MULTI-PROJECT SCHEDULING
UNDER MODE DURATION UNCERTAINTIES

by

E. ARDA ŞiŞBOT

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2011

MULTI-PROJECT SCHEDULING
UNDER MODE DURATION UNCERTAINTIES

APPROVED BY

Prof. Gündüz Ulusoy
(Thesis Co-Supervisor) .....

Assoc. Prof. Can Akkan
(Thesis Co-Supervisor) .....

Prof. Ümit Bilge .....

Assist. Prof. Kerem Bülbül .....

Assist. Prof. Güvenç Şahin .....

DATE OF APPROVAL: 21/7/2011

to my family

Acknowledgments

First and foremost, I want to thank my thesis advisors Prof. Gündüz Ulusoy and Assoc. Prof. Can Akkan for their guidance throughout this thesis. Their expertise, patience and good humor turned my research experience into a pleasure.

I would like to thank my colleagues in the project: Berke Pamay, Gizem Kılıçaslan and Anıl Can for their direct/indirect help to my research. Anıl Can deserves a special mention for all his support at the beginning of this thesis.

I thank Mustafa for being there to help me whenever I needed. Many thanks to Mahir, for making me come to Lab 1021. Among others Gizem Ç., Volkan, Çetin, Semih, Birce, Özge, Yasir, Nükte, Ezgi thank you for all your support and contributions.

Last but not least I'd like to thank my family for their endless support. I am indebted to my brother, Akın for helping me on every occasion and always being such a good model to follow.

To them I dedicate this thesis.

MULTI-PROJECT SCHEDULING UNDER MODE DURATION UNCERTAINTIES

E. Arda Şişbot

Industrial Engineering, Master of Science Thesis, 2011

Thesis Co-Supervisors: Prof. Gündüz Ulusoy, Assoc. Prof. Can Akkan

Keywords: multi-project scheduling, multi-objective genetic algorithm, robust project scheduling

Abstract

In this study, we investigate the multi-mode multi-project resource constrained project scheduling problem under uncertainty. We assume a multi-objective setting with 2 objectives : minimizing multi-project makespan and minimizing total sum of absolute deviations of scheduled starting times of activities from their earliest starting times found through simulation. We develop two multi-objective genetic algorithm (MOGA) solution approaches. The first one, called decomposition MOGA, decomposes the problem into two-stages and the other one, called holistic MOGA, combines all activities of each project into one big network and does not require that activities of a project are scheduled consecutively as a benchmark.

Decomposition MOGA starts with an initial step of a 2-stage decomposition where each project is reduced to a single macro-activity by systematicaly using artificial budget values and expected project durations. Generated macro-activities may have one or more processing modes called macro-modes. Deterministic macro-modes are transformed into random variables by generating disruption cases via simulation. For fitness computation of each MOGA two similar 2-stage heuristics are developed. In both heuristics, a minimum target makespan of overall projects is determined. In the second stage minimum total sum of absolute deviations model is solved in order to find solution robust starting times of activities for each project. The objective value of this model is taken as the second objective of the MOGA's.

Computational studies measuring performance of the two proposed solution approaches are performed for different datasets in different parameter settings. When non-dominated solutions of each approach are combined to a final population, overall results show that a larger ratio of these solutions are genetared by decomposition MOGA. Additionally, required computational effort for decompositon MOGA is much less than holistic approach as expected.

REÇETE SÜRESİ BELİRSİZLİĞİ ALTINDA ÇOKLU PROJE ÇİZELGELEME

E. Arda Şişbot

Endüstri Mühendisliği, Yüksek Lisans Tezi, 2010

Tez Danışmanları: Prof. Gündüz Ulusoy, Doç. Dr. Can Akkan

Anahtar Kelimeler: çoklu proje çizelgeleme, çok amaçlı genetik algoritma, gürbüz çizelgeleme

Özet

Bu çalışmada belirsizlik altında çoklu kaynak reçeteli, kaynak kısıtlı çoklu proje çizelgeleme sorunu incelenmektedir. Sorunun iki amaç işlevinin bulunduğu var sayılmaktadır: bir olasılık limiti dahilinde aşılmaması sağlanan en düşük çoklu-proje süresinin elde edilmesi ve belirlenecek faaliyet başlangıç zamanlarının benzetim ile elde edilen en erken başlangıç sürelerinden toplam mutlak sapmayı en azlayacak biçimde belirlenmesi. İki ayrı çok amaçlı genetik algoritma (ÇAGA) geliştirilmiştir. Ayrışımlı ÇAGA olarak adlandrılan ilk yaklaşım sorunu iki aşamaya ayırmakta, bütünsel ÇAGA ad verilen ise tüm projelerin faaliyetlerini tek bir birleşik ağ olarak ele alıp, bütünsel bir yaklaşım sergilemektedir.

Ayrışımlı ÇAGA yaklaşımında öncelikle iki-aşamalı bir ayrışım uygulanmaktadır. Her proje, farklı yapay bütçe değerlerinin sistematik bir biçimde kullanılmasıyla oluşturulan bir veya daha çok sayıda kaynak reçetesine sahip tek bir makro faaliyete indirgenir. Türetilen makro-faaliyetlerin, makro-kaynak reçetesi adı verilen bir ya da birden fazla kaynak reçetesi olabilir. Makro-faaliyetlerin her biri için rassal olarak türetilen kaynak reçetesi süreleri ile faaliyetlerin belirsizliği modellenmiştir. Her iki ÇAGA'da da amaç işlevlerinin hesaplanmasında alt yöntemleri benzer iki-aşamalı sezgiseller kullanılmaktadır. Çaprazlama ve kromozom temsilleri farklılık göstermektedir. Her iki sezgiselde de ilk aşamada öncelikle düşük bir çoklu-proje süresi elde edilir. İkinci aşamada toplam mutlak sapma modeli en azlanmaktadır. Bu modelin amaç değeri ÇAGA'larn ikinci amaç değerine karşılık gelmektedir.

Bilişimsel çalışmalar, iki ÇAGA için de farklı veri setleri ve ağ parametreleri için yapılmıştır. Her iki yaklaşımın çözümleri birleştirilip domine edilmeyen sınır bulunduğunda, sonuçların büyük bir ölçüde ayrışımlı ÇAGA'dan geldiği ortaya çıkmaktadır. Ayrıca çözümler, ayrışımlı ÇAGA için gereken çözüm süresinin bütünsel ÇAGA'ya göre çok daha az olduğunu göstermektedir.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction and Motivation

The world's ancient architectural masterpieces are often cited as the earliest examples of projects. Egyptian pyramids or Temple of Artemis are perfect examples of projects that are managed throughout the centuries requiring vast amount of resources and manpower, holding extreme importance in the eyes of their executors. Along with many practices of good project management as in the case of Hagia Sophia constructed in 5 years, ancient history is full of cancelled, postponed or tardy projects due to resource inadequacies, unanticipated events or poor management. In today's world, significant projects are widespread: from CERN's hadron collider to an Airbus plane design the importance and complexities of projects are increasing. Correspondingly, management requirements to develop better tools for better project management increases as well.

Basic project scheduling deals with scheduling the activities (tasks) to fullfill a desired objective. Generally project related costs and project duration (makespan) are observed as the most common objective functions. Dating back to fifties, PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method) are widely applied techniques for this problem without any resource constraints. When the resources are shared between activities, the problem is classified under the title Resource Constrained Project Scheduling Problem (RCPSP). As the problem comes from a very real setting, various extensions have been studied in the literature. Operating on the same basis as RCPSP, RCMPSP is an extension of RCPSP to multi-project setting.

Today's competitive environment urges companies to manage more than one project at a time. Big companies allocate same pool of resources to multiple projects simultaneously. Simultaneously managed projects may use common resources with different requirements, may have different deadlines and priorities. Payne [2] suggests that up to 90%, by value, of all projects occur in a multiproject context.

The case with multi-mode availabilities, where each activity may have more than one processing alternatives (modes) yields a better modeling of reality. Often in real life, project managers have the choice of decreasing the duration of activities at the cost of additional resources. In a construction project, for example, a specific task can be accelerated by employing additional workers. The presence of activitiy modes, although realistic, complicates the project and scheduling.

Another aspect of multi-project management is that the performance of each project constitutes an essential part of the multi-project management. With or without precedence relations imposed between projects, projects are inter-related by resource sharing. For that reason, an unanticipated event occurring in a project may effect others and consequently may have a major influence on the multi-project management. Hence, dealing with uncertainty and avoiding unplanned disruptions is extremely important in multi-project settings.

In project scheduling, uncertainty can take many different forms. Activity duration estimates may be off, resources may break down, work may be interrupted due to extreme weather conditions, new unanticipated activities may be identified, etc. All these types of uncertainties may result in a disrupted schedule which leads to higher costs and penalties, undesired resource idleness and poor project performance levels. In general, project management wants to avoid these schedule breakages. Thus the need to protect a schedule from the adverse effects of possible disruptions emerges. This protection is necessary because often project activities are subcontracted or executed by resources that are not exclusively reserved for the current project. A change in the starting times of such activities could lead to infeasibilities at the organizational level (e.g., in a multi-project context) or penalties in the form of higher subcontracting costs or material acquisition and inventory costs.

This study focuses on developing solution approaches to multi-mode RCMPSP under mode duration uncertainties. We assume that uncertainty may only arise as a result of different realized values of the activity modes. Durations of the modes are subject to change within predefined lower and upper bounds. We consider two of the most common objectives in robust project scheduling: solution and quality robustness. Solution robustness refers to the stability of the activity starting times and quality robustness refers to stability of the makespan over all projects.

The first solution approach, inspiring from macro-mode decomposition by Speranza and Vercellis [3], is a decomposition based multi-objective genetic algoritm (D-MOGA). Macro-modes that are systemic transformations of project network by evaluating durations and artificial resource budgets are firstly generated. Then via simulations of composing activity mode durations, each macro-mode is transformed into combinations of random variables. D-MOGA searches for different project sequences and macro-mode assignments. A two-stage heuristic is employed for the evaluation of each solution. In the first stage, the heuristic serially schedules projects considering probability of assuring resource feasibility. Then, buffers are inserted to obtain solution robust starting times for the projects. In the second stage, each project is scheduled individually with a solution robustness objective. Thus, both a multi-project schedule and individual single-project schedules are obtained along with objective pairs (solution robustness objective and makespan). MOGA then finds non-dominated solutions throughout the generations.

Second solution approach applies similar ideas of the described heuristic to whole network without decomposition. Another MOGA, called holistic MOGA (H-MOGA) is developed which progresses on all activitities of all projects and their selected modes. Having a longer chromosome length, this approach requires more computational power as the results on section 6.5 suggest.

1.1 Contributions

The primary purpose of the present study is developing solution procedures to multi-mode RCMPSP with mode duration uncertainties. The following list shows the contributions of this study:

- To the best of our knowledge, there is no study dealing with multi-mode RCMPSP under uncertainy. It can be said that even the studies on single project RCPSP with multi-mode duration uncertainties are rather scarce. [4], [5], [6], [7]
- As a solution procedure we proposed 2 heuristic approaches one taking its roots from 2-stage decomposition and the other approaching the problem in a holistic fashion.
- Macro-mode decomposition used solely on deterministic settings is applied to this stochastic problem. Deterministic macro-modes are transformed into random variables.
- This is the first study in robust scheduling that adopt a multi-objective setting rather than a composite measure of multiple objectives or a single measure of robustness.

1.2 Outline

Chapter 2 reviews briefly the literature. Chapter 3 presents the problem environment and the notation used. The solution procedure, a decomposition based MOGA is described in Chapter 4. Another solution approach is given in detail in Chapter 5. Afterwards we present computational studies in Chapter 6. Finally we close by concluding thoughts and future research directions in Chapter 7.

CHAPTER 2

Literature Review

In its simplest form, RCPSP is defined on a deterministic single project network with known activity durations and resource requirements. This problem intends to determine an optimal schedule which satisfies generalized precedence relations and resource constraints and with an objective function generally defined as the makespan or some financial function. In the past decade as Brucker et al. observed in 1998 [8], the literature on RCPSP has extended fast such that major research tracks on variants and extensions of RCPSP are now discussed. Major extensions of the problem include multi-mode RCPSP (MRCPSP), RCMPSP and project scheduling under uncertainty. In MRCPSP the activities have more than one mode and one wishes to determine the optimal assignment of modes for the desired objective. RCMPSP aims to extend the research to multiple project case, which makes the problem harder to solve.

Project scheduling under uncertainty has been attracting the attention of researchers particularly in the last decade. The schedule determined by deterministic RCPSP is called the baseline schedule. Activity durations may not be constant, thus may take more or less time than estimated. The arriving times of resources may incur delays; priorities or due dates of activities may change. Resources may break down, work may be interrupted due to extreme wheather conditions or new unanticipated activities may appear. All these types of uncertainties may result in the infeasibility of the baseline schedule or a disrupted schedule with inferior performance levels. Thus the need to protect the initial baseline schedule from the adverse effects of possible disruptions emerges. This can be achieved by generating a

baseline schedule in a proactive way trying to anticipate certain types of disruptions so as to minimize their effect, if they occur. If the schedule would still break down despite these proactive planning efforts, a reactive scheduling policy will be needed to repair the infeasible schedule.

Correspondingly, the validity of deterministic project scheduling has been questioned and new tracks of research have emerged in the literature. According to Herroelen and Leus [9], research on project scheduling under uncertainty has been focusing on 4 major research tracks: proactive scheduling, reactive scheduling, stochastic project scheduling and fuzzy project scheduling. Proactive (robust) scheduling corresponds to determining a robust schedule facing the least disruptions during project execution. Reactive scheduling includes attempts to restore and update the schedule whenever an unexpected event occurs. Stochastic project scheduling literature includes application of stochastic optimization procedures. Finally fuzzy project scheduling uses fuzzy activity durations and produces fuzzy schedules. Our study corresponds to proactive and stochastic project scheduling literature, thereby we present here selected works from the related literatures.

Based on the work of Tavares et al. [10], Leus [11] and Herroelen and Leus [12]; Van de Vonder et al. [13] investigate the tradeoff between stability and makespan of a schedule. The authors describe a heuristic procedure for generating buffered baseline schedules for projects with ample renewable resource availability. After generating a schedule via exact optimization methods (see, e.g. Demeulemeester and Herroelen [14]), starting times of each activity are modified according to the so called activity dependent float factors, functions of the weights of the predecessors and successors of an activity. This modification of starting times guarantees prececedence feasibility, however, it may yield resource infeasible schedules. To answer this need, Van de Vonder et al. [13] propose resource flow-dependent float factor heuristic (RFDFF), which considers resource flows in calculation of float factors. In RFDFF heuristic, a new project network is created, where the resource flows among activities in the baseline schedule are implemented as additional precedence relationships. Starting times of the activities are modified with respect to new float factors taking

into account the new predecessors/successors from resource flow, therefore yield a precedence and resource feasible schedule.

Van de Vonder et al. [15] have performed simulation-based experiments in order to measure the performance of various buffering heuristics. In addition to RFDFF heuristic, virtual activity duration heuristic use standard deviations of activities to estimate possible disruptions and starting time criticality (STC) heuristic which exploits information about the variance of the activity durations are among many heuristics evaluated. STC outperforms others by incorporating the uncertainty in a probabilistic way and making use of the variability in every stage.

Chtourou et al. [16] propose a two-stage priority-rule-based algorithm considering both quality and solution robustness. After forming an activity list by a priority rule, an earliest start schedule is generated. To increase variability of the schedule, random partial destruction and reconstruction techniques are employed along with generation of a backward schedule. Schedule (forward or backward) resulting in smaller makespan is selected as the input of the second stage problem. In the second stage, taking the previously found makespan as threshold same heuristic is re-run to obtain a schedule with better robustness value and smaller makespan. To measure robustness the authors make use of different measures such as sum of free slack and average percentage increase in activity duration.

Another two-phase algoritm is developed by Hazir et al. [4] but in multi-mode setting with the objective of total budget minimization and robustness maximization. In order to select the most representative robustness metric they perform experiments on measures such as average slack, weighted slack, slack utility function, dispersion of slacks, percentage of potentially critical activities and project buffer. The robustness measure that the has the highest correlation with a performance measure is selected as the best metric to represent robustness. The authors provide empirical evidence that the project buffer size is the more appropriate robustness measure regardless of the network complexity. Based on this finding, they develop a two-phase approach for generating robust schedules, where in the first phase the minimum required budget is determined and in the second stage this bud-

get is slightly inflated by a specified amplification factor and then the buffer size is maximized.

Bruni et al. [17] address project scheduling problem with random activity durations. As a solution procedure, they propose a heuristic which uses joint probabilistic constraints. For scheduling activities firstly a priority rule is employed to decide which new activity to assign at a time point. If with the new activity, the schedule's probability of not exceeding the projected makespan is within limits, then the activity is selected. If an activity with higher probability does not satisfy that probabilistic constraint, then the algorithm passes to next activity. Thus, the proposed heuristic limits the schedule's probability of exceeding projected makespan. The authors conclude that their approach demonstrates the effectiveness of rigorous treatment of uncertainty leading to better uncertainty hedging.

For objective values differing between expected makespan and expected expenses Golenko-Ginzburg and Gonik [18] consider random activity durations and propose a heuristic in which each activity is prioritized by the product of its probability (determined by simulation) of lying on the critical path and its average duration. Golenko-Ginzburg and Gonik [19] consider in additon two types of renewable resources: rare and not-rare. Golenko-Ginzburg et al. [20] extend the previous research by incorporating uncertainties regarding activity resource usages. The authors approach combines simulation, a cyclic coordinate descent method and a knapsack resource reallocation model. Golenko-Ginzburg and Gonik [21] enlarge the problem of Golenko-Ginzburg and Gonik [18] into multi-project case.

