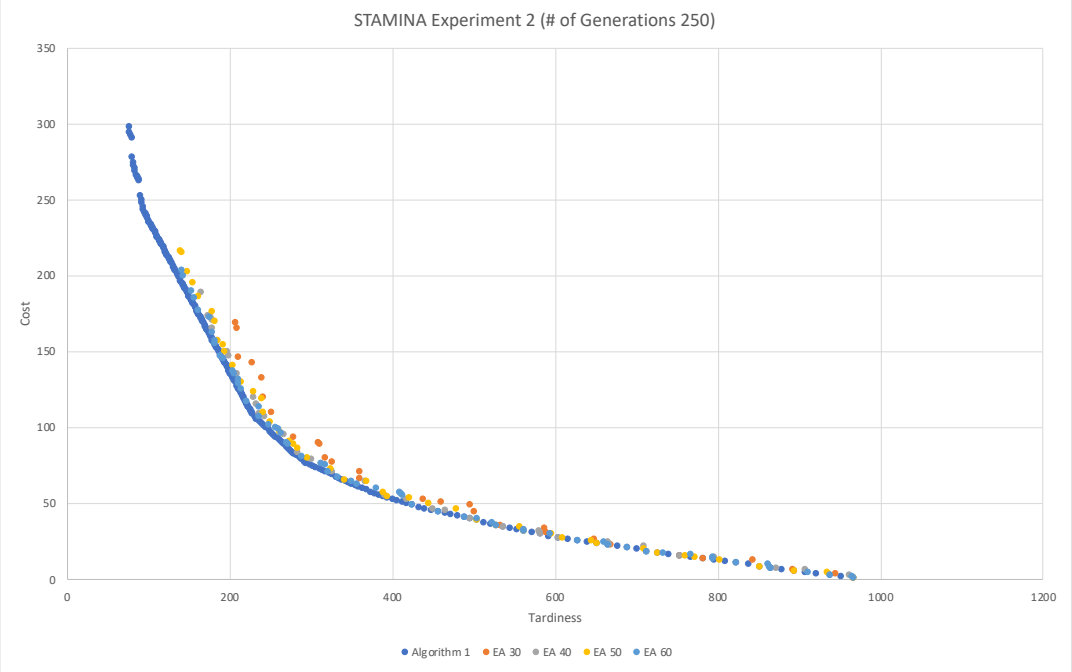

Figure 5.4 STAMINA Experiment 2 (# of Generations: 250)

Table 5.2 CPU Times Algorithm 1 vs EA

Algorithm 1 CPU Time		*EA CPU Time*		*Number of Generations*			
				100	***150***	***200***	***250***
7.22 (68 solutions)	HSMonitor	*Population Size*	***30***	3.08	4.47	5.97	7.34
			40	4.33	6.07	8.49	10.54
			50	5.53	7.89	10.78	13.61
			60	6.57	9.99	13.01	16.14
Algorithm 1 CPU Time		*EA CPU Time*		*Number of Generations*			
				100	***150***	***200***	***250***
23.96 (261 solutions)	STAMINA	*Population Size*	***30***	3.31	5.14	6.67	8.35
			40	4.61	6.9	9.09	11.43
			50	5.87	8.81	11.76	14.75
			60	7.27	11.07	14.48	18.48

Table 5.3 Number of Optimal Solutions Found by EA

		# of Generations: 100				***# of Generations: 250***			
	Population Size	***30***	***40***	***50***	***60***	***30***	***40***	***50***	***60***
HSMonitor	***# of optimal solutions found***	11	12	17	27	6	13	17	32
	# of unique solutions found	29	32	39	40	25	35	36	42
STAMINA	***# of optimal solutions found***	4	4	6	9	4	8	8	12
	# of unique solutions found	30	35	45	57	30	38	46	62

Parameter setting for the heuristic algorithms plays an essential role in both the solution times expressed in CPU (Central Processing Unit) seconds and the solution quality. In our case, increasing the population size and the number of generations yields an increase in the CPU time of the algorithm. On the other hand, increasing these two parameters produces better results in better convergence to the exact Pareto front.

The dark blue points on Figures 5.1 through 5.4 represent the exact Pareto front found by Algorithm 1. Orange, grey, yellow, and blue colors represent the results for population sizes of 30, 40, 50, and 60, respectively. The comparison of Figures 5.1 and 5.2 shows that increasing the number of generations and the initial population improves the performance of the EA. When we compare the Figure 5.1 and Figure 5.2, Figure 5.2 represents improved output. Increasing the number of generations and the initial population resolves the high fluctuation from the Pareto front at an expense of increased CPU time.