Zhu et al. [22] propose a two- stage stochastic programming model for minimizing the expected deviations and total cost. First stage consists of the problem of setting target finish times for each activity with respect to cost associated with the target times. Second stage finds optimal starting times in order to minimize the expected cost of deviating from the original plan. They show that in the absence of a budget constraint, the second stage problem can be transformed into a minimum cost network problem and therefore can be efficiently embedded in a stochastic programming algorithm. They use the L-shaped method to solve LP relaxation of

stochastic program for the case without a budget constaint.

Klerides et al. [5] propose a decomposition-based stochastic programming approach for the project scheduling problem under time-cost trade-off settings and uncertain durations. Assuming static assignment of activity modes, they show that the stochastic extension of the discrete time-cost trade-off problem (SDTCTP) can be formulated as a two-stage stochastic integer program with recourse. The execution modes for the activities are determined in the first stage in a context where the exact duration of each activity for that particular mode is not known in advance. Given these first stage decisions, the values of the activity starting time variables (second stage or recourse) are determined based on the realizations of the activity durations. Their approach combines a path based formulation of the deterministic discrete time−cost trade−off problem, and a delayed constraint generation procedure, which allows for the decoupling of the different scenario subproblems via decomposition. The proposed solution methodology contains effective constraint selection criteria at each iteration and many large and hard test instances can be solved in reasonable computational time.

Zhu et al. [6] study reactive procedures for RCPSP with finish to start precedence constraints. They propose a classification scheme for the different types of disruptions. By forming an integer linear model and solving it with hybrid mixed-inter programming/constraint programming procedures, authors show that by defining appropriate recovery time windows and penalty functions optimal solutions to the recovery problem are well within reach.

Deblaere et al. [7] formulates a reactive scheduling problem for MRCPSP. Given a baseline schedule and a resource or activity duration disruption that occurs during the execution of the baseline schedule, their objective is to obtain a reactive schedule that minimizes the rescheduling costs. If thougroughout the schedule an activity switch its mode from the previous schedule, then mode switching costs are incurred. In addition, rescheduling costs include the deviation in starting times of each activity from the baseline schedule. They propose a branching scheme based on mode and delaying alternatives for optimally solving the reactive scheduling prob-

lem. Given the high complexity due to structure of problem the authors to explore other strategies than regular branch-and-bound namely: iterative deepening, binary search and branch-and-bound with tabu seach. Their computational studies are in favor of using branch-and-bound with tabu seach where they propose the use of a tabu search procedure to obtain a heuristic solution for the reactive scheduing problem, and to use the objective value of this heuristic solution as an upper bound to be used in the regular branch-and- bound procedure.

CHAPTER 3

Problem Environment

The examined problem environment contains multiple projects consisting of activities which have multiple mode alternatives. Each mode alternative has a duration that is a triangular distributed random variable with pregiven lower and upper bounds. It is assumed that activities cannot be preempted.

3.1 Resources

We consider two types of resource constraints: renewable and non-renewable. Renewable resources are constrained on a periodic basis and are assumed to be available throughout the project. Examples for a renewable resource would be workforce or available equipment. Nonrenewable resources on the other hand, are consumed and are limited over the entire planning horizon with no restrictions within each period. Supply of material or capital available are examples of nonrenewable resources.

3.2 Network structure

The project network is of activity-on-node (AoN) type with finish to start zero time lag type precedence relations. The composite multi-project network is generated by combining single project networks employing one dummy start node and one dummy finish node. Figure 3.1 illustrates an example multi-project network.

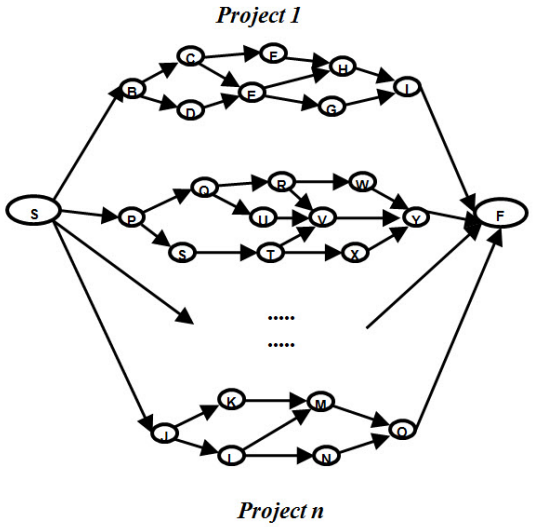

Figure 3.1: Composite multi-project network with N projects and dummy start-finish nodes

3.3 Problem formulation

A mathematical programming formulation is formed to represent this MRCMPSP under uncertainty. With a decision environment considering 2 objectives, the proposed formulation includes both makespan and total sum of absolute deviations ($TSAD$).

3.3.1 Sets and indices

K = set of all realizations of activities

S = set of all projects including dummy projects

S^a = set of all actual projects

s = project indices; $s \in S = \{1, 2, \ldots, |S|\}$

V = set of all activities including dummy activities

V_s = set of activities in project s including dummy activities

i,k = activity indices; $i, k \in V_s$

P = set of precedence relations between all activities $i \in V$

P_s = set of precedence relations between all activities $i \in V_s$ in project s

M_{si} = set of modes of activity i of project s

j = activity execution mode indices; $j \in M_i = \{1, 2, \ldots, |M_i|\}$

$\mathcal{R}$ = set of renewable resources

r = renewable resource indices; $r \in \mathcal{R} = \{1, 2, \ldots, |\mathcal{R}|\}$

$\mathcal{N}$ = set of non-renewable resources

n = non-renewable resource indices; $n \in \mathcal{N} = \{1, 2, \ldots, |\mathcal{N}|\}$

$\mathcal{T}$ = set of time periods

t,θ = time indices; $t \in \mathcal{T} = \{1, \ldots, |\mathcal{T}|\}$

3.3.2 Parameters

d_{sij} = processing time for activity i of project s performed in mode j (Random)

$\bar{d}_{sij}$ = expected processing time for activity i of project s performed in mode j

d_{sij}^{min} = minimum processing time for activity i of project s performed in mode j

d_{sij}^{max} = maximum processing time for activity i of project s performed in mode j

E_{si}^{k} = earliest starting time period for activity i of project s in realization k

e_{si} = earliest starting time period for activity i of project s

l_{si} = latest starting time period for activity i of project s

W_r = amount of available renewable resource r

Q_n = amount of available non-renewable resource n

w_{sijr} = amount of renewable resource r utilized by activity i of project s performed in mode j

q_{sijn} = amount of non-renewable resource n consumed by activity i of project s performed in mode j

T = total length of the time horizon

$TargetMakespan$ = overall multi-project duration

All parameters except e_{si}, l_{si} and T must be initially given to solve the problem. Due to the stochastic nature of d_{sij} values, e_{si}, l_{si} are stochastic as well. However, for a given schedule e_{si}, l_{si} can be computed by generating various schedules by K simulations and measuring various starting times for each activity $i \in V$. T, for

example, can be set by just summing up the maximum durations of the longest modes of each activity.

3.3.3 Decision variables

A binary variable x_{sijt} based on starting time period and mode selection of activities is introduced along with two other integer variables based on it. It was also possible to represent the precedence relation constraints without defining T_i and D_i but they are included for practical purposes.

$x_{sijt} = 1$ if activity i of project s starts at time period t in mode j; $= 0$ otherwise.
T_{si} = Actual starting time of activity i of project s; $e_s i \leq T_{si} \leq l_{si}$,
D_{si} = Actual duration of activity i of project s; $min_{j \in M_s i}\{d_{sij}^{min}\} \leq D_i \leq max_{j \in M_s i}\{d_{sij}^{max}\}$

3.3.4 Mathematical model

The mathematical model described here has two objectives: *(i)* minimization of TSAD of activities from their early start times and *(ii)* minimization of the makespan over all projects. Mimimization of TSAD objective (3.1) relates to solution robustness and aims to obtain a schedule in which an activity related disruption causes a delay to another activity's starting time the minimum way possible. Makespan minimization of overall projects (3.2) relates to quality robustness where assurance of a minimum makespan is desired with a probability constraint (3.9). Note that there is a tradeoff between these two objectives. A highly solution robust schedule may be obtained by inserting long time-buffers between activities thus result in a higher makespan. On the other hand, one may obtain a very compact schedule with a low makespan and a high level of quality robustness but this schedule in general will not be solution robust due to lack of time buffers between activities.

Constraint set (3.3) represents the start times and constraint set (3.4) the durations for the projects. Constraint set (3.5) ensures the precedence relationships between the activities. Constraint set (3.6) is the capacity constraint for the renewable resources and constraint set (3.7) is the capacity constraint for the non-renewable

resources. Constraint set (3.8) ensures that for each project a mode alternative is selected and it is started at some point. The zero-one variables x_{ijt} are expressed in constraint set (3.10). Note that d_{ij} are random variables and hence starting times are also random.

Model MPS **:**

Objective 1:

$$\min TSAD = \sum_{s\in S}\sum_{i\in V_s}\sum_{k\in K}|T_{si} - E^k_{si}| \tag{3.1}$$

Objective 2:

$$\min \quad TargetMakespan \tag{3.2}$$

s.t.

$$T_{si} = \sum_{j\in M_{si}}\sum_{t=e_{si}}^{l_{si}} t x_{sijt} \qquad i \in V_s, s \in S, \tag{3.3}$$

$$D_{si} = \sum_{j\in M_{si}} d_{sij}\sum_{t=e_{si}}^{l_{si}} x_{sijt} \qquad i \in V_s, s \in S, \tag{3.4}$$

$$T_{sk} - T_{si} \geq D_{si} \qquad (i,k) \in P_s, s \in S, \tag{3.5}$$

$$\sum_{s\in S}\sum_{i\in V_s}\sum_{j\in M_{si}}\sum_{\theta=\max(e_{si},t-d_{sij}+1)}^{\min(l_{si}+d_{sij}-1,t)} w_{sijr}x_{sij\theta} \leq W_r \quad r \in \mathcal{R}, t \in \mathcal{T}, s \in S, \tag{3.6}$$

$$\sum_{s\in S}\sum_{i\in V_s}\sum_{j\in M_{si}} q_{sijn}\sum_{t=si}^{l_{si}} x_{sijt} \leq Q_n \qquad n \in \mathcal{N}, \tag{3.7}$$

$$\sum_{j\in M_{si}}\sum_{t=e_{si}}^{l_{si}} x_{sijt} = 1 \qquad i \in V_s, s \in S, \tag{3.8}$$

$$Prob(\max_{i\in V_s} T_{si} < TargetMakespan) \geq Limit \qquad s \in S, \tag{3.9}$$

$$x_{sijt} \in \{0,1\}, d^{min}_{sij} \leq d_{sij} \leq d^{max}_{sij}, i \in V_s, \qquad j \in M_{si}, t \in \mathcal{T} \tag{3.10}$$

Note the randomness of d_{sij} causes T_{si} and D_{si} to be random variables and brings a stochastic nature to the MRCMPSP. The model presented above is a conceptual model that we are not going to operationalize.

CHAPTER 4

Decomposition Heuristic Approach

In this chapter a 2-stage decomposition approach incorporating stochastic duration information is presented. To get a grasp of the idea, first a general look is presented below and details of the subprocedures are given in the following subsections.

Speranza and Vercellis [3] distinguished between a tactical and operational level in project scheduling where in tactical level higher management sets due date of projects and performs resource allocation, whereas in operational level project managers determine the starting times and selected modes of the activities. Approaching the problem in 2 stages as in Speranza and Vercellis [3] approximates the NP-hard problem by simpler subproblems thus decreasing computational burden. In the proposed decomposition, projects are transformed into macro-activities. Hence the multi-project network becomes a single project network where the activities in this network are macro-activities each representing a project. Figure 4.1 [1] illustrates the described transformation.

Can [1] proposed a 2-stage decomposition approach for deterministic RCMPSP with multi-modes. He applied a shrinking model for macro-mode generation proposed by Speranza and Vercellis [3]. Afterwards, he solved the problem for NPV maximization at the higher level and makespan minimization at the lower level. The approach presented here inspires from Can's thesis [1] and its $2-$stage decomposition approach, however, the nature of the problem at hand is different than Can's. Mode duration uncertainties bring a stochastic dimension to MRCMPSP. When stochastic activity durations are included, the decomposition is even more beneficial due to high computational burden of solving stochastic models.

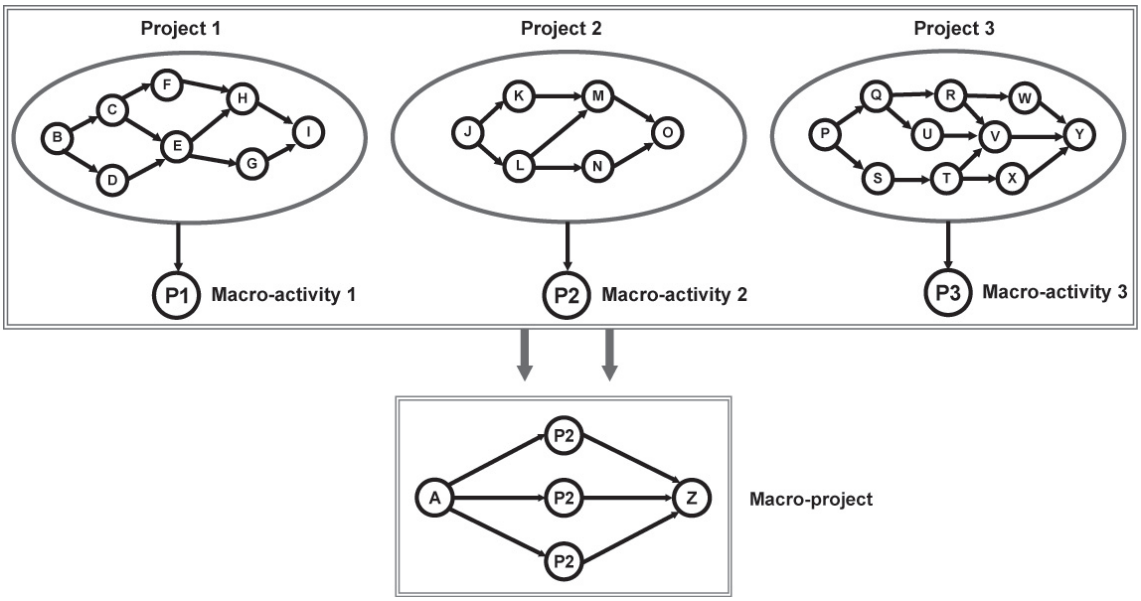

Figure 4.1: Macro-activities and macro-project [1]

Firstly macro-modes are generated with expected mode durations as in the deterministic case. Then with the uncertainty in mode durations macromode schedule disruptions are formed via simulation (section 4.1.3). To search feasible solutions of project sequence and macro-mode assignment a MOGA is introduced in Section 4.3. For each solution a target makespan, which the realized schedule guaranteed is not to exceed, is determined in Section 4.3.3.2. Also, activities are scheduled with a minimum deviation robustness objective and robust starting times are determined as in Section 4.3.3.3. Figure 4.2 presents a general flow of this approach.

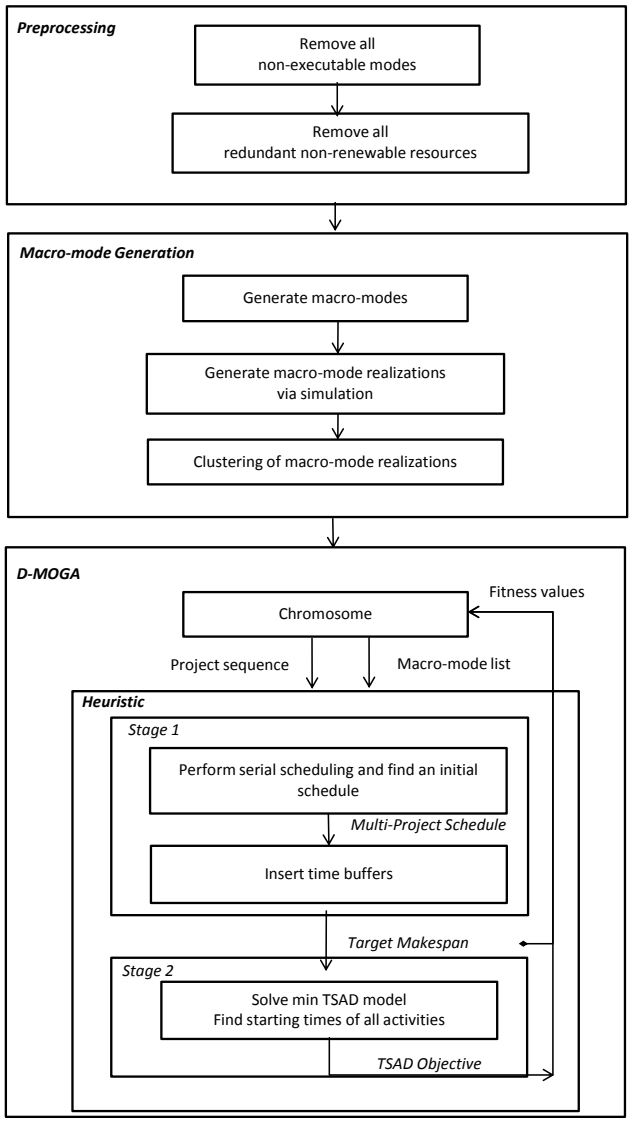

Figure 4.2: Flow chart of the 2-stage decomposition approach

4.1 2-Stage decomposition

Whole procedure begins by preprocessing methods as discussed in Sprecher et al. [23]. The decomposition procedure is started after eliminating all non-executable modes due to insufficient resource capacities and removing the redundant non-renewable resource constraints.

Single project MRCPSPs are solved with artificial budget constraints of resource usage and their various combinations of resource allocation are evaluated in order

to form different macro-modes. In generation of macromodes, expected durations of modes are employed.

In MRCPSP models activity mode durations are assumed to be in their expected values, however, the variability of macro-modes associated with mode duration uncertainties is represented by simulations. Previously found macro-mode schedule is disrupted via random realizations of activity mode durations resulting in a disrupted schedule. Each such disrupted schedule is called a realization of the macro-mode and in each realization macro-mode can have different resource profiles. With a high number of randomly generated disruption cases, we obtain macro-mode realizations which we define as data points in the discrete probability distribution of macro-modes. A clustering procedure is employed in order to reduce the number of realizations when the computational burden of evaluating high number of realizations is troublesome. A brief summary is given in Algorithm 1:

Algorithm 1 2-Stage decomposition approach

Stage 0 - Data Reduction:
Remove all non-executable modes
Delete the redundant nonrenewable resources
Stage 1 - Macro-mode Generation and Realization:
for $s = 1$ to $|S|$ **do**
    Generate macro-modes for project$_s$
    Generate realizations for macro-modes obtained
    Apply K-means clustering to group realizations
end for
Apply transformation to resource profiles to eliminate time dimension
Decomposition MOGA
Fitness computation
for all chromosome $c \in$ Population **do**
    Stage 1a - Macro-Project Scheduling:
    Serial scheduling with respect to probability bounds
    Buffer insertion between projects
    Target makespan calculation
    Stage 2b - Min TSAD model
    for $s = 1$ to $|S|$ **do**
        Solve project$_s$ for min TSAD
    end for
end for

4.1.1 Data reduction

At the very beginning of the whole procedure, two of the preprocessing methods discussed in Sprecher et al. [23] are applied to each project $s \in S^a$ in order to reduce the data size. First, all non-executable modes are eliminated and then all redundant non-renewable resources are removed.

4.1.1.1 Eliminating non-executable modes

Comparing modes may show that some modes are dominated by the others in the sense that a dominated mode of an activity will perform worse than or at the best as good as the other modes of that activity regarding processing time and resource usage efficiencies. These dominated modes, also referred to as non-executable modes, can never be selected in an optimal schedule and hence are eliminated.

A mode m_i of an activity i can be non-executable with respect to either a renewable and/or a non-renewable resource. Mode m_i is a non-executable mode with respect to a renewable resource $r \in R$, if $w_{im_ir} > W_r$. Further, denoting minimal request of activity i for non-renewable resource n as $q_{in}^{min} = min\{q_{ijn}|j = 1, ..., |M_i|\}$, we call m_i non-executable with respect to non-renewable resource n if $\sum_{b=1;b\neq i}^{|V_s|} q_{bn}^{min} + q_{im_in} > Q_n$.

4.1.1.2 Eliminating redundant non-renewable resources

A non-renewable resource is redundant, if there is enough capacity to meet even the maximal demand possible.

Let maximal request of activity i for non-renewable resource n be denoted as $q_{in}^{max} = max\{q_{ijn}|j = 1, ..., |M_i|\}$. Non-renewable resource n is redundant, if $\sum_{i=1}^{|V_s|} q_{in}^{max} \leq Q_n$.

4.1.2 A shrinking method: macro-mode generation

Here the objective is to identify efficient macro-modes for each project. It should be noted that as the number of macro-modes per project increases, the overall

complexity of assigning a macromode to a project increases. With this in mind, we adopt the shrinking procedure by Speranza and Vercellis [3]. In this procedure, an efficient search for makespan and resource usage costs of macro-modes is performed and non-dominated macro-modes are generated. Two shrinking models M_s^1 and M_s^2, which utilize artificial mode costs and an alterable budget based on resource usages are used in the generation of macro-modes.

c_r^u refers to the variable cost of utilizing one unit of renewable resource r for one time period and c_n^u refers to the variable cost of consuming one unit of non-renewable resource n. Usage costs for renewable and non-renewable resources are incurred periodically throughout each project. It is assumed that an activity's consumption of non-renewable resources as well as the variable cost distribution associated with its consumption are uniform over the execution period of that activity. c_r^u and c_n^u are both assumed to be 3 in our experiments in Section 6.

4.1.2.1 Shrinking models

Model M_s^1:

$$x_{ijt} = \begin{cases} 1 & \text{if activity } i \text{ starts at time } t \text{ in mode } j \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$\min T_{z_s} \tag{4.2}$$

$$\text{s.t.} \qquad T_i = \sum_{j \in M_i} \sum_{t=e_i}^{l_i} t x_{ijt} \qquad i \in V_s, \tag{4.3}$$

$$D_i = \sum_{j \in M_i} \bar{d_{ij}} \sum_{t=e_i}^{l_i} x_{ijt} \qquad i \in V_s, \tag{4.4}$$

$$T_k - T_i \geq D_i \qquad (i,k) \in P_s, \tag{4.5}$$

$$\sum_{i \in V_s} \sum_{j \in M_i} \sum_{\theta=\max(e_i, t-\bar{d}_{ij}]+1)}^{\min(l_i+\bar{d}_{ij}-1, t)} w_{ijr} x_{ij\theta} \leq W_r \qquad r \in \mathcal{R}, t \in \mathcal{T}_s, \tag{4.6}$$

$$\sum_{i \in V_s} \sum_{j \in M_i} q_{ijn} \sum_{t=e_i}^{l_i} x_{ijt} \leq Q_n \qquad n \in \mathcal{N}, \tag{4.7}$$

$$\sum_{j \in M_i} \sum_{t=e_i}^{l_i} x_{ijt} = 1 \qquad i \in V_s, \tag{4.8}$$

$$g_{ij} = \sum_{r \in \mathcal{R}} \bar{d}_{ij} w_{ijr} c_r^U + \sum_{n \in \mathcal{N}} q_{ijn} c_n^U \qquad j \in M_i, i \in V_s, \tag{4.9}$$

$$\sum_{i \in V_s} \sum_{j \in M_i} \sum_{t=e_i}^{l_i} g_{ij} x_{ijt} \leq k_s \tag{4.10}$$

$$x_{ijt} \in \{0,1\}, \quad i \in V_s, j \in M_i, t \in \mathcal{T}_s \tag{4.11}$$

As in a typical MRCPSP, constraints regarding start times (4.3), durations (4.4), precedence relations (4.5), assignments (4.8), resource capacities (4.6) and (4.7) and integrality (4.11) are included in Model M_s^1. For project s, there is also an artificial budget k_s constraining the resource usages. Considering renewable and non-renewable resources, variable usage costs, g_{ij}, are calculated as given in (4.9) and are constrained by a budget k_s as given in (4.10).

Model M_s^2:

$$\min k_s \tag{4.12}$$

$$\text{s.t.} \quad \sum_{i \in V_s} \sum_{j \in M_i} g_{ij} \sum_{t=e_i}^{l_i} x_{ijt} = k_s \tag{4.13}$$

$$T_{z_s} \leq T_s^h \tag{4.14}$$

$$(4.3), (4.4), (4.5), (4.6), (4.7),$$

$$(4.8) \text{ and } (4.11) \text{ from Model } M_s^1$$

In Model M_s^2, the budget is not included as a constraint (4.13) but it is taken as the objective (4.12). An additional constraint (4.14) is introduced here, which sets an upperbound, T_s^h, on the makespan of the project. It should be remembered that there is a negative relation between project makespan and budget.

4.1.2.2 Macro-mode generation method

Algorithm 2 Macro-Mode Generation

```
1: for s = 2 to |S| − 1 do
2:     for all activity i ∈ V_s do
3:         for all mode j ∈ M_i do
4:             Calculate g_ij
5:         end for
6:     end for
7:     Shift g_ij   i ∈ V_s, j ∈ M_i to 0
8:     Remove all inefficient modes i ∈ V_s, j ∈ M_i
9:     Calculate k_s^max
10:    Solve Model M_s^1 with k_s = k_s^max and find D_s^min
11:    Solve Model M_s^1 with k_s = 0 and find D_s^max
12:    for τ = D_s^min to D_s^max step size= 1 do
13:        Solve Model M_s^2 with T_s^h = τ
14:        if k_s decrease and a new macro-mode is generated
15:    end for
16: end for
```

Macro-mode generation procedure summarized in Algorithm 2 is initialized by calculating the artificial mode costs as expressed in (4.9). Then artificial mode costs

are shifted to zero by calculating minimal artificial mode costs g_i^{min} for each activity $i \in V_s$ (4.15) and subtracting it from each artificial mode cost for each mode $j \in M_i$ (4.16).

$$g_i^{min} = min\{g_{ij} | j = 1, ..., M_i\} \qquad i \in V_s \tag{4.15}$$

$$g_{ij} = g_{ij} - g_i^{min} \qquad j \in M_i, i \in V_s \tag{4.16}$$

Later, inefficient modes are identified examining their durations and artificial costs. A mode j of activity i is considered as inefficient, if there exists a mode h of activity i such that $\bar{d}_{ij} \geq \bar{d}_{ih}$ and $g_{ij} \leq g_{ih}$. After removing all the inefficient modes, maximum budget required, k_s^{max} is computed by calculating maximal artificial mode costs g_i^{max} for each activity $i \in V_s$ (4.17) and adding them up (4.18).

$$g_i^{max} = max\{g_{ij} | j = 1, ..., M_i\} \qquad i \in V_s \tag{4.17}$$

$$k_s^{max} = \sum_{i \in V_s} g_i^{max} \qquad j \in M_i, i \in V_s \tag{4.18}$$

Considering duration range $[D_s^{min}, D_s^{max}]$ for T_s^h, the upper limit on the makespan is computed by solving Model M_s^1 once setting k_s equal to 0 and once setting it equal to k_s^{max}. Duration range for T_s^h signifies the durations for possible macro-modes to be generated. Solving Model M_s^2 gives a schedule with a makespan equal to T_s^h and most efficient mode selections regarding the resource usage cost budget. Starting from D_s^{min}, T_s^h is increased by one at each step until D_s^{max} is reached. At each step, Model M_s^2 is solved and if k_s value is lower than that in the previous solution, a new macro-mode v is generated using the duration and the renewable resource profile obtained in the solution of the model as shown in (4.19) and (4.20).

$$q_{svn} = \sum_{i \in V_s} \sum_{j \in M_i} q_{ijn} \sum_{t=e_i}^{l_i} x_{ijt} \tag{4.19}$$

$$w_{svrt} = \sum_{i \in V_s} \sum_{j \in M_i} \sum_{\theta = \max(e_i, t - \bar{d}_{ij} + 1)}^{\min(l_i + \bar{d}_{ij} - 1, t)} w_{ijr(t-\theta+1)} x_{ij\theta} \tag{4.20}$$

This condition that k_s decreases is considered in order to ignore schedules representing identical mode selections but having different durations and, of course, it is not checked for the first solution of Model M_s^1 where $T_s^h = D_s^{min}$ and an initial k_s value to be compared with has not been determined yet.

4.1.3 Macro-mode realization generation

The macro-modes generated with the expected mode duration assumption are unable to cope with disruptions resulting in higher resource usage or makespan level. In that respect, in this stage, by simulation we are generating disruption cases resulting from random generation of mode durations with each macro-mode profile. Resulting disrupted macro-modes are called realizations. A disruption in a pregiven macro-mode schedule may result in (a) higher makespan, (b) higher resource consumption or (c) both. Each realization corresponds to a disruption scenario and thus with a high number of realizations we represent the macromode as a combination of random variables with known discrete values. The probability of each case corresponds to its frequency in the overall realizations.

The makespan level (realized makespan) of a macro-mode is especially important in both setting a minimum makespan level and satisfying it with high probability. Thus makespan level and maximum resource consumption level are the main performance parameters of realizations hence it becomes possible to group the realization data or eliminate common elements through these parameters.

Considering common or similar resource and makespan levels among numerous realizations, it may become handy to group realizations and pick the most representative ones for computational purposes. Although for our algorithm clustering is not a necessary technique to adapt, higher number of realizations result in more usage of computational power. Hence after weighing the benefits and drawbacks of using a clustering techique to reduce the realization number, it has been decided to employ one. In the next subsection, we present the clustering procedure employed in detail.

4.1.4 Macro-mode realization clustering

Cluster analysis refers to techniques designed to find groups of similar elements within a data set, and its assignment to representative groups. Each of these groups is called a cluster and represents a region in which densitiy of objects are locally higher than in other regions. The goal is to achieve the greater similarity (difference) within (between) the clusters so that the clustering is more distinct.

Different types of clustering methods include hierarchical clustering, partitional procedures, exclusive, overlapping and fuzzy algoritms [24]. Hierarchical clustering methods produce a hierarchy of clusters from sub-clusters (smaller groups) within large clusters. Hierarchical methods include divisive and agglomerative approaches where the first approach progressively divides large clusters into smaller ones and the latter one starts from small sized clusters and iteratively adds similar elements to clusters. Partitional procedures essentially aim to divide the data set into a pre-determined number of groups. Exclusive clustering groups data in such an exclusive way that a data point belongs to a single cluster. However, in overlapping algorithms a data point may belong to multiple clusters through the employement of fuzzy clusters. Finally in fuzzy clustering, every object belongs to every cluster with a membership weight. This broad suite of techniques is employed in many fields where the interpretation of data is especially important. Information retrieval, psychology, biology and medicine are among the many fields where cluster analysis is frequently used.

One of the oldest and most widely used clustering procedures is K-means, an exclusive partitional clustering algorithm that creates a one-level partitioning of data objects and finds a user-specified number (K) of clusters [24]. K-means is a local search procedure and the cluster number K fed into the algorithm is an input. We perform experiments of different cluster numbers in Section 6.5.4.

K-means clustering algorithm aims to reduce total sum of squares (TSS) which is a distance quantity defined in 4.21. Algorithm starts by randomly selecting K centroids. Each point is then assigned to its closest centroid where closeness is defined as the minimum euclidian distance between a point and a centroid. Centroids

are then updated with the change of data points belonging to each cluster. This assigment is repeated until no change in the centroids occur.

$$TSS = \sum_{k \in K} \sum_{p \in Cluster_k} \sqrt{(Centroid_k - p)} \tag{4.21}$$

One of the drawbacks of K-means clustering is the possibility of obtaining empty clusters if case there are no points assigned to any cluster. At the end of algoritm, some clusters may be empty which may result in an undesired solution. In that case, a replacement stategy can be employed. There are two commonly used strategies to prevent empty clusters: First is to find the farthest away point from all centroids and assign it to the empty cluster- that strategy is beneficial since it at least reduces TSS . Second is to choose the replacement centroid from the cluster that has the highest sum of square errors. In case empty clusters are found, we employ first strategy to obtained K-means solution, thus prevent void clusters. Algorithm 3 presents the described K-means algorithm.

Algorithm 3 K-Means clustering algorithm

```
Select K points as initial centroids
repeat
    Form K clusters by assigning each point to its closest centroid
    Recompute the centers of each centroid
until Centroids do not change
if Empty clusters found then
    for all empty cluster c ∈ C do
        Find the farthest point to all other points in data
        Assign that point to empty cluster
    end for
end if
```

4.1.5 Macro-mode generation, realization and clustering example

Here we present the same example given in [1] for the application of macro-mode generation subprocedure. A project network with six activities is presented in Figure 4.3. Activities 0 and 5 are dummy activities. There is one renewable resource $r^{'}$ with 10 units capacity and one non-renewable resource $n^{'}$ with 50 units capacity.

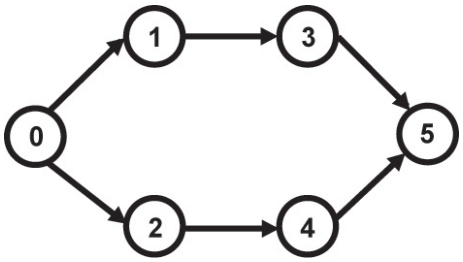

Figure 4.3: Macro-mode generation example network

Usage costs are given as: $c^u_{r'} = 1$ and $c^u_{n'} = 2$. Duration and resource requirements data about execution modes of activities are shared in Table 4.1 along with mode cost calculations and shifted mode cost values. Note that dummy activities have 0 mode costs.

i	j	d_{ij}	$w_{ijr'}$	$q_{ijn'}$	$(d_{ij})(w_{ijr'})(c^u_{r'}) + (q_{ijn'})(c^u_{n'}) = g_{ij}$	g'_{ij}
1	1	3	3	2	(3)(3)(1)+(2)(2)= 13	3
	2	4	2	1	(4)(2)(1)+(1)(2)= 10	0
2	1	5	4	3	(5)(4)(1)+(3)(2)= 26	8
	2	7	2	2	(7)(2)(1)+(2)(2)= 18	0
3	1	4	3	4	(4)(3)(1)+(4)(2)= 20	8
	2	6	2	0	(6)(2)(1)+(0)(2)= 12	0
4	1	1	2	0	(1)(2)(1)+(0)(2)= 2	0
	2	2	0	3	(2)(0)(1)+(3)(2)= 6	4

Table 4.1: Macro-mode generation example data

By using (4.17) and (4.18), k_s^{max} is calculated to be 23. Solving Model M_s^1 once setting k_s equal to 0 and once setting it equal to 23 provides a duration range of $[7, 10]$. Then four macro-modes presented in Figure 4.4 are generated by solving Model M_s^2 with the upperbound values in the duration range set. At all steps following the first one, a decrease in the objective function value is observed thus making all the macro-modes generated acceptable as defined in the procedure.