Similar observations hold for STAMINA project illustrated in Figures 5.3 and 5.4. Increasing the parameters improves the quality of the solutions.

Additionally, Table 5.3 represents the number of exact solutions found by the EA. We observe that increasing the number of generations and the population size increases the number of exact solutions found by the EA for most cases.

Recall that in our case, the project manager leads the project team to complete the projects within the pre-planned time and cost limitations. Therefore, generating the whole Pareto front offers a valuable projection for the overall project.

We observe that the EA finds representative objective vectors from different portion of the Pareto front, hence the project manager can select a proper solution according to all target tardiness ranges. The project manager can check and analyze different objective vectors which in turn enables him/her to focus on any specific portion of the Pareto front to select the proper strategy.

We also observe that for small tardiness values, the marginal cost for one unit decrease in tardiness is high, whereas for large tardiness values the marginal cost is relatively small. Therefore, the project manager can focus on a promising section where the gained time is worthy of the target cost.

CHAPTER 6

COMPUTATIONAL EXPERIMENTS

In this chapter, we discuss the results of our experiment that is designed to test the performances of our mixed integer linear programming (MILP) model and heuristic algorithm. Section 6.1 presents our data generation scheme, Section 6.2 gives the measures used to evaluate the performance, and Section 6.3 discusses the performance of the algorithms.

6.1 Data Generation

Recall that the following parameters characterize a problem instance.

1. Precedence network (precedence relations of tasks with/without lags)
2. Processing modes (task time and cost pairs) of the tasks
3. Due dates of the tasks

In our experiments we use 3 data sets: i) two real-life projects' networks, ii) large-sized networks from literature, and iii) small-sized networks from literature. The first data set includes the networks of HSMonitor and STAMINA, which are two real projects of the MOH. These projects' data (instances) have their own real parameter values.

We selected 4 large-sized networks (with 87-138 tasks) and 3 small-sized networks (with 31-40 tasks) from the data sets given in (Akkan *et al.* 2005). For each n value, there are three instance types. The types differ by their task times and the number of modes. Hence a total of 21 instances are taken from the literature.

Our problem additionally uses due dates. We aim to generate due dates similar to the two real-life projects. In our real-life projects, tasks with due dates are the deliverables that require input from several tasks, hence the deliverables are observed

frequently in the paths. We assign due dates to these frequently observed tasks to mimic this pattern. Accordingly, we generate several paths and count the occurrences of the tasks on the paths. We select the top 30 percent of the tasks, which are commonly observed within the paths, and set due dates only to those tasks. In doing so, we schedule using the longest task times and find the latest possible completion times. Thereafter, we assign a defined percentage of the completion times as due dates. We create instances that neither have nonrealistic and lazy due dates nor too tight and unrealizable due dates. To see the effect of loose and tight due dates on the performance, two percentages of the completion times, 60% and 30%, are used. We hereafter refer 60% and 30% sets as DueDate Set 1 and DueDate Set 2, respectively.

Five different *D* (maximum allowable total tardiness) values are selected using the following procedure. We generate the efficient frontier of the instance using two hours termination limit. We take *D* values of the solutions that have the first 5 longest Central Processing Unit (CPU) times. When we cannot generate the whole efficient frontier in two hours, we choose the *D* values among the partial efficient frontier, which we can generate within the time limit.

Table 6.1 below represents the instance status over finding the complete efficient frontier within the time limit. The table indicates "Yes" if we can find the efficient frontier within the time limit and "No" otherwise. The numbers represent the number of solutions found within the time limit.

Table 6.1 Efficient Frontier Generation in 2 hours

n	***Instance Type***	***CPU Time DueDate Set 1***	***CPU Time DueDate Set 2***
32	1	572 (Yes)	753 (Yes)
	2	575 (Yes)	996 (Yes)
	3	473 (Yes)	759 (Yes)
36	1	481 (Yes)	625 (Yes)
	2	591 (Yes)	977 (Yes)
	3	610 (Yes)	730 (Yes)

Table 6.1 Efficient Frontier Generation in 2 hours (continued)