Afterwards for each macro-mode generated their realizations are generated by simulation. In each simulation, a realization of all activities in their selected modes are obtained and a disrupted schedule is obtained. Figure 4.5 shows a visual example of macro-mode 1 and one of its realizations. Activities that are marked red, have a longer duration than their expected duration's in this particular realization example.

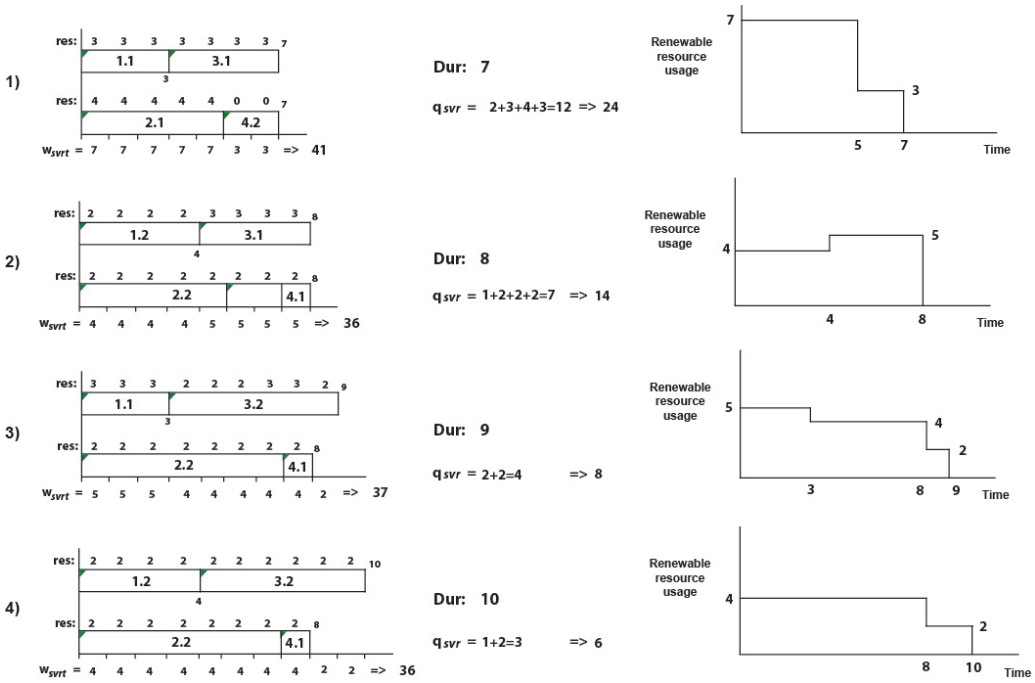

Figure 4.4: Schedules and resource profiles for generated macro-modes [1]

Similarly K realizations are generated. Furthermore, representative realizations in K realizations are determined via K-means algoritm. Suppose pre-given number of clusters is 10, then total realization number of each macro-mode has been reduced to 10.

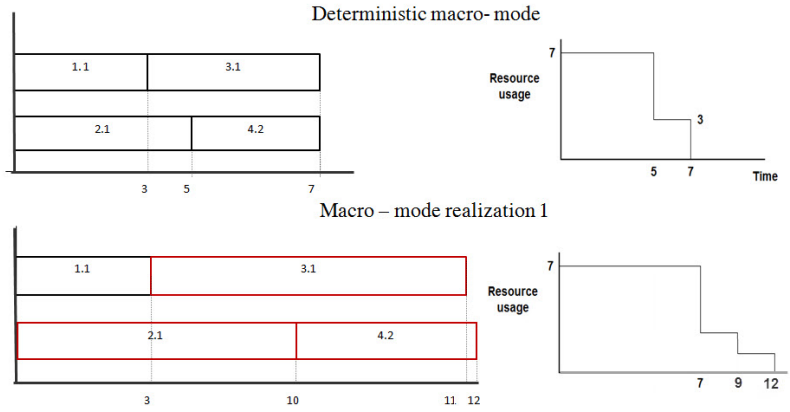

Figure 4.5: An example of macro-mode and one realization

4.2 Macro-project scheduling

The macro-project scheduling model described here is a stochastic extension of MR-CMPSP. The amount of renewable resource r utilized by an activity performed in one of its modes is both time and distribution dependent. The randomness in the resource amount increases the complexity of the problem, which is NP-hard in the deterministic case. When it comes to scheduling resource profiles that are random variables, two possible approaches emerge. The first option is to use the mean or another frequently used statististical measure and solve the problem as in the deterministic case. The other option is to use the random variables' probability information and compute joint probabilities when determining feasibility of constraints. Although the latter is much harder to compute, probabilistic constraints are widely employed in literature (see e.g., [17], [19]) and probability limits give room for parametric analysis and allow for observing the difference in resulting schedule.

It should be noted that the amount of randomness is a key issue affecting the model. As macromodes are turned into realizations in which a specific macromode may result in multiple resource profiles, the macro-modes themselves become combinations of random variables. In addition to resulting makespan value, resource dimension of each time point is a random variable respresented by realizations.

Model MP**:**

$$\tilde{x}_{svt} = \begin{cases} 1 & \text{if project } s \text{ starts at period } t \text{ in macro-mode } v \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

$$\min TSAD = \sum_{s\in S}\sum_{i\in V_s}\sum_{k\in K}|T_{si} - E^k_{si}| \tag{4.23}$$

$$\min TargetMakespan \tag{4.24}$$

s.t.
$$T_s = \sum_{v\in M_s}\sum_{t=e_s}^{l_s} t\tilde{x}_{svt} \qquad s \in S, \tag{4.25}$$

$$D_s = \sum_{v\in M_s} d_{sv}\sum_{t=e_s}^{l_s}\tilde{x}_{svt} \qquad s \in S, \tag{4.26}$$

$$T_k - T_s \geq D_s \qquad (s,k) \in P_s, \tag{4.27}$$

$$T_{si} \geq T_s \qquad i \in V_s, s \in S, \tag{4.28}$$

$$Prob(\sum_{s\in S}\sum_{v\in M_s}\sum_{\theta=\max(e_s,t-d_{sv}+1)}^{\min(l_s+d_{sv}-1,t)} w_{svr(t-\theta+1)}\tilde{x}_{sv\theta} \leq W_r) \geq Limit1 \qquad r \in \mathcal{R}, t \in \mathcal{T}, \tag{4.29}$$

$$Prob(\max_{s\in S} T_s < TargetMakespan) \geq Limit2 \tag{4.30}$$

$$\sum_{s\in S}\sum_{v\in M_s} q_{svn}\sum_{t=e_s}^{l_s}\tilde{x}_{svt} \leq Q_n \qquad n \in \mathcal{N}, \tag{4.31}$$

$$\sum_{v\in M_s}\sum_{t=e_s}^{l_s}\tilde{x}_{svt} = 1 \qquad s \in S, \tag{4.32}$$

$$w^{min}_{svr\theta} \leq w_{svr\theta} \leq w^{max}_{svr\theta},\ d^{min}_{sv} \leq d_{sv} \leq d^{max}_{sv}, \tag{4.33}$$

$$\tilde{x}_{svt} \in \{0,1\}, T_{si}\ discrete, s \in S, \quad v \in M_s, t \in \mathcal{T} \tag{4.34}$$

Constraint set (4.25) represents the start times and constraint set (4.26) represents the durations for the projects. Note that d_{sv} is a random variable, thus durations of projects and their starting times consequently become random. Constraint set (4.27) ensures the precedence relationships between the projects. Constraint set (4.28) assures the activities belonging to a project start later than the project itself. Probabilistic constraint set (4.29) is the capacity constraint for the renewable resources and constraint set (4.29) is the capacity constraint for the non-renewable resources. Constraint (4.30) assures that the makespan of the multi-project schedule is less than the target with a given probability limit. Constraint set (4.32) ensures that for each project a macro-mode alternative is selected and is started at some point in the interval $[e_s, l_s]$. Although W_r, the renewable resource availability for each renewable resource r is pre-known, still a probabilistic constraint (4.29) is employed because of the uncertainty in resource usage levels of macro-modes.

4.3 Decomposition based multi-objective GA

As the name implies, GAs are global optimization search techniques inspired by natural selection and evolution. GAs compose a pool of possible solutions named chromosomes which are based on a particular representation scheme. Starting by generating an initial population of solutions, algorithm progressively updates solution pool by reproduction procedures (generating new individuals by modifying parent individuals selected from population) or mutation procedures which modify an existing individual in order to generate diversity.

Over the last decades, GA has been employed in many fields. One of the reasons of their popularity is thier power in finding a global optimum and their flexibility in their applicability to discrete spaces, nonlinear constraints, etc. [25]. The first principles of GA were conceived in the seventies, Davis [26] composed the first application of GA to scheduling context. Many researchers have developed GA approaches to both single project scheduling and multi-project scheduling [27]. Hartmann applied GA to project scheduling problems with multiple modes [28].

Dealing with multi-objectives (in our case makespan and total sum of absolute deviations) differentiates the algoritm's structure. Presence of multi-objectives affect especially parent selection and population update procedures and also definition of an elite. A chromosome designated as an elite chromosome in a generation, is reproduced in the next generation because of its desired fitness value. However, in multi-objective case defining an elite can become troublesome since there is no single dominating solution. For that purpose, algorithms, which use non-dominated frontiers corresponding to elite solutions, have been developed. A very successful MOGA algoritm (NSGA-II), suggested by Deb et al. [29], uses an explicit diversity generation procedure along with an elite-preservation procedure. Thus we adopt here NSGA-II in our solution procedure. Figure 4.6 presents a general flow of the proposed MOGA. Chromosome evaluation and population management are explained in detail in 4.3.2 and 4.3.6, respectively.

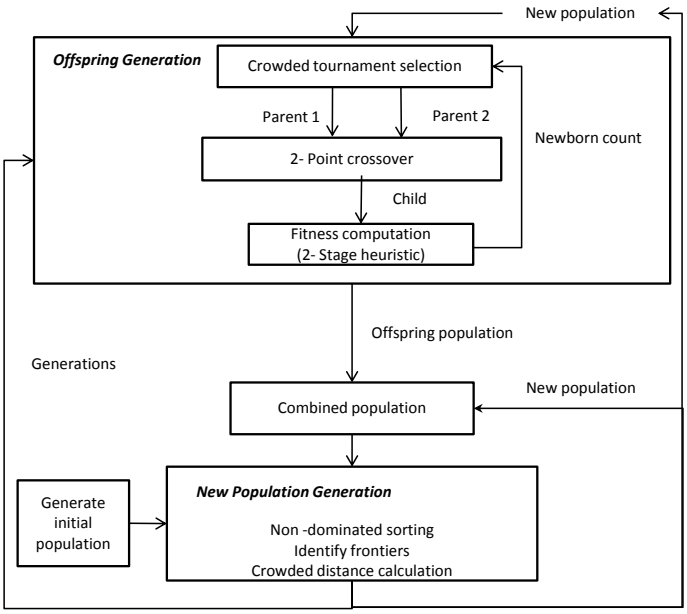

Figure 4.6: Flow chart of the decomposition approach MOGA

4.3.1 Chromosome representation

First decision on the design of GA is the chromosome representation. In RCPSP literature, there has been various representation of an individual. Although Kolisch and Hartmann [30] distinguish up to 5 different schedule representations, activity-list representation and random key representation are the most widely used ones [31]. Although these two representations have their own advantages, simulation experiments performed by Kolisch and Hartmann [32] reveal that performance of activity-list representation is superior to other discussed representations. In activity-list representation an activity's position represents its relative priority to other activities. In our case latest activity to be scheduled is the activity at the end of the activity list.

In MRCPSP a single list of activities is not sufficient with the possible selection of modes. Consequently, as in Alcaraz [33] we use two lists for representation of an individual: a list of projects (P_c) and an ordered list of macro-mode assignments (MM_c). The mode assignment list represents the macro-mode executed for each project. We do not use an ordered list for projects because it is easier to produce offsprings from non-ordered lists. By definition there are no-precedence constraints

between the projects so there is no precedence-feasibility conditions required. However, for a generated chromosome non-renewable resource feasible, so non-renewable resource constraints must be evaluated for each chromosome.

5	4	1	2	7	6	3
1	1	3	2	1	2	1

Figure 4.7: Chromosome representation

An example for a schedule representation with 7 projects and with macro-modes each is given in Figure 4.1. Top row of the chromosome represents the priority sequence for the activities meaning that project 5 has to be scheduled first and then projects 4,1,2,7,6, and 3 have to follow in that order. Bottom row of the chromosome represents the list of macro-mode selections for the activities. For example, project 2 is executed in its second macro-mode and project 7 is executed in its first macro-mode.

4.3.2 Evaluation of chromosomes

A two stage serial scheduling heuristic, described in the following section is used to calculate the objective values of a given chromosome. First part of the heuristic, which consists of serial scheduling and buffer insertion, determines the target makespan of the schedule (section 4.3.3.2), thus the first objective. The second objective corresponds to the objective value of the minimum deviation linear programming model (section 4.3.3.4) used to find robust starting times of activities.

4.3.3 A 2-stage serial scheduling heuristic

Here, the objective is to find a schedule that satisfies the renewable resource bounds with a predetermined probability and also finds the minimum makespan bound associated with the schedule obtained. A two-stage serial scheduling heuristic is developed here, where in the first stage a serial scheduling routine along with buffer

insertion is performed and for the purpose of improving solution robustness constitutes the second stage. Makespan bound should be satisfied with a probability of not exceeding as in the resource case. Using similarly the joint probabilistic constraints as Bruni et al. [17], the heuristic schedules each project one by one from the sequence list. If a project does not satisfy the probabilistic constraint, then the algoritm passes to the next project. Thus the proposed heuristic limits the schedule's probability of exceeding resource capacity and targeted makespan.

It should be noted that computation of probabilistic constraints becomes a challenge considering high number of random variables and the lack of a probability distribution. With only discrete cases in hand, probability computation necessitates exhaustive enumaration of possible cases thus becoming very expensive computationally. In order to overcome this challenge, we propose a resource profile transformation to decrease the number of random variables within a macromode. We propose to take solely the maximum of a random variable and its duration. So we represent the randomness in resource consumption of each time point by a single random variable. The details of this transformation may be found in the resource profile transformation section.

4.3.3.1 Resource profile transformation

Previously generated macro-modes contain time dependent resource profiles whose resource consumption at time t is a random variable that can take on different values throughout realizations. Randomness in resource consumption at each time point increases the complexity of resource feasibility constraints, decreases the performance of the heuristic and consequently takes away the benefit of 2- stage decomposition. On the other hand, a direct observation in later scheduling steps shows us that when scheduling 2 or more time-dependent resource profiles at same time point whilst sharing the same resource, the maximum amount of resource used by profiles becomes an important indicator on whether an overlapping is possible or not. In other words, the over-use of a resource at a time point generally caused by a conflict of higher resource levels of macromodes. Based on this observation, we pro-

pose a transformation of time dependent resource profiles into time constant profiles where the amount of resource used is equal to the maximum resource consumption in time-dependent profile.

Figure 4.8 below illustrates the described idea.

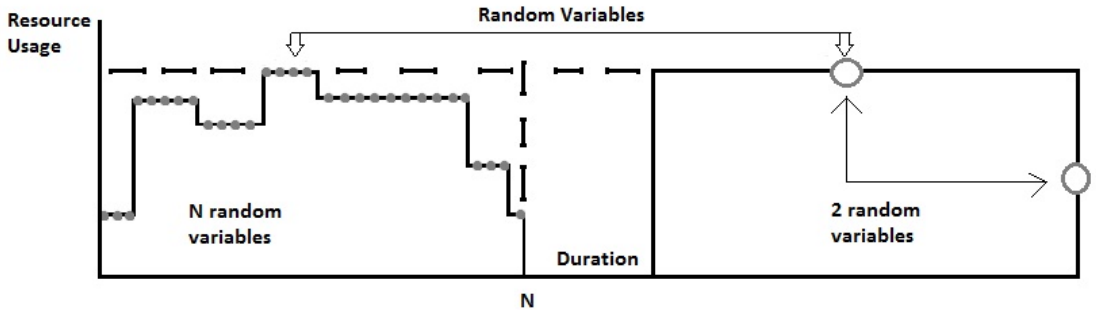

Figure 4.8: Resource profile transformation

Via the proposed transformation, we replace constraints (4.29) with (4.35) :

$$Prob(\sum_{s\in S} \sum_{v\in M_s}^{\min(l_s+d_{sv}-1,t)} w_{svr}\tilde{x}_{sv\theta} \leq W_r) \geq ResourceProbLimit \quad r \in \mathcal{R}, t \in \mathcal{T} \quad (4.35)$$

One possible drawback of this transformation is that it tends to overestimate resource consumption by taking maximum values of each realization. A schedule with overestimated resource consumption may result ina higher makespan than in otherwise would. However, by imposing different limits posed on probabilistic constraints(4.30,4.35), one may avoid this overestimation. If the resulting schedule highly overestimates the resource usage, probability limits can be decreased in order to compenstate overestimation. At the end of this step, time-dependent random resource profiles of macromodes are transformed into rectangular resource profiles with resource consumption levels and makespan being represented by discrete random variables.

4.3.3.2 Scheduling stage 1- serial scheduling

In this stage given sequence and macro-mode list, projects with their selected macro-modes are serially scheduled one by one. Algorithm 4 presents the the flow of serial scheduling stage.

Algorithm 4 Serial scheduling

```
1: Already Scheduled Projects S_sch = ∅;
2: feasibletimepoints = ∅
3: for all Projects s ∈ S do
4:     for time t ∈ feasibletimepoints do
5:         if ∀r ∈ R, ResourceProbabilityViolation(S_sch, s, i, r) = false;
           then
6:             Schedule Project s to time t
7:             feasibletimepoints.Add(t + ExpDuration[s]);
8:             S = S \ s;
9:             S_sch = S'_sch ∪ s;
10:        end if
11:    end for
12:    Sort feasibletimepoints in non-decreasing order;
13: end for
14: Find minimum Target Makespan, such that:
15: Probability(∑_{cp∈criticalpathprojects} cp_makespan ≤Target Makespan)≥ Limit;
```

In order to evaluate at time t, if renewable resource proababilistic constraint is violated or not, projects that simultaneously share resources should be found. We call RSL_{rt}(resource sharing list), a list of projects using resource r in interval $[t, ExpDuration[s] + t]$. These lists should be formed first and then the probability that the sum of resource usages of all projects belonging RSL_{rt} should be computed. Figure 4.9 presents resource sharing lists on an example.

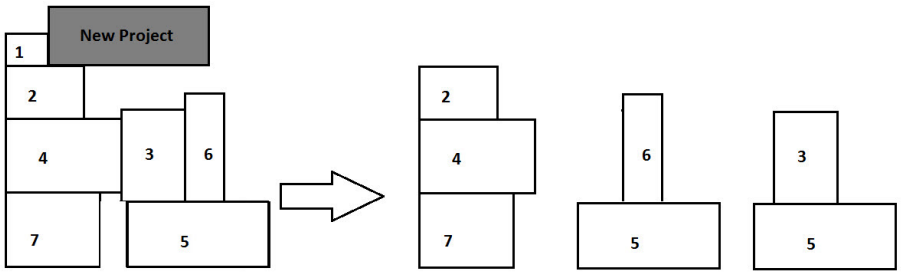

Figure 4.9: Example - identifying resource sharing lists

Computation of resource violation probability is given in Algorithm 5.

Algorithm 5 ResourceProbabilityViolation (Scheduled projects S_{sch}, time t, project s , renewable resource r)

1: $violation = false$;
2: A list of all RSL_{rt} : $All_{RSL_{rt}} \rightarrow 0$;
3: **if** $\exists p \in S_{sch}$ using resource at $[t, t + Expdur(s)]$ **then**
4: **if** $All_{RSL_{rt}}$ is empty **then**
5: new RSL_{rt}; $RSL_{rt}.Add(p)$;
6: $All_{RSL_{rt}}.Add(RSL_{rt})$;
7: **else if** $\exists a \in RSL^i_{rt} \in All_{RSL_{rt}}$ s.t. a and p use simultaneously use r **then**
8: $RSL^i_{rt}.Add(p)$;
9: **else**
10: new RSL_{rt}; $RSL_{rt}.Add(p)$;
11: $All_{RSL_{rt}}.Add(RSL_{rt})$;
12: **end if**
13: **end if**
14: **for all** $RSL_{rt} \in All_{RSL_{rt}}$ **do**
15: $RSL_{rt}.Add(s)$;
16: Compute $Probability(\sum_{p \in RSL_{rt}} p_r \leq W_r))$;
17: **if** $Probability(\sum_{p \in RSL_{rt}} p_r \leq W_r) \leq Limit)$ **then**
18: $violation = true$;
19: break;
20: **end if**
21: **end for**
22: Return violation;

We assume that resource flow is fixed after obtaining an unbuffered schedule hence even with the insertion of time-buffers between projects, resource flow stays the same. The list of projects where same resource flow is taking place are denoted as resource flow sequences(RFS).Figure 4.10 illustrates four RFS's.

4.3.3.3 Scheduling stage 2 - buffer insertion

When robustness is desired for a schedule, one of the most common approaches is to take an unbuffered schedule and insert time buffers to necessary time periods. Although many robustness measures are reported in literature, many of these measures aim to minimize the deviation between preset and realized starting times. In a robust schedule, disruptions in one activity affect another the minimum amount

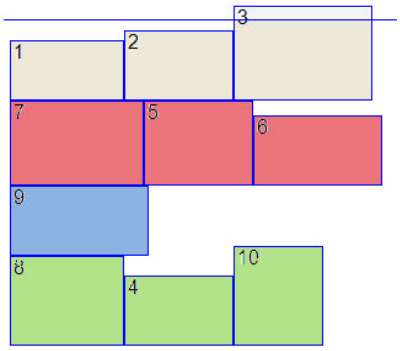

Figure 4.10: Example - resource flow sequences

possible so that the preset starting times of activities are not much different than in reality.

In literature there have been various approaches presented on generating buffered schedules. Van de Vonder et al. [15] compare various buffering heuristics by simulation based experiments. Their results favored the use of starting time criticality (STC) heuristic. Therefore here we adopt a similar approach as in STC while inserting buffers to the schedule generated by Stage 1.

The basic idea is to start from an initial unbuffered schedule and iteratively create intermediate schedules by adding a one-unit time buffer in front of that project that needs it the most in the current intermediate schedule, until adding more safety would no longer improve stability. For each project its scheduled time is denoted as $\mathbf{s}_j$. The actual starting time of project j which is a random variable is denoted as s_j.

To quantify for each project how critical its current starting time is in the current intermediate baseline schedule, stc_j is defined as the probability that project j cannot be started at its scheduled starting time.

$$stc(j) = P(\mathbf{s}_j \geq s_j) \tag{4.36}$$

If we define $k(i, j)$ the event that predecessor i disturbs the planned starting time

of project j, then the probability of that event to happen:

$$P(k(i,j)) = P(s_i + d_i + LPL(i,j) \geq \mathbf{s}_j) \tag{4.37}$$

Where $LPL(i,j)$ denotes total sum of durations in the longest path between from i to j and defined as:

$$LPL(i,j) = \sum_{h \in LongestPath(i,j)} \bar{d}_h \tag{4.38}$$

In STC heuristic as defined in Van de Vonder [15] STC is measured as:

$$stc(j) = P(\sum_{i,j \in A} d_i \geq \mathbf{s}_j - s_i - LPL(i,j)) \tag{4.39}$$

This measure approximates the probability by addition thus does not describe the actual probability. For that reason we propose to use a different formulation for criticality of an project:

$$stc(j) = 1 - \prod_{i \in j'sPredecessors} (1 - P(d_i \geq \mathbf{s}_j - s_i - LPL(i,j))) \tag{4.40}$$

The flow of buffering heuristic is given in Algorithm 6.

Algorithm 6 Buffering Heuristic

```
For each project s ∈ S calculate stc_s and store in array_stc
while array_stc[index] ≤ 0 do
        index = 0
        Sort array_stc in decreasing order, choose p ∈ S with highest stc
        if adding 1 time unit in front of p is feasible then
                Shift p and its successors by 1 time unit
                Update array_stc
                Pick the next project with highest stc
        end if
end while
```

Unbuffered and buffered starting times of each project s is denoted as ST_s^{unbuff} and ST_s^{buff} and these project stating values are used in the following model. Note

that RFS do not change in the buffer insertion step.

4.3.3.4 Scheduling individual projects for TSAD minimization

After setting the resource capacities and the start times of the projects, a separate min TSAD model is solved for each project. Each project is scheduled individually therefore an individual target finish date for each project must be set. We define $Target_s$ as below:

$$Target_s = \begin{cases} ST_p^{buffered} & if\ \exists \text{ project } p \text{ immidiately preceding } s \text{ in any } RFS \\ Target\ Makespan & \text{Otherwise} \end{cases} \tag{4.41}$$

Note that a mode index is not employed in this formulation, since the modes of the activities are already fixed in the generation of the macro-modes. Additionally since this model is solved for each project, project index s is fixed. For project s the corresponding Model S_s is given below:

Model S_s:

$$x_{it} = \begin{cases} 1 & \text{if activity } i \text{ starts at period } t \\ 0 & \text{otherwise} \end{cases} \tag{4.42}$$

$$\min TSAD_s = \sum_{i \in V_s} \sum_{k \in K} |T_i - E_i^k| \tag{4.43}$$

s.t.

$$T_i = \sum_{t=e_i}^{l_i} t x_{it} \qquad i \in V_s, \tag{4.44}$$

$$T_j - T_i \geq \bar{d}_i \qquad (i,j) \in P_s, \tag{4.45}$$

$$\sum_{i \in V_s} \sum_{\theta = \max(e_i, t-\bar{d}_{ij}+1)}^{\min(l_i+\bar{d}_{ij}-1, t)} w_{ijr} x_{ij\theta} \leq \tilde{W}_{rt}^s \quad r \in \mathcal{R}, t \in \mathcal{T}_s, \tag{4.46}$$

$$\sum_{t=e_i}^{l_i} x_{it} = 1 \qquad i \in V_s, \tag{4.47}$$

$$T_i \leq Target_s \qquad i \in V_s, \tag{4.48}$$

$$x_{it} \in \{0,1\}, t \in \mathcal{T} \tag{4.49}$$

The constraints for project duration (4.3), start time (4.4), precedence (4.5), assignment (4.8) and integrality (4.11) constraints as in Model M_s^1 are included. The right hand side of resource capacity constraints in (4.46) in Model S_s are different from the ones in Model M_s^1. $\tilde{W}_{rt}^s$ in constraint (4.46) are set as the average values of the resource profiles of macro-mode assigned for project s. The renewable resource capacity constraint does not (4.46) have a time index through resource profile transformation in Section 4.3.3.1. Constraint 4.48 assures the non-violation of the previously found target makespan and buffered project starting time values. After solving model S_s for each $s \in S$, total $TSAD$ value is the sum of each $TSAD_s$ value of all projects.

$$TSAD = \sum_{s \in S} TSAD_s \tag{4.50}$$

4.3.3.5 Heuristic example

In this section, a visual example of described heuristic is presented. Suppose chromosome c, composed of project list P_c and macro-mode assignment list MM_c is given as:

$$P_c = \{3, 6, 8, 5, 2, 10, 1, 9, 7, 4\} \tag{4.51}$$

$$MM_c = \{3, 1, 2, 2, 1, 3, 3, 2, 1, 1\} \tag{4.52}$$

Projects in their respective macro-mode selections in MM_c, are scheduled by the order of P_c. The output of Algorithm 4 is an unbuffered schedule which then becomes the input of Algoritm 6, hence a buffered schedule is generated. Figure 4.11 illustrates a the output of stage 1 vs. the output of stage 2 of the heuristic.

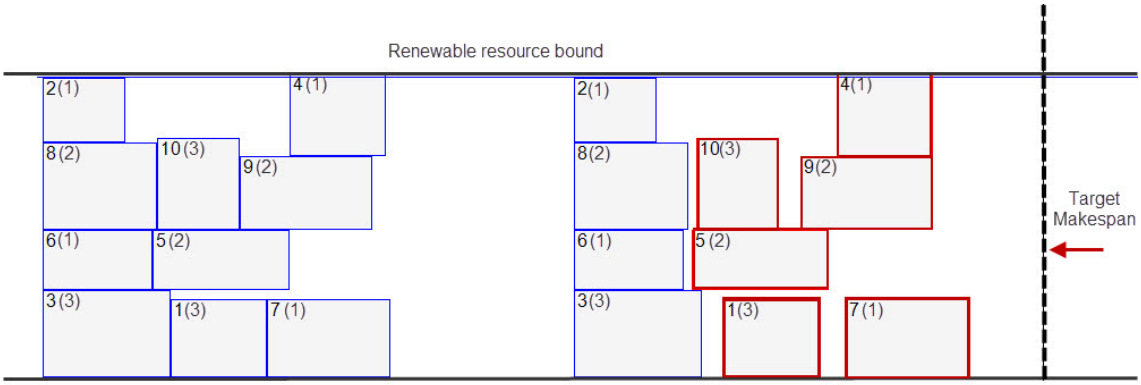

Figure 4.11: Example : non-buffered schedule vs. buffered schedule

At the end of stage 2 of the heuristic, starting times of the projects and a target overall makespan is determined. Afterwards for each project $p \in P_c$, Model S_s is solved. For example, in case of project 7, given its buffered starting time 15 and *Target Makespan* determined as 21, the $Target_s$ parameter in constraint (4.48), is set as : $21 - 15 = 6$.

4.3.4 Crossover

Crossover operators are extremely important in creating diversified generations and one of the most important factors on GA's success. Here we use 2-point crossover considering that there are no precedence constraints among projects. In the 2-point crossover procedure, primarily 2 random points (genes) are selected. First parent choromosome's genes exluding the ones between two random points, are transferred to child chromosome.The remaining genes are transferred from the second parent so as to complete the child chromosome.

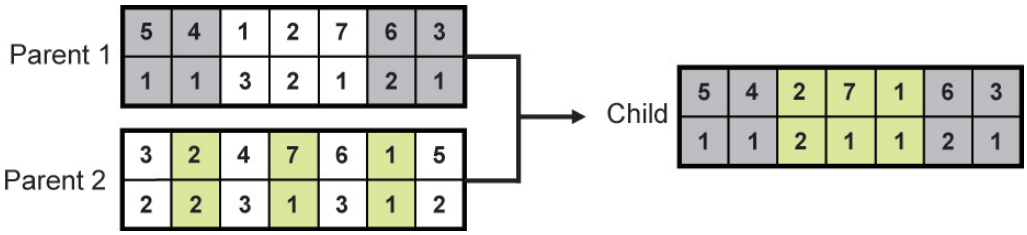

Figure 4.12: Crossover representation

4.3.5 Mutation

Modification of newly produced chromosomes plays an important part on increasing population's diversity. For that reason, we use two mutation operators : the swap mutation operator and the bit mutation operator.

Swap mutation: It is executed on the priority order list to obtain different sequences, which may or may not lead to a different schedule, by swapping the places of two activities randomly selected. For example, in Figure 4.13, activities 1 and 4 are swapped.

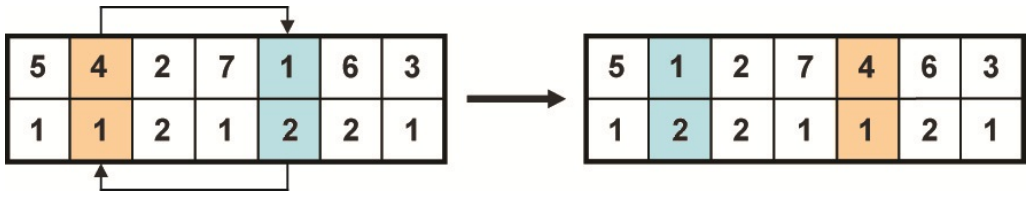

Figure 4.13: Swap mutation

Bit mutation: A project is selected randomly and the mode selection associated

with this project is replaced with another randomly chosen mode value as shown in Figure 4.14.

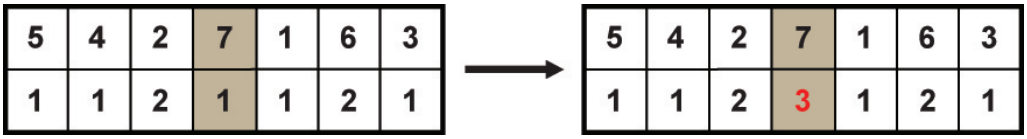

Figure 4.14: Bit mutation

4.3.6 Population management

Presence of multi-objectives complicates population management of MOGA's compared to single objective GA's. The steps of population management is given in Algorithm 7.

The exact offspring will depend on the chosen pair of solutions participating in a tournement and the chosen crossover and mutation operators, therefore parent selection is very important. We fist duplicate population and then select random pairs from this population. The idea here is to obtain different pairs of individuals. Afterwards for each solution pair a tournament selection is performed where a solution is selected if it belongs to a better frontier or has a higher crowding distance value. Details of the crowded tournament selection can be found in Algorithm 8.

Algorithm 7 MOGA Population Management

```
1: Set populations R_t, P_t, Q_t → 0;
2: Generate initial Population;
3: Pop = InitialPopulation;
4: for generation ∈ TotalGenerations do
5:     for q = 0 → NewBornCount do
6:         Crowded Tournament Selection for selecting 2 Parents;
7:         Generate a child chromosome c by 2-point crossover;
8:         if RandomNo ≤ SwapProbability then
9:             ⇒ SwapMutation
10:        end if
11:        if RandomNo ≤ BitProbability then
12:            ⇒ BitMutation
13:        end if
14:        Evaluate child c & find fitness pairs
15:        Q_t.Add(c);
16:    end for
17:    Combine Population and Offspring : R_t = P_t + Q_t;
18:    Perform non-dominated sort on R_t;
19:    Identify different frontiers F_all in R_t;
20:    Set new Population P_t → 0, count → InitialPopulationCount;
21:    while count > 0 do
22:        for Frontier F ∈ F_all do
23:            if |F| < count then
24:                Add members of F → P_t
25:                count ⇐ count − |F|
26:            else
27:                Calculate crowded distance(F)
28:                Add k = count individuals with highest cd_i to P_t
29:                count ⇐ 0
30:            end if
31:        end for
32:    end while
33: end for
```

Algorithm 8 Crowded Tournament Selection

Given population P_t composed of N individuals
Parent population $Par_t \rightarrow 0$;
Set new popularion $R_t = P_t + P_t$;
for $i = 0 \rightarrow |R_t|$ **do**
    individual $c = R_{t_i}$, randomly pick an individual $d \in R_t[i, R]$
    if c belongs to a better frontier than d **then**
        $Par_t.Add(c)$;
    else if d belongs to a better frontier than c **then**
        $Par_t.Add(d)$;
    end if
    if c and d belong to the same frontier **then**
        if $cd_c \leq cd_d$ **then**
            $Par_t.Add(c)$;
        else if $Par_t.Add(d)$ **then**
            $Par_t.Add(c)$;
        end if
    end if
end for
Randomly select 2 individuals from $Par_t \rightarrow$ Parent 1 and Parent 2

We use crowding distance metric to compare solutions when selecting the offspring and forming a new population. This quantity is an estimate of the density of solutions surrounding a particular solution. Details can be found in Algorithm 9.