40	1	550 (Yes)	339 (**No)**
	2	608 (Yes)	381 (**No)**
	3	530 (Yes)	232 (**No)**
87	1	590 (Yes)	416 (Yes)
	2	553 (Yes)	392 (Yes)
	3	614 (Yes)	387 (Yes)
104	1	503 (**No)**	167 (**No)**
	2	447 (**No)**	170 (**No)**
	3	427 (**No)**	151 (**No)**
121	1	340 (**No)**	98 (**No)**
	2	392 (**No)**	158 (**No)**
	3	364 (**No)**	281 (**No)**
138	1	500 (**No)**	87 (**No)**
	2	583 (**No)**	131 (**No)**
	3	303 (**No)**	82 (**No)**

As we can observe from Table 6.1, we have found the complete efficient frontier for the instances in the small-sized data set for both DueDate Set 1 and DueDate Set 2 except the instances with n equals 40 with the tight due date set. Table 6.1 also shows that we could not find the complete efficient frontier for large-sized data set except for the instances with 87 tasks. Therefore, we can conclude that the *D* values are representative of the instances where we can generate the complete efficient frontier.

We have 21 instances retrieved from the literature. We select 5 different *D* values with 2 different due dates and get 220 problem instances. For the real-life projects, we use real due dates with 5 *D* values, hence getting a total of 10 instances. Thus, our data sets include a total of 230 problem instances.

We solve our mathematical model and its LPR using Gurobi Optimizer with Python interface. The algorithms are also coded in the Python programming language. We

run our experiments on a macOS Catalina operating system with a 2.3 Ghz Quad-Core Intel Core i7 Processor, 32 GB Memory (RAM).

6.2 Performance Measures

We use the following measures to evaluate the performance of the MILP.

1. Average CPU time (in seconds) of 5 instances (5 D values)
2. Maximum CPU time (in seconds) over 5 instances

For the heuristic procedure, additional deviation based performance measures are defined below:

1. Average deviation of the heuristic solution from the optimal solution by the MILP over 5 instances

   A deviation of an instance is defined as

   $$\frac{(Z\ value\ of\ the\ Heuristic\ Procedure - Optimal\ Z\ value)}{Optimal\ Z\ Value} * 100$$

2. Maximum deviation over 5 instances

Number of times the heuristic finds the optimal solution (the deviations are zero), out of 5 instances.

6.3 Analysis of the Results

In this section, we discuss the performance of the mathematical model and the heuristic approach.

First, we give the performance results for the real-life MOH projects. Table 6.2 reports the CPU times of the MILP and the heuristic procedure, whereas Table 6.3 reports the deviation based performance measures for the heuristic procedure.

Table 6.2 The CPU times of the MOH projects for 5 instances (5 *D* values)

n	MOH Projects	MILP		Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum	Average	Maximum
135	HSMonitor	4.56	5.80	0.03	0.04	4.20	4.81
155	STAMINA	0.74	0.79	0.07	0.08	4.97	5.39

Table 6.3 The Deviations of the MOH projects for 5 instances (5 *D* values)

n	MOH Projects	Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum
135	HSMonitor	3.51	12	0(5)*	0
155	STAMINA	0.74	3.7	0(5)	0

*The numbers in parentheses give the number of times the heuristic finds the optimal solution (out of 5)

Note from the above tables that the Two-Step Heuristic finds an optimal solution to each of the 10 instances within 5 seconds. We observe the satisfactory behavior of the heuristic for the construction phase as well. The respective deviations are 3.51% and 0.74% for 135 and 155 task instances. This indicates that the LPR of the model is quite satisfactory. Due to this satisfactory behavior, we observe small CPU times spent by the MILP.

We next investigate the performances on the 220 instances taken from the literature and report the associated results in Tables 6.4 through 6.11. Tables 6.4 and 6.6 indicate the performance measures of the small-sized instances with DueDate Set 1. Tables 6.5 and 6.7 represent the large-sized instances' analysis results with DueDate Set 1. Tables 6.8 and 6.10 illustrate the CPU times, and Tables 6.9 and 6.11 represent the deviations with the DueDate Set 2 for small and large-sized instances, respectively.

Improvement Phase CPU times indicated in Tables 6.2, 6.4, 6.5, 6.8, 6.9 also include the construction phase CPU times.

Table 6.4 The CPU Times _ Small Sized Instances -- DueDate Set 1 for 5 instances (5 *D* values)

n	Instance No	MILP		Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum	Average	Maximum
32	1	12.84	20.04	0.03	0.03	0.42	0.51
	2	5.41	6.60	0.03	0.03	0.26	0.32
	3	24.44	39.65	0.03	0.03	0.40	0.44
36	1	41.30	57.78	0.03	0.03	0.50	0.52
	2	6.72	9.34	0.03	0.03	0.61	0.74
	3	5.27	5.44	0.03	0.03	0.46	0.64
40	1	1148.38	1767.30	0.04	0.04	0.57	0.58
	2	816.59	982.08	0.04	0.04	0.81	0.87
	3	726.99	2037.13	0.04	0.04	0.67	0.78

Table 6.5 The CPU Times _ Large Sized Instances -- DueDate Set 1 for 5 instances (5 *D* values)

n	Instance No	MILP		Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum	Average	Maximum
87	1	18.35	35.07	0.09	0.10	2.12	2.54
	2	128.14	165.65	0.09	0.11	2.49	2.95
	3	91.28	190.15	0.09	0.10	2.27	3.14
104	1	628.40	1216.42	0.11	0.12	4.27	4.85
	2	803.58	1225.69	0.11	0.14	4.57	5.34
	3	1047.24	1636.02	0.11	0.13	5.48	6.71
121	1	182.85	243.35	0.15	0.17	6.43	7.06
	2	343.40	389.78	0.13	0.14	7.69	8.42
	3	6.63	9.22	0.13	0.15	4.97	6.51
138	1	11.95	19.89	0.16	0.17	9.57	17.04
	2	17.63	29.26	0.15	0.17	7.32	10.88
	3	11.99	14.26	0.16	0.17	8.73	10.55

Table 6.6 The Deviations of the Heuristics _ Small Sized Instances -- DueDate Set 1 for 5 instances (5 *D* values)

n	Instance No	Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum
32	1	8.84	10.45	0.04(4)*	0.18
	2	13.24	20.24	0.70(1)	2.96
	3	16.44	17.29	0.24(4)	1.21
36	1	10.48	16.01	0.04(4)	0.18
	2	17.37	20.93	1.70	2.69
	3	8.27	13.07	2.65	5.17
40	1	11.99	13.79	0.03(4)	0.17
	2	5.90	6.58	0.00(5)	0.00
	3	9.74	11.22	0.09(3)	0.29

*The numbers in parentheses give the number of times the heuristic finds the optimal solution (out of 5)

Table 6.7 The Deviations of the Heuristics _ Large Sized Instances -- DueDate Set 1 for 5 instances (5 *D* values)

n	Instance No	Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum
87	1	19.95	34.21	0.00(5)*	0.00
	2	20.94	25.28	0.19(3)	0.48
	3	11.39	18.14	0.46(3)	1.44
104	1	24.16	25.07	0.18(3)	0.44
	2	8.71	11.45	0.83(1)	1.26
	3	13.08	15.54	0.70(2)	3.24
121	1	9.01	12.13	1.06(1)	2.60
	2	13.38	13.77	0.47	0.82
	3	18.56	21.24	0.54(1)	1.04
138	1	25.77	30.50	3.60	6.46
	2	16.43	25.00	0.40(3)	1.43
	3	41.84	46.90	7.02	10.22

*The numbers in parentheses give the number of times the heuristic finds the optimal solution (out of 5)

Table 6.8 The CPU Times _ Small Sized Instances -- DueDate Set 2 for 5 instances (5 *D* values)

n	Instance No	MILP		Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum	Average	Maximum
32	1	42.65	52.52	0.02	0.02	0.50	0.62
	2	39.58	56.25	0.02	0.02	0.47	0.51
	3	30.24	35.29	0.02	0.03	0.49	0.55
36	1	53.42	61.23	0.03	0.03	0.55	0.61
	2	9.40	9.82	0.03	0.03	0.69	0.82
	3	6.68	7.30	0.03	0.03	0.67	0.70
40	1	312.42	320.19	0.03	0.03	0.67	0.71
	2	667.83	721.30	0.03	0.03	0.71	0.76
	3	445.92	487.90	0.03	0.03	0.53	0.55

Table 6.9 The CPU Times _ Large Sized Instances -- DueDate Set 2

n	Instance No	MILP		Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum	Average	Maximum
87	1	139.27	153.05	0.08	0.09	3.23	3.67
	2	195.31	201.02	0.08	0.10	3.75	4.05
	3	340.64	365.58	0.08	0.09	3.89	4.52
104	1	366.20	611.56	0.09	0.10	6.38	7.75
	2	1655.43	1969.11	0.10	0.10	6.94	7.78
	3	1162.49	1386.82	0.10	0.11	7.02	7.31
121	1	1137.37	1548.20	0.12	0.12	9.99	10.83
	2	337.82	353.06	0.12	0.13	9.80	11.14
	3	61.93	81.57	0.12	0.14	11.41	11.97
138	1	327.23	456.06	0.13	0.14	16.73	18.47
	2	214.77	238.36	0.14	0.14	14.23	14.96
	3	484.66	515.88	0.14	0.16	14.96	15.26