Algorithm 9 Calculate crowding distance (Frontier F)

f_m^l denotes the m^{th} objective value of solution l.
$fmax_m$, $fmin_m$ denote the maximum and minimum m^{th} objective values of all solutions in F.
For each solution $i \in F$, initialize $cd_i = 0$.
for Objective $m \in Obj$ **do**
    Sort the set in increasing order of f_m
    $cd_1 = cd_{|F|} = \infty$
    for Solution $l = 2 \rightarrow |F| - 1$ **do**
        $cd_l = cd_l + \frac{f_m^{l+1} - f_m^{l-1}}{fmax_m - fmin_m}$
    end for
end for

CHAPTER 5

Holistic Heuristic Approach

Decomposing a problem into stages transforms a complex problem into simpler subproblems, however, the solution of the decomposition may be different from a holistic approach. Previous approach employs many stages in which approximations and heuristic procedures have been employed thus one may wonder the effect of these procedures both on CPU time and performance quality. In order to make such a comparison, we develop another GA based algoritm not using any decomposition (macro-modes) but instead progressing on total activity list while generating schedules. Figure 5.1 illustrates an example network by combining 3 projects. To make the comparison significant, the main intuition behind chromosome evaluation is similar but contains minor differences as a result of the different chromosome representations of two algorithms. On the other hand, this approach as well employs a multi-objective setting. Population management and parent selection steps are exactly the same as in the decomposition approach.

Figure 5.2 presents an example of a schedule generated by holistic approach. With a single renewable resource constraint, 3 projects composed of different number of activities (5, 3, 2) are combined into single project network with 10 activities preserving inter-project precedence relations. Hence, activities of a project may not necessarily be processed consecutively. Note that mode durations are still uncertain. Hence, in Figure 5.2 the finishing times of activities show the expected finishing times and not the realized finishing times.

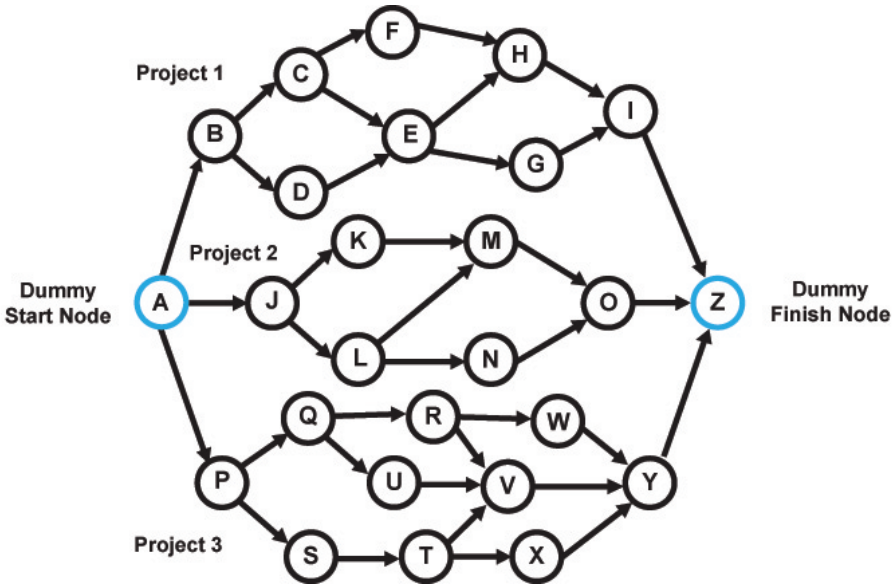

Figure 5.1: Holistic approach network structure composed of 3 projects

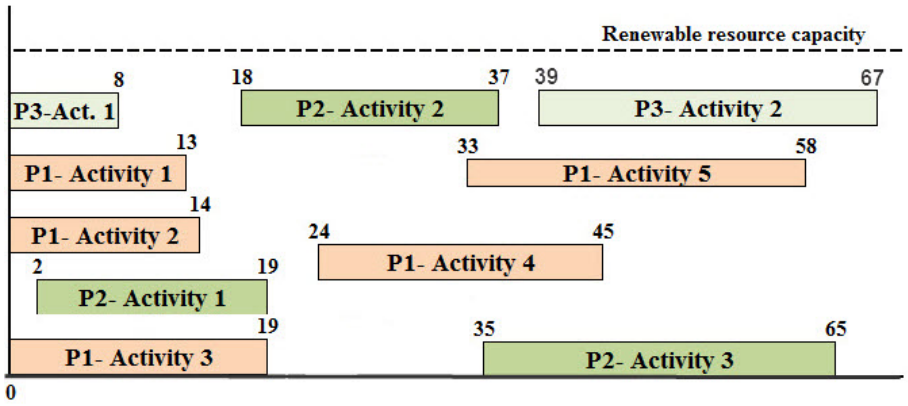

Figure 5.2: Gantt chart of a sample schedule generated as an output of the holistic approach

5.1 Chromosome representation

In decomposition approach, chromosomes are composed of project sequence list and selected macro-mode list. Since we do not have macro-modes in this approach, we employ a different encoding. A chromosome is composed of an activity sequence list and a corresponding mode list. Although projects are independent from one another, there are precedence relations between activities of each project. So, for each chromosome a precedence check must be performed, which increases the computational burden. Non-renewable resource constraints are again satisfied for each

chromosome as in the decomposition case. Note that, in this approach chromosomes are much longer because of a much larger network structure compared to decomposition approach.

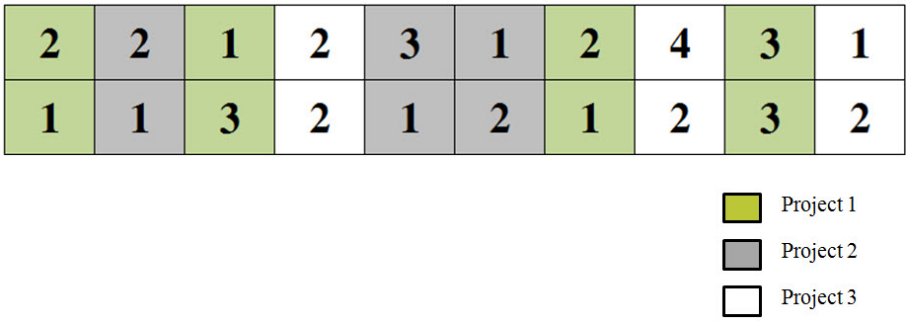

Figure 5.3: Chromosome representation

An example for a schedule representation is given in Figure 5.3 with 3 projects where one project consists of 4 activities and the other two projects consists of 3 activities (in total of 10 activities). Top row of the chromosome represents again the priority sequence for the activities and bottom row of the chromosome represents the list of mode selections for the corresponding activities. Note that the activities of a project do not need to appear consecutively.

5.2 Evaluation of chromosomes

The fitness value for a chromosome is again calculated by a heuristic similar to the one in decomposition approach. The heuristic, shown in Algorithm 10, is composed of two stages where in the first stage a target makespan is determined and in the second stage minimum TSGA solution robustness objective is calculated in order to compute robust starting times of activities.

5.2.1 Stage 1 : Target makespan computation

In stage 1, we employ a serial scheduling procedure by generating durations of activity modes by simulation. For each simulation step, the duration of selected mode of project sequence list is randomly generated. Afterwards, all activities in

Algorithm 10 Holistic Approach Chromosome Evaluation

```
Given chromosome c, composed of activity list A and selected mode list M
for K simulations do
    Randomly generate durations for each mode m ∈ M of a ∈ A
    for all Activity a ∈ A do
        Schedule a to its earliest sequence and precence feasible starting time
    end for
    Generate schedule and store its makespan value in array_makespan
end for
Determine Target Makespan from array_makespan sorted in increasing order
Solve Minimum TSAD Model and compute robust starting times
```

the sequence list are serially scheduled employing their mode selections in the mode list.

In each simulation step, activities are scheduled to their earliest precedence and sequence feasible starting time. Thus for K simulation steps we obtain K schedules each having its own makespan and starting time values for all activities. $array_{makespan}$ denotes the array of makespan values in each simulation step. After sorting $array_{makespan}$ in increasing order, selected target makespan value is set as the n^{th} value of $array_{makespan}$, where $n = RoundUp(ProbabilityLimit * K)$.

5.2.2 Stage 2 : TSAD minimization model

The solution robustness objective model in the holistic approach is the same with the one presented in Section 4.3.3.4 with the only difference being that here a single model containing all activities of all projects is solved. Hence, the objective value of TSAD model becomes the second objective of the chromosome.

5.3 Crossover

Presence of precedence constaints force the employement of precedence check procedures for any child generated from selected two parents. Checking precedence feasibility is computationally expensive. Hence a procedure is needed which generates very diverse child chromosomes and preserves precedence feasibility throughout the generation of the new chromosome as well. For that reason we make use of

a form of uniform crossover procedure [34], which applies crossover to the parents chromosome's gene by gene with a random choice ratio of 0.5.

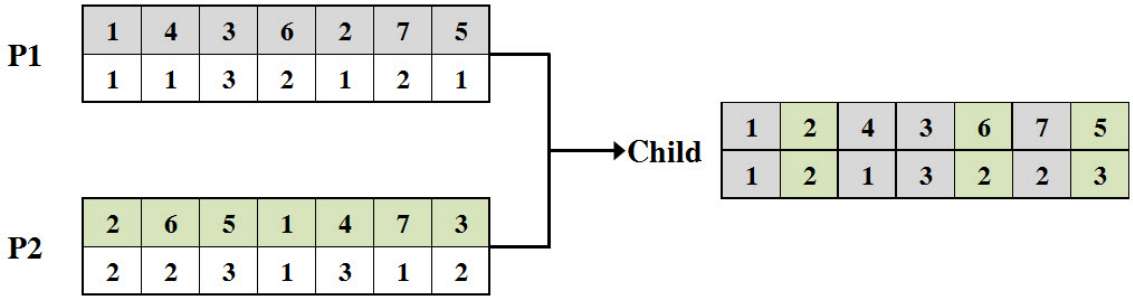

Figure 5.4: Uniform crossover

Note that since the first and second parents are precedence feasible, the resulting child chromosome is precedence feasible. Thus no additional precedence check procedure is necessary. However, non-renewable resource constaints must be checked for violation with each newly generated off-spring.

5.4 Mutation

Employed mutation procedures are the same with the procedures in the decomposition approach, however, after each mutation there is an additional precedence feasibility condition imposed. The details of swap and bit mutation can be found in section 4.3.5.

5.5 Population management

Considering the use of the same MOGA by Deb et al. [29] initial population generation, parent selection (crowded tournament selection) and crowded distance assignment procedure are the same as in the decomposition procedure. The details of these procedures can be found in section 4.3.6.

CHAPTER 6

Computational Studies

Two solution approaches described in the previous two chapters are executed in environments in which various factors affect the behaviour of solution procedures significantly. Project size, resource based factors, probability limit and the range of uncertainty in (mode) durations are common factors affecting both decomposition and holistic solution procedures. Selected cluster number for grouping macro-mode realizations affects solely decomposition procedure. Also, since both solution approaches are MOGA's, GA parameter selection(population size, generation number, mutation probabilities and newborn ratio) is also an important step while evaluating their performance. A series of computational experiments are carried out in order to observe the affects of these parameters and factors on the solution quality and required computational effort.

6.1 Data

Data sets generated for MRCMPSP [1] are adapted to the current problem by adding additional parameters. These data sets include a various combinations of single project instances with different activity sizes developed by Kolisch and Sprecher [35]. Probability limits and duration ranges are added in these datasets.

6.1.1 Resource conditions

Resource factor (RF_τ) and resource strength (RS_τ), which were defined to represent the resource based conditions of resource categories $\tau \in \{\mathcal{R}, \mathcal{N}\}$ and shown to exercise (Kolisch et al. [35]) a strong effect on the behavior of RCPSP solution pro-

cedures, are adapted here for multi-project scheduling environment. RF_τ measures the usage and consumption of resource type τ and RS_τ measures the strength of resource availabilities of resource type τ.

6.1.1.1 Resource factor

Resource factor of resource $r \in R$ (or $n \in N$), reflects the average proportion of the resource r (n) used and consumed [35]. $RF_{\mathcal{R}}$ is given by (6.1) and (6.3); and $RF_{\mathcal{N}}$ is given by (6.2) and (6.4).

$$y_{ijr} = 1 \text{ if } w_{ijr} > 0; \quad 0 \text{ otherwise} \tag{6.1}$$

$$z_{ijn} = 1 \text{ if } q_{ijn} > 0; \quad 0 \text{ otherwise} \tag{6.2}$$

$$RF_{\mathcal{R}} = \frac{1}{|\mathcal{R}|}\frac{1}{|S|-2}\sum_{s=2}^{|S|-1}\frac{1}{|V_s|-2}\sum_{i=2}^{|V_s|-1}\frac{1}{|M_i|}\sum_{j\in M_i}\sum_{r\in\mathcal{R}} y_{ijr} \tag{6.3}$$

$$RF_{\mathcal{N}} = \frac{1}{|\mathcal{N}|}\frac{1}{|S|-2}\sum_{s=2}^{|S|-1}\frac{1}{|V_s|-2}\sum_{i=2}^{|V_s|-1}\frac{1}{|M_i|}\sum_{j\in M_i}\sum_{n\in\mathcal{N}} z_{ijn} \tag{6.4}$$

6.1.1.2 Resource strength

As described by Kolisch and Sprecher [36], $RS_\tau \in \{0,1\}$ is a scaling parameter expressing the resource availability as a convex combination of a minimum and maximum level. Minimum and maximum levels for each non-renewable resource $n \in \mathcal{N}$ are expressed by K_n^{min} as in (6.5) and K_n^{max} as in (6.6), respectively.

$$K_n^{min} = \sum_{i\in V}\min_{j\in M_i}\{q_{ijn}\} \tag{6.5}$$

$$K_n^{max} = \sum_{i\in V}\max_{j\in M_i}\{q_{ijn}\} \tag{6.6}$$

Minimum and maximum levels for each renewable resource $r \in \mathcal{R}$ are expressed by K_r^{min} as in (6.7) and K_r^{max} is determined by the peak per period usage of renewable resource r required in the earliest start schedule obtained by selecting project modes

with the greatest requirements for renewable resource r.

$$K_r^{min} = \max_{i \in V} \left\{ \min_{j \in M_i} \{w_{ijr}\} \right\} \tag{6.7}$$

Employing the maximum and minimum levels, resource availabilities for renewable and non-renewable resources are determined as in (6.8) and (6.9), respectively.

$$K_n^{\tau} = K_n^{min} + round(RS_{\tau}(K_n^{max} - K_n^{min})) \tag{6.8}$$

$$K_r^{\tau} = K_r^{min} + round(RS_{\tau}(K_r^{max} - K_r^{min})) \tag{6.9}$$

6.1.2 Problem sets

Four problem sets (A,B,C,D) represent a variety of different environmental factors.

- Problem set A is formed to analyze the effect of resource based factors by fixing other factors. It includes multi-project instances all having the same number of projects and activities but different resource requirements and resource availability levels, categorized by RS and RF values for renewable and non-renewable resources. Each instance includes 10 projects consisting of 14 activities each as shown in the first two columns of Table 6.1. Two levels for RF_R, RF_N, RS_R and three levels RS_N are selected as given in the next three columns of Table 6.1. To avoid any infeasibilities due to insufficient non-renewable resources, a minimum value for resource strength factor of non-renewable resources, RS_N^{min}, is determined by simple testing and a medium level is also calculated by $RS_N^{mid} = RS_N^{min} + (1 - RS_N^{min})/2$. Also the effect of probability limit in computation of resource contraint and target makespan is taken into account by 3 different probability limit values in the last column. Combinations of these five variable factors with different levels results in problem set B with 72 instances in total.

noProj	noAct	RF_R	RF_N	RS_R	RS_N	$ProbLimit$
14	10	$\{0.75, 1\}$	$\{0.75, 1\}$	$\{0.6, 0.9\}$	$\{RS_N^{min}, RS_N^{mid}, 1\}$	$\{0.7, 0.9, 0.96\}$

Table 6.1: Problem set A

- Problem set B focuses on the effects of different number of projects and activities. In these multi-project instances, three levels are set for number of projects and four levels are set for number of activities per project as provided in the first two columns of Table 6.2. RF values for renewable and non-renewable resources are fixed to be 0.5 as shown in the third and fourth columns of Table 6.2. Two levels are determined for RS_N values as shown in the third column of Table 6.2. Levels for RS_N values are set using $RS_N^{mid1} = RS_N^{min} + (1 - RS_N^{min})/3$ and $RS_N^{mid2} = RS_N^{min} + 2 * (1 - RS_N^{min})/3$ as in [1]. The probability limit is fixed to 0.96. Problem set B includes in total 24 instances.

noProj	noAct	RF_R	RF_N	RS_R	RS_N	$ProbLimit$
$\{10, 15, 20\}$	$\{10, 14, 20, 30\}$	0.5	0.5	$\{0.7\}$	$\{RS_N^{mid1}, RS_N^{mid2}\}$	$\{0.96\}$

Table 6.2: Problem set B

- Problem set C is formed to analyze the effect of duration ranges. Similar to problem set A, instances having the same number of projects consisting of the same number of activities, are compared with different mode duration ranges categorized by d_{ij}^{min} and d_{ij}^{max} values. Each instance includes 14 projects consisting of 10 activities each as shown in the first two columns of Table 6.3. Three levels are selected for duration ranges $[d_{ij}^{min}, d_{ij}^{max}]$ are as given in the last four columns of Table 6.3. Problem set C includes 32 instances.

noProj	noAct	RF_R	RF_N	RS_R	RS_N	d_{ij}^{max} - d_{ij}^{min}	$ProbLimit$
14	10	$\{0.5, 0.75\}$	$\{0.75, 1\}$	$\{0.6\}$	$\{1\}$	$\{2, 5, 10, 20\}$	$\{0.7, 0.96\}$

Table 6.3: Problem set C

- Problem set D is formed to analyze the effect of various cluster numbers as an input of the decomposition procedure. Multi-project instances having the same number of projects consisting of the same number of activities, are compared with different cluster numbers as an input of macro-mode genetation. Each instance includes 14 projects consisting of 10 activities each as shown in the first two columns of Table 6.3. Three levels for RF_R, RF_N, RS_R are selected. Four different number of cluster settings are compared. The last value in the last column of Table 6.4, represents the case without any clustering procedure employed. A total of 84 instances are generated.

noProj	noAct	RF_R	RF_N	RS_R	$noCluster$
14	10	$\{0.5, 0.75, 1\}$	$\{0.5, 0.75, 1\}$	$\{0.3, 0.6, 0.9\}$	$\{5, 10, 20, None\}$

Table 6.4: Problem set D

6.2 Software and hardware information

All codes were written in GNU C# and the MIP solver in CPLEX 12.1. All experiments were performed on a HP Compaq dx 7400 Microtower with a 2.33 GHz Intel Core 2 Quad CPU Q8200 processor and 3.46 GB of RAM.