Table 6.10 The Deviations of the Heuristics _ Small Sized Instances -- DueDate Set 2 for 5 instances (5 *D* values)

n	Instance No	Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum
32	1	4.44	6.00	1.10	2.03
	2	6.74	11.18	2.56	4.05
	3	8.13	9.68	1.05	2.29
36	1	3.34	4.32	1.20	2.59
	2	10.32	12.97	0.96	1.38
	3	9.94	18.25	1.52	2.67
40	1	4.17	9.15	1.28(1)*	2.47
	2	3.74	5.13	0.31	0.48
	3	5.13	8.32	0.33(1)	0.73

*The numbers in parentheses give the number of times the heuristic finds the optimal solution (out of 5)

Table 6.11 The Deviations of the Heuristics _ Large Sized Instances – DueDate Set 2 for 5 instances (5 *D* values)

n	Instance No	Construction Phase		Improvement Phase	
		Average	Maximum	Average	Maximum
87	1	8.88	12.11	6.11	9.15
	2	7.10	8.97	1.71	2.97
	3	7.13	8.95	2.16	4.13
104	1	13.23	17.11	1.68	2.99
	2	7.75	8.54	3.73	5.06
	3	7.79	8.46	3.24	3.83
121	1	10.86	12.66	6.51	6.71
	2	5.46	8.12	0.68	1.08
	3	14.68	15.62	5.02	7.02
138	1	16.65	19.11	3.42	7.20
	2	6.89	7.80	0.59	1.32
	3	13.29	14.38	2.25	3.33

The comparison of the MILP columns of Table 6.4 and Table 6.8, which show the CPU times for small-sized problems for DueDate Sets 1 and 2, respectively, can lead us that the CPU times tend to increase as the due dates are tightened under the assumption that the other parameters remain unchanged. We note that n=40 is an exception since the *D* values for DueDate Set 2 is not representative.

Additionally, tight due dates and nontrivial network types have the potential to dramatically increase the CPU times of the MILP model since reaching the optimal solution requires more mode combinations to explore. Therefore, observing the inconsistent CPU times over different instances is an expected result. Table 6.5 shows that the MILP solved the 3rd instance with 104 tasks for one of the *D* values in 1636.02 CPU time and the 3rd instance with 121 tasks for one of the *D* values in 9.22 CPU time. These two instances are an example of the inconsistency of the MILP model in terms of CPU times.

Besides, even though D values are not entirely representative for some instances, especially for large-sized instances, Table 6.5 and Table 6.9 show that tight due dates may affect the CPU times.

Our first expectation was that the heuristic solution would be less dependent on the number of tasks of the instances compared to the MILP model. As we can observe from Tables 6.4 and 6.5 (DueDate Set 1) and Tables 6.8 and 6.9 (DueDate Set 2), the results are in line with our expectations. The construction and improvement phases of our heuristic solutions show a slight increase in the CPU times while the number of tasks of the instances increases. Additionally, it is safe to mention that the slight increase in the CPU times while the number of tasks increases is linear.

One of the main advantages of our heuristic is the ability to solve large instances in short CPU times. Table 6.5 indicates that the maximum CPU time required to find a solution for the instances with n=138 is 17.04 seconds. Our observation does not change when we tighten the due dates. The consistent CPU times for finding solutions are preferable compared to the inconsistent CPU times of the MILP.

Moreover, the CPU time spent to find a solution remains consistent when *D* values vary. Tables 6.4 and 6.5, and Tables 6.8 and 6.9 show that the difference between the average and maximum CPU times are very close. Also, the average CPU times slightly change while the number of tasks increases.

As we mentioned earlier, our LPR-Based Heuristic consists of two phases, which are construction and improvement phases. First, we construct a feasible solution using the LPR, and then we try to improve the constructed solution by using the proposed method. We analyzed the deviations for both loose and tight due date sets, i.e., DueDate Set 1 and DueDate Set 2, respectively. Tables 6.6 and 6.7 report the results for DueDate Set 1, and Tables 6.10 and 6.11 are for DueDate Set 2.

Tables 6.6 and 6.7 report the deviations of the heuristic solutions from the optimal solutions and the number of optimal solutions found out of 5 different *D* values for the DueDate Set 1. Although we cannot find any optimal solution by the construction

phase, the improvement phase helps a lot to reach optimal solutions. We find optimal solutions for many instances, and we find 5 optimal solutions out of 5 *D* values for some instances. Besides, we cannot find optimal solutions for several instances but the average deviations from the optimal solutions for these instances vary between 0% and 7.02%, closer to the lower part of the interval.

Tables 6.10 and 6.11 report the deviations of the heuristic solutions from the optimal solutions and the number of optimal solutions found out of 5 different *D* values for DueDate Set 2. Note that tightening the due dates does not have a negative effect on the deviations. Even though the number of optimal solutions found decreases when we tighten the due dates, the deviations are still within an acceptable limit. The maximum deviation among instances for DueDate Set 2 is below 10 percent, and the average deviation is 2.26 among all instances for DueDate Set 2.

Table 6.12 below represents the CPU times for instances that we could not solve within two hours limit. We can observe that Two-Step Heuristic CPU times are consistent and preferable compared to the MILP.

Table 6.12 The CPU Times for Instances Unsolved within Time Limit

		Construction Phase		***Improvement Phase***	
n	***DueDate Set***	***Average***	***Maximum***	***Average***	***Maximum***
104	DueDate Set 1	0.0948	0.1032	3.91928	4.2845
138	DueDate Set 1	0.13546	0.1542	8.6189	11.1361
40	DueDate Set 2	0.03002	0.0305	0.70736	0.7395
104	DueDate Set 2	0.09562	0.1152	8.7043	8.7747
138	DueDate Set 2	0.13368	0.1409	17.71974	21.0114

Table 6.13 below summarizes the Coefficient of Network Complexities (CNC) of the instances used. We expect that the increase in CNC may lead increase in the CPU times for the MILP. On the other hand, we observe that increase in CNC does not have a significant effect on the Two-Step Heuristic CPU times.

Table 6.13 CNC Values of the Networks

Networks	***Instance***	***n***	***CNC***
Real-Life Projects	HSMonitor	135	1.31
	STAMINA	155	1.51
Small-Sized Networks	All instances	32	2
	All instances	36	2
	All instances	40	2
Large-Sized Networks	All instances	87	5
	All instances	104	6
	All instances	121	7
	All instances	138	8

We can conclude that the heuristic procedure produces excellent results in very small CPU times for all instances of the problem set. Hence it can be used as a powerful decision making tool by the managers of the Ministry of Health.

CHAPTER 7

CONCLUSIONS

In this study, we consider a discrete time-cost trade-off problem in project networks. We assume that the processing time requirement of a task can be reduced by additional cost. We aim to find the task times for the constrained optimization problem of minimizing the total cost while meeting the maximum total tardiness amount. We formulate the problem as a Mixed Integer Linear Programming model. Additionally, to find near-optimal solutions, we introduce a two-step heuristic algorithm. The heuristic algorithm benefits from the optimal solutions of the Linear Programming Relaxations of the mathematical model in the first step and modifies the solution by several pairwise exchange mechanisms, in the second step.

Our experiments show that the solution times of the mathematical model for the constrained optimization problem can be too high for many values of the total tardiness limits, in particular when there are many tasks. On the other hand, our proposed heuristic algorithm demonstrates a consistent and slight increase in the CPU times with the increases in the number of tasks. We also observe that the tightness of the due dates affects the quality of the solutions: the tighter the due dates, the higher is the deviations from the optimal solutions.

We use the optimal solutions of the constrained optimization problem to generate the exact set of the nondominated objective vectors with respect to the total tardiness and total cost criteria. We also present an Evolutionary Algorithm to find near-exact nondominated objective vectors. We illustrate the outputs of our algorithms in the two real-life projects from the Ministry of Health.

To the best of our knowledge, we present the first study on the discrete time-cost trade-off problems with tardiness penalties. We hope that our work contributes to the literature of the discrete time-cost trade-off problems; brings light and motivation for

future research on the field. Some note-worthy further research directions are the development of implicit enumeration algorithms to find exact solutions in our constrained optimization problem. Such algorithms may use the optimal solutions of the Linear Programming Relaxations for defining lower bounds and for guiding the direction of the search space. Moreover, different constrained optimization problems like minimizing total tardiness while obeying the budget on the task time reductions, may be a fruitful research direction. In addition, a preference-based EA can be developed to generate objective vectors that are interest of the decision-maker, rather than the whole Pareto front. Also, future research may consider a dynamic environment where the task data change during the execution of the project. The procedures that quickly react to those changes would be invaluable tools for the practitioners.