6.3 Measuring the performance of MOGA's

The presence of multi-objectives in the model complicates the evaluation of an algorithm's performance. Some performance metrics include setting a utility weight for each objective and compute a weighted sum of objectives thereby transforming multi-objective formulation into a single objective one [37].

This approach does not fit to our setting since we don't have information on preference weights a priori. There are also approaches where non-dominated final frontier solutions are compared to the pareto-optimal frontier which corresponds to complete set of non-dominated solutions which are not dominated by any other solution in the solution space. This does not fit into our problem neither since we don't know the pareto-optimal frontier.

For these reasons we propose to compare the non-dominated solutions of decomposition to the non-dominated solutions of holistic approach. After obtaining final non-dominated frontier for both H-MOGA and D-MOGA, we combine them into a combined final non-dominated frontier and eliminate dominated solutions. Solutions belonging to decomposition and holistic approaches in the combined final non-dominated frontier are counted and reported. Figure 6.1 demonstrates described idea.

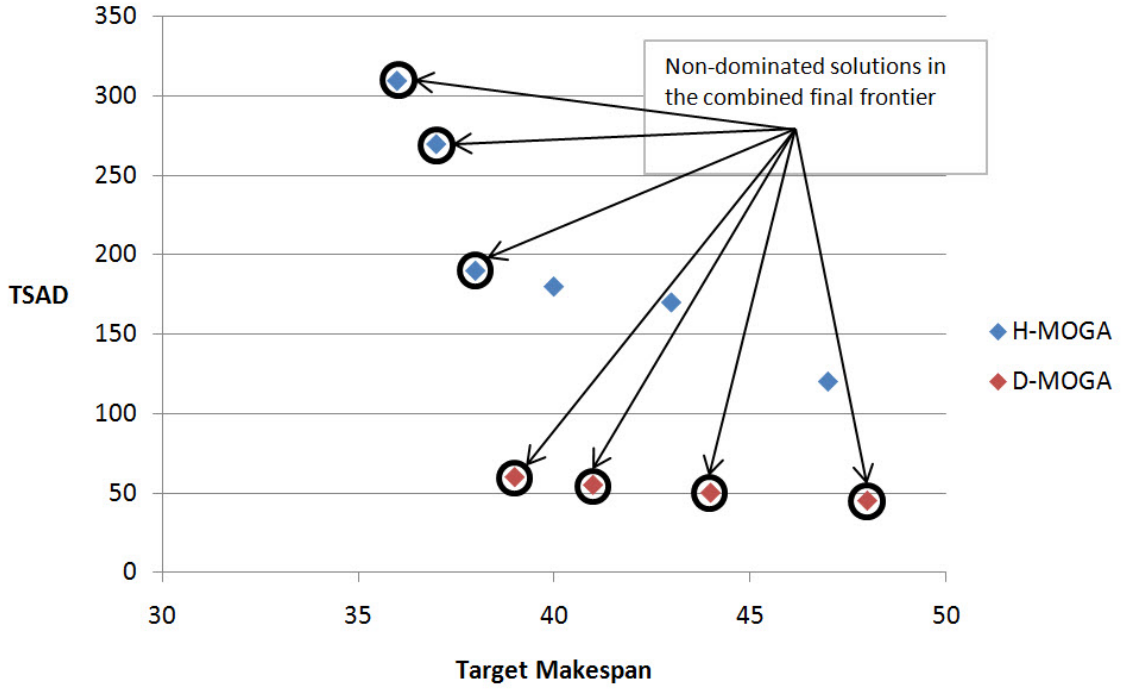

Figure 6.1: Example - combined final frontier solutions

In order to measure the solution quality we propose to use the following measure:

- *Ratio of solutions in the final combined frontier* belonging to D-MOGA and to H-MOGA to the total number of solutions in the combined final frontier, respectively.

In Figure 6.1, 4 of the 7 solutions in the combined final frontier belong to D-MOGA and 3 solutions belong to H-MOGA. Thus solution quality is determined as 4/7 for D-MOGA and 3/7 for H-MOGA.

As additional measures of performance, we propose to use the following metrics:

- *Domination ratio* is the ratio of the number of instances where an approach dominates another to the number of all instances.

- *Disjoint solution region ratio* (DSRR) is a metric that measures whether solutions in the final frontiers of each approach belong to disjoint solution regions. D-Moga and H-Moga solutions in Figure 6.2 are separated both in target makespan and *TSAD* dimensions, thus they belong to disjoint solution regions. Hence DSRR, is the ratio of the number of instances where the solutions of two approaches are not separated in makespan and *TSAD* dimensions, to the number of all instances.

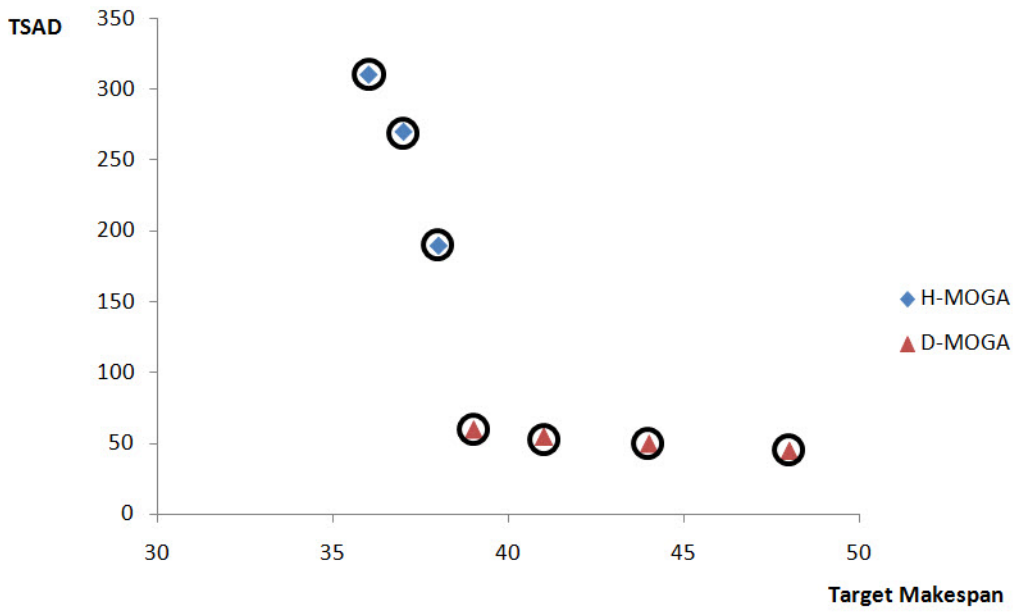

Figure 6.2: Example - disjoint final frontier regions

We also measure computation time performance of solution approaches denoted for D-MOGA as CPU_{D-MOGA} and for H-MOGA as CPU_{H-MOGA}.

6.4 MOGA parametric analysis

Performance of a GA depends heavily on selected parameters: population size, generation number, newborn ratio, swap and bit mutation probability. In order to measure the performance of D-MOGA and H-MOGA approaches in described data sets A, B and C, we first perform experiments in order to catch the behaviour of two algoritms under different parameter settings. Hence, we randomly selected 6 instances from generated data sets and obtained non-dominated frontier solutions of each approach individually under different population number and number of generations. Under each setting, swap mutation and bit mutation probability are set to 0.3.

Setting no	Population size	Number of generations
1	20	20
2	100	20
3	20	100
4	100	100
5	100	200
6	200	100
7	200	200

Table 6.5: MOGA parameter selection analysis

We present below the progression of final frontier of a sample problem instance. The instance is selected from data set A, and is composed of 14 projects each having 10 activities. Both D-MOGA and H-MOGA approaches are analyzed individually. Figure 6.3 presents the final non-dominated frontier for D-MOGA, and final non-dominated frontier H-MOGA is presented in Figure 6.4.

Note that for the sample problem instance, the final non-dominated frontier of H-MOGA approach require higher population size and generation number compared to D-MOGA. This result is due to the fact that H-MOGA approach has a much larger chromosome length and thus requires more computation in order to obtain rather stable final-frontiers. This result is repeated among all randomly selected instances.

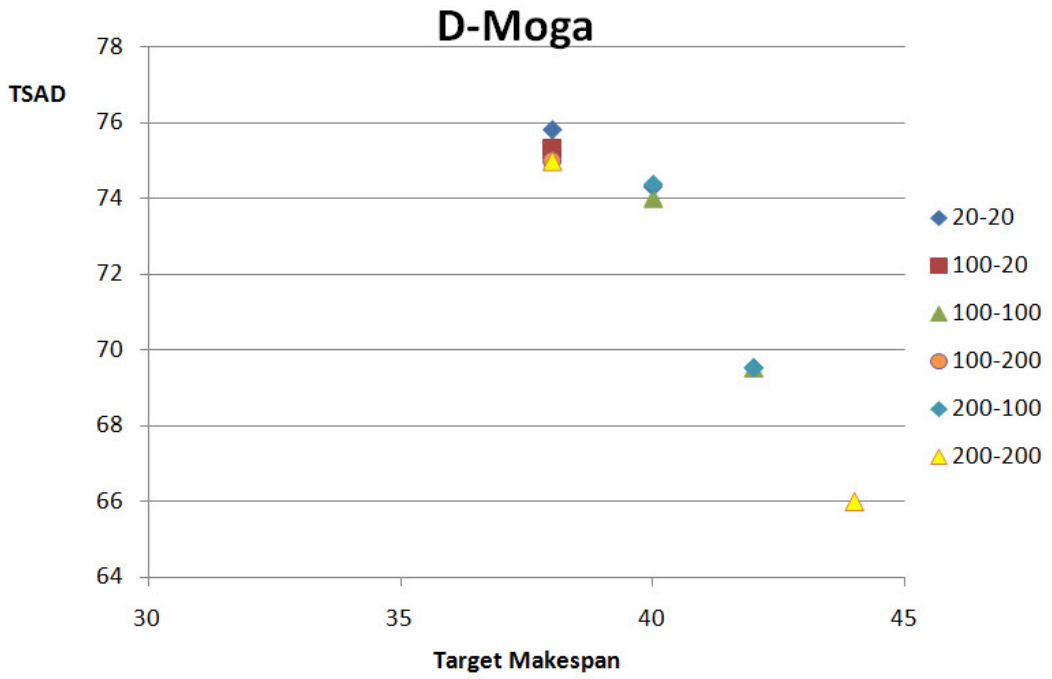

Figure 6.3: Example - progression of non-domimated frontier under different parameter settings - D-MOGA

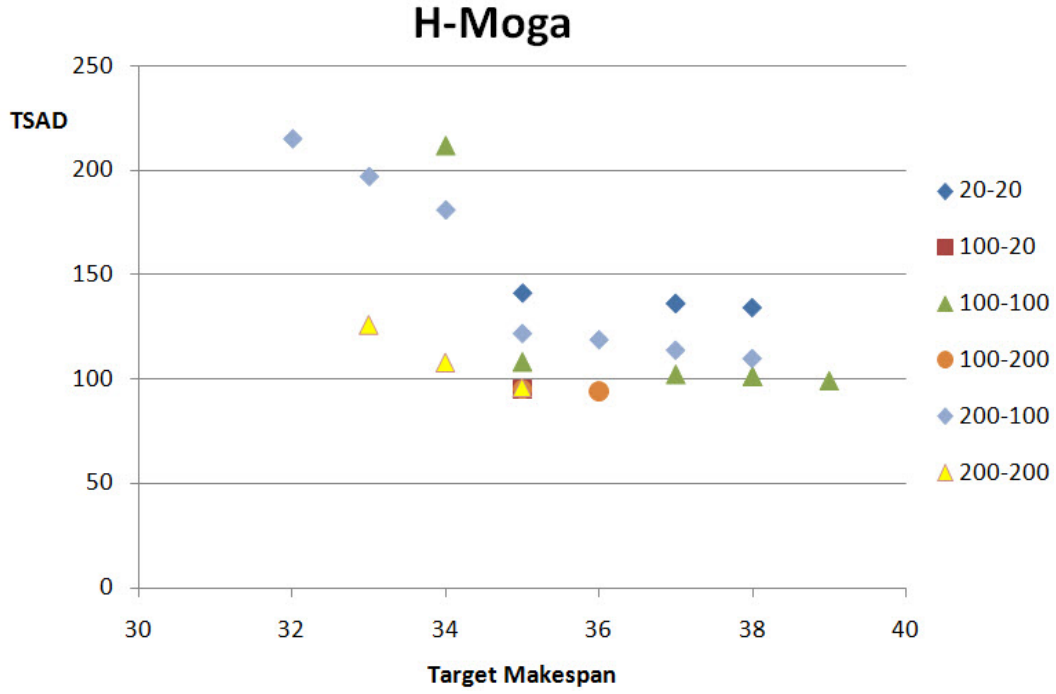

Figure 6.4: Example - progression of non-domimated frontier under different parameter settings - H-MOGA

Although the results of the parametric analysis are in favor of bigger population sizes and higher number of generations, these parameter selections are very limiting with respect to time performance. For the same example instance, Figure 6.5 shows the required CPU time for each approach under different parameter settings.

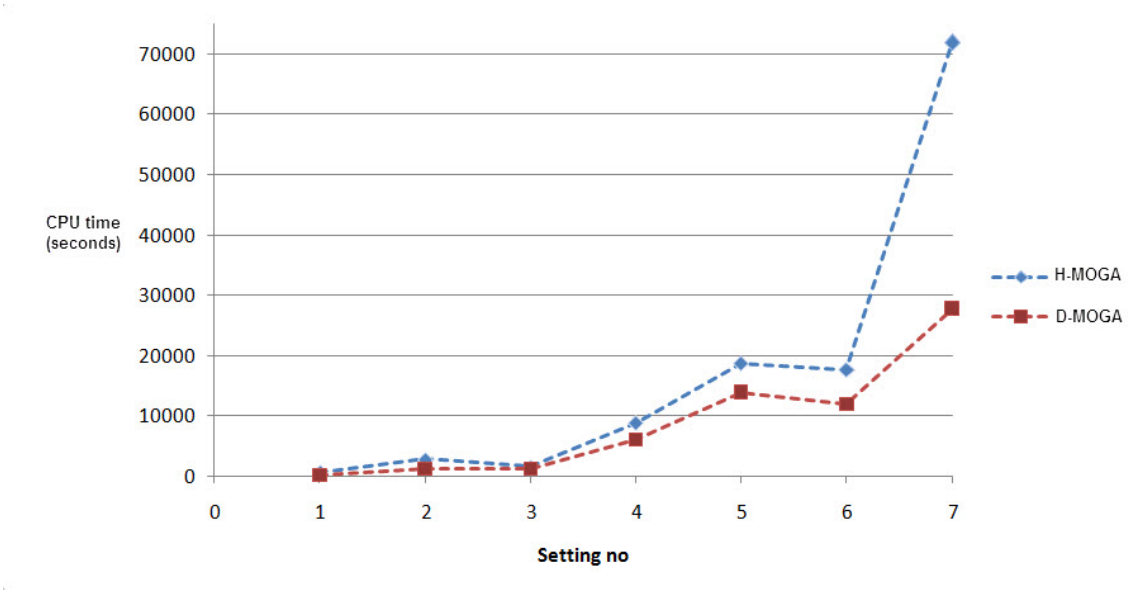

Figure 6.5: Example - required CPU time under different parameter settings

Note that some of these results are obtained despite their violation the pregiven CPU time limit (3 hours). For example, H-MOGA requires approximately 20 hours under parameter setting 7. Anticipating that the CPU times would increase with higher number of projects and activities we decided to set MOGA parameters population count, generation number, newborn ratio, swap mutation and bit mutation probability respectively as 20, 20, 0.5, 0.4, 0.4 for data set B, C and D. For data set A, all results are obtained easily within the runtime limit. Thus, we decided to increase the population count to 50 in order to obtain better results.