REFERENCES

Akkan, C., Drexl, A., and Kimms, A., 2005. Network decomposition-based benchmark results for the discrete time-cost tradeoff problem. *European Journal of Operational Research*, 165 (2), 339–358.

Aminbakhsh, S. and Sonmez, R., 2016. Discrete particle swarm optimization method for the large-scale discrete time-cost trade-off problem. *Expert Systems with Applications*, 51, 177–185.

Chassiakos, A.P., Asce, A.M., and Sakellaropoulos, S.P., 2005. Time-Cost Optimization of Construction Projects with Generalized Activity Constraints. *Journel of Construction Engineering and Management*, 131 (10), 1115–1124.

Choi, B.C. and Chung, J., 2014. Complexity results for the linear time-cost tradeoff problem with multiple milestones and completely ordered jobs. *European Journal of Operational Research*, 236 (1), 61–68.

Choi, B.C. and Park, M.J., 2015. A continuous time-cost tradeoff problem with multiple milestones and completely ordered jobs. *European Journal of Operational Research*, 244 (3), 748–752.

Crowston, W. and Thompson, G.L., 1967. Decision CPM: A Method for Simultaneous Planning, Scheduling, and Control of Projects. *Operations Research*, 15, 407–426.

Crowston, W.B., 1970. *Decision CPM: Network Reduction and Solution*. Cambridge, No. 457.

De, P., Dunne, E.J., Ghosh, J.B., and Wells, C.E., 1995. The discrete time-cost tradeoff problem revisited. *European Journal of Operational Research*, 81, 225–238.

De, P., Dunne, J., Ghosh, J.B., and Wells, C.E., 1997. Complexity of The Discrete Time-Cost Trade-off Problem for Project Networks. *Operations Research*, 45 (2), 302–306.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 6 (2).

Değirmenci, G., 2008. The Budget Constrained Discrete Time/Cost Trade-off Problem in Project Networks. Middle East Technical University.

Değirmenci, G. and Azizoğlu, M., 2013. Branch and bound based solution algorithms for the budget constrained discrete time/cost trade-off problem. *Journal of the Operational Research Society*, 64 (10), 1474–1484.

Demeulemeester, E., Herroelen, W.S., and Elmaghraby, S.E., 1996. Optimal procedures for the discrete time/cost trade-off problem in project networks. *European Journal of Operational Research*, 88, 50–68.

Demeulemeester, E., Reyck, D.B., Foubert, B., Herroelen, W., and Vanhoucke, M., 1998. New computational results on the discrete time/cost trade-off problem in project networks. *Journal of the Operational Research Society*, 49 (11), 1153–1163.

Eynde, R. van and Vanhoucke, M., 2022. Journal Pre-proof A reduction tree approach for the Discrete Time/Cost Trade-Off Problem A reduction tree approach for the Discrete Time/Cost Trade-Off Problem. *Computers and Operations Research*.

Feng, C.-W., Liu, L., Member, A.A., Burns, S.A., and Member, A., 1997. Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems. *Journal of Computing in Civil Engineering*, 11 (3), 184–189.

Fulkerson, D.R., 1961. A Network Flow Computation for Project Cost Curves. *Operations Research*, 7 (2), 167–178.

Get prepared - H2020 Online Manual [online], 2022. Available from: https://ec.europa.eu/research/participants/docs/h2020-funding-guide/grants/applying-for-funding/submit-proposals/get-prepared_en.htm [Accessed 30 Mar 2022].

Grigoriev, A. and Woeginger, G.J., 2004. Project scheduling with irregular costs: Complexity, approximability, and algorithms. *Acta Informatica*, 41 (2–3), 83–97.

Hafızoğlu, A.B. and Azizoğlu, M., 2010. Linear programming based approaches for the discrete time/cost trade-off problem in project networks. *Journal of the Operational Research Society*, 61 (4), 676–685.

Haimes, Y.Y., Lasdon, L.S., and Wismer, D.A., 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Journals & Magazines*, 1 (3), 296–297.

Hazir, Ö., Erel, E., and Günalay, Y., 2011. Robust optimization models for the discrete time/cost trade-off problem. *International Journal of Production Economics*, 130 (1), 87–95.

Hazir, Ö., Haouari, M., and Erel, E., 2010a. Discrete time/cost trade-off problem: A decomposition-based solution algorithm for the budget version. *Computers and Operations Research*, 37 (4), 649–655.

Hazir, Ö., Haouari, M., and Erel, E., 2010b. Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research*, 207 (2), 633–643.

He, Z. and Xu, Y., 2008. Multi-mode project payment scheduling problems with bonus-penalty structure. *European Journal of Operational Research*, 189 (3), 1191–1207.

Hindelang, T.J. and Muth, J.F., 1979. A Dynamic Programming Algorithm for Decision CPM Networks. *Source: Operations Research*, 27 (2), 225–241.

Horizon 2020 | European Commission [online], 2022. Available from: https://ec.europa.eu/info/research-and-innovation/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-2020_en [Accessed 30 Mar 2022].

Horizon Europe | European Commission [online], 2022. Available from: https://ec.europa.eu/info/research-and-innovation/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-europe_en [Accessed 30 Mar 2022].

HSMonitor - Health Status Monitor - ICT-enabled monitoring [online], 2022. Available from: https://hsmonitor-pcp.eu/ [Accessed 30 Mar 2022].

Li, H., Xu, Z., and Wei, W., 2018. Bi-Objective Scheduling Optimization for Discrete Time/Cost Trade-Off in Projects. *Sustainability (Switzerland)*, 10 (8).

Li, X., He, Z., Wang, N., and Vanhoucke, M., 2020. Multimode time-cost-robustness trade-off project scheduling problem under uncertainty. *Journal of Combinatorial Optimization*, 1–30.

Lim, S.M., Sultan, A.B.M., Sulaiman, M.N., Mustapha, A., and Leong, K.Y., 2017. Crossover and mutation operators of genetic algorithms. *International Journal of Machine Learning and Computing*, 7 (1), 9–12.

Ministry of Health, 2022. Duties and Powers [online]. Available from: https://www.saglik.gov.tr/EN,15621/duties-and-powers.html [Accessed 30 Mar 2022].

OECD, 2022. Health spending (Indicator) - OECD Data [online]. Available from: https://data.oecd.org/healthres/health-spending.htm [Accessed 30 Mar 2022].

Pinho, R. and Saraiva, F., 2020. A Comparison of Crossover Operators in Genetic Algorithms for Switch Allocation Problem in Power Distribution

Systems. *In*: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 1–8.

Project Management Institute Inc., 2021a. *The standard for project management and a guide to the project management body of knowledge (PMBOK guide)*.

Project Management Institute Inc., 2021b. *Pulse of the Profession ® 2021 Flex to the Future*.

Project Management Institute Inc., 2021c. *Talent Gap: Ten-Year Employment Trends, Costs, and Global Implications*.

Robinsonf, D.R., 1975. A Dynamic Programming Solution to Cost-Time Tradeoff for CPM. *Management Science*, 22 (2), 158–166.

Sağlık Bilgi Sistemleri Genel Müdürlüğü, 2020. Genel Müdürlüğümüzün Görevleri [online]. Available from: https://sbsgm.saglik.gov.tr/TR-1275/genel-mudurlugumuzun-gorevleri.html [Accessed 30 Mar 2022].

Said, S.S. and Haouari, M., 2015. A hybrid simulation-optimization approach for the robust Discrete Time/Cost Trade-off Problem. *Applied Mathematics and Computation*, 259, 628–636.

Skutella, M., 1998. Approximation algorithms for the Discrete Time-Cost Tradeoff Problem. *Mathematics of Operations Research*, 23 (4), 909–929.

Szmerekovsky, J.G. and Venkateshan, P., 2012. An integer programming formulation for the project scheduling problem with irregular timecost tradeoffs. *Computers and Operations Research*, 39 (7), 1402–1410.

The STAMINA Project - Stamina [online], 2022. Available from: https://stamina-project.eu/about/ [Accessed 30 Mar 2022].

Vanhoucke, M. and Debels, D., 2005. *The Discrete Time/Cost Trade-Off Problem Under Various Assumptions Exact And Heuristic Procedures*.

Vanhoucke, M., Demeulemeester, E., and Herroelen, W., 2002. Discrete time/cost trade-offs in project scheduling with time-switch constraints. *Journal of the Operational Research Society*, 53 (7), 741–751.