6.5 Experimental studies

Data sets A,B and C are solved for both D-MOGA and H-MOGA. The ratio of D-MOGA solutions compared to H-MOGA solutions in the combined non-dominated frontier, domination ratio and DSRR values are presented. Table 6.6 shows ratio of

non-dominated solutions of the two approaches for each data set.

	Ratio of solutions in the final combined frontier					
	Problem Set A		Problem Set B		Problem Set C	
	Average	Stdev	Average	Stdev	Average	Stdev
D-MOGA	0.705	0.390	0.670	0.170	0.872	0.127
H-MOGA	0.294		0.330		0.235	

Table 6.6: Ratio of solutions in the final combined frontier for data set A, B and C

The overall results for each data set show that a higher ratio of solutions in the final combined frontier are coming from D-MOGA. Table 6.7 shows that in most of the instances in data set A and C, solutions of D-MOGA fully dominate the solutions generated by H-MOGA. However, for data set B, we do not see a similar result and this fact could be due to the separation of solution regions along with the increase in the number of projects and activities. Note that the solution regions in data set B are fully disjoint. In Table 6.7 domination ratio is solely given for solutions where D-MOGA fully dominates H-MOGA, because there is not a single instance where solutions of H-MOGA fully dominate solutions of D-MOGA in all solved instances.

	Domination Ratio (D-MOGA)	DSRR
Data set A	0.708	0.445
Data set B	0.173	1
Data set C	0.593	0.25

Table 6.7: Additional comparison measures for datasets A, B and C

Table 6.8 shows required CPU times for the two approaches in data sets A, B and C. Overall results show that D-MOGA requires much less time compared to H-MOGA. This is expected since the chromosome length is much larger in H-MOGA and precedence feasibility checks are additionaly employed.

	CPU time (sec)								
	Problem Set A			Problem Set B			Problem Set C		
	Min	Average	Max	Min	Average	Max	Min	Average	Max
D-MOGA	374.2	679.3	1260.6	333.5	712.1	1451.6	145.1	259.4	501.5
H-MOGA	1212.3	1354.5	1564.2	1417.2	6053.9	20211.5	286.5	507.3	897.8

Table 6.8: CPU times for data sets A, B and C

In the following subsections we present general observations obtained for each data set in their own setting.

6.5.1 Resource and probability limit analysis

In order to measure the effect of resource based factors (RF_R, RF_N, RS_R and RS_N) and probability limit values problem set A is solved and analyzed. Results in Table 6.9 point out that an increase in (RS_R) results in an increase of D-MOGA solutions in the ratio of solutions in the final combined frontier. Hence, this observation shows that when the multi-project problem has more renewable resources the overestimation of resource usage (4.3.3.1) is compensated for.

	Ratio of solutions in the final combined frontier			
	D-MOGA		H-MOGA	
Prob. Limit	$RS_R = 2$	$RS_R = 3$	$RS_R = 2$	$RS_R = 3$
0.7	0.765	0.787	0.234	0.212
0.9	0.635	0.638	0.364	0.361
0.96	0.665	0.743	0.334	0.257

Table 6.9: Effect of RS_R on ratio of solutions in the final combined frontier for data set A

Note that with decreasing RS_R as the abundance of renewable resources decreases in the network, the ratio of H-MOGA solutions in the final combined frontier increases. Hence, in tight resource conditions a mixed approach employing both D-MOGA and H-MOGA simultaneously, could be developed as a future research direction.

The effect of RS_R values on the required computational effort is also interesting. As Table 6.10 shows, CPU_{D-MOGA} increases with an increase in RS_R whereas CPU_{H-MOGA} decreases. For D-MOGA it can be argued that with the more renewable resource availabilities RSL_{rt} lists defined in section 4.3.3.2 will be longer which would make the computation of probability constraints harder, thus requiring more computational effort. Whereas for H-MOGA as the renewable resource tightness decreases, the renewable resource-based infeasibilities in Model S_s (Section 4.3.3.4) are prevented. It should be noted that in case a feasible solution is not found in

Model S_s, the model is re-run with an updated *target makespan* thereby requiring additional computational effort.

	Average CPU (sec)			
	D-MOGA		H-MOGA	
Prob. Limit	$RS_R = 2$	$RS_R = 3$	$RS_R = 2$	$RS_R = 3$
0.7	669.3	702.3	1356.6	1347.2
0.9	623.6	708.5	1364.5	1346.5
0.96	673.2	712.7	1370.4	1338.6

Table 6.10: Effect of RS_R on average CPU for data set A

6.5.2 Effect of number of projects and activities

The effect of number of projects and activities is analyzed in problem set B. Results of the instances having different number of projects (Table 6.11) that as the project number increases in the multi-project network, required computational time sharply increases in H-MOGA. This fact coincides with the expectation that the number of projects in the problem environment has a significant impact on the problem difficulty. However, for D-MOGA approach, higher number of projects may not result in higher CPU times and this fact could be due to joint probability calculations in formulation. In some cases, infeasibilities of probabilistic constaints may be easily computed by checking maximum renewable resource usage levels of simultaneously resource sharing projects thereby accelerating computation of the overall algorithm.

	CPU time (sec)			
	D-MOGA		H-MOGA	
noProj	Average	Stdev	Average	Stdev
10	433.2	137.8	4055.2	2071.2
15	814.8	254.1	6032.3	2456.2
20	802.375	396.1	8076.8	6563.3

Table 6.11: Effect of number of projects on CPU time for data set B

Table 6.12 presents the average CPU_{D-MOGA} and CPU_{H-MOGA} required to solve the instances from problem set B and having different number of activities. In Table 6.12 we observe that D-MOGA's CPU time requirement increases along with number of activities. This fact is expected, since an increase in number of activities results

in an increase of variable numbers in Model S_s for each project. Although a similar result is expected for H-MOGA, such a conclusion cannot be reached considering the resulting time values. Although instances composed of 20 activities required less computational time than instances composed of 14 activities, a sharp increase in standart deviations is observed. Hence these average values can be deceptive to reach a decisive conclusion.

	CPU time			
	D-MOGA		H-MOGA	
noAct	Average	Stdev	Average	Stdev
10	539.2	219.2	2690.6	842.8
14	588.8	229.3	6883.4	590.4
20	595.33	240.7	4983.5	3123.2
30	1065.16	330.8	10907.5	5732.5

Table 6.12: Effect of number of activities on CPU time for data set B

6.5.3 Duration bound analysis

We assume that duration of activitiy i in its mode j is a triangular distributed random variable with pregiven lower and upper bounds, $d_{ij}^{min}, d_{ij}^{max}$. Duration range between d_{ij}^{min} and d_{ij}^{max} is an important parameter effecting the uncertainty of the model. Hence, data set C includes three different range settings: 2, 5 and 10. Expected durations kept constant in each instance, lower and upper duration bounds are adjusted. Table 6.13 shows that both CPU_{D-MOGA} and CPU_{H-MOGA} increases along with an increase of duration range in the instances. This fact is related to the increase of the range between earliest and latest starting times obtained by simulation. As the mode duration range increases, earliest starting times of activities are pushed backward and latest starting times are pushed forward. Thereby, the solution space of $TSAD$ models increases thus result in higher computational times.

Range	Average CPU time	
	D-MOGA	H-MOGA
2	180.1	317.2
5	189.2	378.1
10	241.3	506.6
20	426.3	825.7

Table 6.13: Effect of duration bound on CPU time for data set C

6.5.4 Decomposition clustering analysis

Clustering analysis described in Section 4.1.4 aims to pick the most representative K realizations among many realizations generations generated for a macro-mode. Each realization symbolizes a point in the discrete probability distribution of a macro-mode and joint probability constraints (4.29) and (4.30) are making use of this probability distribution in evaluation. However, each realization has a usage level for each resource and a makespan by resource profile transformation discussed in Section 4.3.3.1. On the other hand, probabilities in (4.29) and (4.30) are computed by conditioning on each variable and the more the number of points the harder the computation. This fact motivated us to perform a clustering analysis and decide on the pregiven number of clusters for K-means clustering algorithm.

In order to catch the behaviour of D-MOGA under different number of clusters, four number of cluster levels (5, 10, 20, None) are included in data set D. Table 6.14 shows the effect of selected number of clusters on the ratio of solutions in the final combined frontier.

Number of clusters	Ratio of solutions in the final combined frontier
5	0.236
10	0.273
20	0.317
None	0.197

Table 6.14: Effect of number of clusters on the ratio of solutions in the final combined frontier

In 23 of the 84 instances solved, decompositon MOGA without employing any clustering procedure could not generate solutions within the pregiven time limit (1 hour for initial population generation). For that reason, Table 6.15 presents revised

results in which those instances are not included. Note that when time limit is exceeded for any instance, its non-dominated final frontier does not include any solutions thereby gets dominated by other procedures with pre-set cluster numbers.

	Number of Clusters			
NoProj	5	10	20	None
10	0.271	0.208	0.238	0.281
20	0.203	0.182	0.509	0.107
(10+20)	0.273	0.153	0.296	0.276

Table 6.15: Effect of number of clusters on the ratio of solutions in the final combined frontier - revised results

Along with general solution performance findings, in Table 6.15 also an interesting finding appears in project-wise comparison. Note that although 10 number of clusters dominates 5 number of clusters in all instances, this result is not obtained after eliminating instances where time limit is exceeded for no-clustering case. Thereby, as the results suggests a different number of cluster can be set depending on network conditions as there seems to exist different highly performing number of clusters.

Table 6.16 shows the effect of number of clusters on CPU time for the given instances. In time performance results indicate the benefit of using a clustering procedure, however, employing less number of clusters does not necessarily result in lower CPU time requirements for all instances. One of the reasons this has occured in 20 project instances is that it is possible to obtain different schedules with different RSL_{rt} and RFL sequences. This difference may lead to increase or decrease in the required computational effort of computing joint probabilistic constraints. In addition to solution performance, CPU time results also show that different number of clusters may be more suitable depending on the instance under consideration.

	Number of Clusters			
NoProj	5	10	20	None
10	451.7	485.3	539.1	726.2
20	2962.8	3645.4	3235.2	4628.5

Table 6.16: Effect of number of clusters on CPU time

CHAPTER 7

Conclusions and Future Work

In this study, we investigate MRCPMSP under uncertainty. The problem has a multi-objective setting related to quality and solution robustness. First objective is obtaining a minimum multi-project duration target from which the realized schedule is not to exceed with a preset probability. Second objective is the $TSAD$ of scheduled starting times of activities from their earliest starting times generated by simulation. We develop two MOGA solution approaches, first one decomposing the problem into 2-stages and the other approaching the problem in a holistic fashion.

Initial step of D-MOGA includes a two-phase decomposition where each project is reduced to a single macro-activity by systematical using of artificial budget values and expected project durations. Generated macro-activities may have one or more processing modes (macro-modes). The uncertainty in macro-modes is modeled via simulations where randomly generated activity mode durations result in disrupted macro-activity schedules. These disruptions compose the discrete probability function of macro-activities and representative cases are selected by K-means clustering procedure.

Afterwards, nondominated sorting genetic algoritm (NSGA-II) is executed [38]. For fitness computation a two stage heuristic is developed. In the first stage given the probability constraints, a serial scheduling routine along with a buffering procedure is applied to a project list and a minimum target makespan of overall projects is determined. In the second stage minimum total absolute sum of deviations model is solved in order to find solution robust starting times of activities for each project. The objective value of this model is taken as the second objective of the MOGA.

Using the same multi-objective formulation and population management, another MOGA is developed. H-MOGA combines all activities of each project into one big network and does not require that activities of a project are scheduled consecutively. There is also a similar two-stage heuristic for assigning fitness values. In the first stage, for K simulations given the same activity sequence, mode durations of activities are randomly generated. Activities in their respective modes in the mode list are serially scheduled thereby K schedules are obtained. Resulting from makespan values of these K schedules, a target makespan is computed. In the second stage, a similar $TSAD$ minimization model is solved as in D-MOGA is solved.

Computational studies measuring performance of the two proposed solution approaches are performed for different datasets. When non-dominated solutions of each approach combined in a final population, overall results show that a larger ratio of these solutions are genetared by D-MOGA. Additionaly, required computational effort for D-MOGA is much less than the holistic approach. As the the abundance of renewable resources decreases with decreasing RS_R in the network, the difference between the ratio of solutions in the final combined frontier belonging to two approaches decreases.

As a future research direction fast heuristics could be employed in solving minimum $TSAD$ models which currently compose a major bottleneck on the computational performance of the proposed algorithms. Another direction would be on setting individual project target makespans in addition to multi-project target and on obtaining more solution robust schedules at project level. Furthermore, resource based uncertainties could be incorporated into the mode. Also experiments could be performed with different distribution assumptions. Considering future research possibilities and the voidness of robust scheduling literature in MRCMPSP, the topic has a rich potential of problems and extensions. We hope this first study on MRCMPSP under uncertainty would motivate researchers to this intriguing field.

Bibliography

[1] A. Can, "Multi-project scheduling with 2-stage decomposition," Master's thesis, Sabanci University, Istanbul, 2010.

[2] J. H. Payne, "Management of multiple simultaneous projects: a state-of-the-art review," *International Journal of Project Management*, vol. 13, pp. 163–168, 1995.

[3] M. G. Speranza and C. Vercellis, "Hierarchical models for multi-project planning and scheduling," *European Journal of Operational Research*, vol. 64, pp. 312–325, 1993.

[4] O. Hazr, M. Haouari, and E. Erel, "Robust scheduling and robustness measures for the discrete time/cost trade-off problem," *European Journal of Operational Research*, vol. 207, no. 2, pp. 633–643, Dec. 2010.

[5] E. Klerides and E. Hadjiconstantinou, "A decomposition-based stochastic programming approach for the project scheduling problem under time/cost trade-off settings and uncertain durations," *Computers and Operations Research*, vol. 37, no. 12, pp. 2131–2140, 2010.

[6] G. Zhu, J. F. Bard, and G. Yu, "Disruption management for resource-constrained project scheduling," *Journal of the Operational Research Society*, vol. 56, no. 4, pp. 365–381, Oct. 2004.

[7] F. Deblaere, E. Demeulemeester, and W. Herroelen, "Reactive scheduling in the multi-mode RCPSP," *Computers and Operations Research*, vol. 38, no. 1, pp. 63–74, Jan. 2010.

[8] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.

[9] W. Herroelen and R. Leus, "Project scheduling under uncertainty: Survey and research potentials," *European Journal of Operational Research*, vol. 165, no. 2, pp. 289–306, Sep. 2005.

[10] L. Tavares, J. Antunes Ferreira, and J. Silva Coelho, "On the optimal management of project risk," *European Journal of Operational Research*, vol. 107, no. 2, pp. 451–469, 1998.

[11] R. Leus, "The generation of stable project plans," Ph.D. dissertation, Oct. 2004.

[12] W. Herroelen and R. Leus, "Robust and reactive project scheduling: a review and classification of procedures," *International Journal of Production Research*, vol. 42, no. 8, pp. 1599–1620, Apr. 2004.

[13] S. Van De Vonder, E. Demeulemeester, W. Herroelen, and R. Leus, "The trade-off between stability and makespan in resource-constrained project scheduling," *International Journal of Production Research*, vol. 44, no. 2, pp. 215–236, Jan. 2006.

[14] E. Demeulemeester and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem," *Management science*, 1992.

[15] S. Vonder, E. Demeulemeester, and W. Herroelen, "A classification of predictive-reactive project scheduling procedures," *Journal of Scheduling*, vol. 10, no. 3, pp. 195–207, May 2007.

[16] H. Chtourou and M. Haouari, "A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling," *Computers and Industrial Engineering*, vol. 55, no. 1, pp. 183–194, Aug. 2008.

[17] M. Bruni, P. Beraldi, F. Guerriero, and E. Pinto, "A heuristic approach for resource constrained project scheduling with uncertain activity durations," *Computers and Operations Research*, vol. 38, no. 9, pp. 1305–1318, Sep. 2011.

[18] D. Golenko-Ginzburg and A. Gonik, "Stochastic network project scheduling with non-consumable limited resources," *International Journal of Production Economics*, vol. 48, no. 1, pp. 29–37, 1997.

[19] D. Golenkoginzburg and A. Gonik, "A heuristic for network project scheduling with random activity durations depending on the resource allocation," *International Journal of Production Economics*, vol. 55, no. 2, pp. 149–162, Jul. 1998.

[20] D. Golenko-Ginzburg, A. Gonik, and S. Sitniakovski, "Resource supportability model for stochastic network projects under a chance constraint," *Communications in Dependability and Quality Management*, vol. 3, no. 1, pp. 89–102, 2000.

[21] D. Golenko-Ginzburg, S. Lyubkin, V. Rezer, and S. Sitnyakovskii, "Algorithms of optimal supply of resources to a group of projects (stochastic networks)," *Automation and Remote Control*, vol. 62, no. 8, pp. 1366–1375, 2001.

[22] G. Zhu, J. F. Bard, and G. Yu, "A two-stage stochastic programming approach for project planning with uncertain activity durations," *Journal of Scheduling*, vol. 10, no. 3, pp. 167–180, May 2007.

[23] A. Sprecher, S. Hartmann, and A. Drexl, "An exact algorithm for project scheduling with multiple modes," *OR Spektrum*, vol. 19, pp. 195–203, 1997.

[24] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson Addison Wesley Boston, 2006.

[25] D. Montana, M. Brinn, S. Moore, and G. Bidwell, "Genetic algorithms for complex, real-time scheduling," *SMC'98 Conference Proceedings. 1998 IEEE Inter-*

national Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218), pp. 2213–2218.

[26] L. Davis, "Job shop scheduling with genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 136–140.

[27] S. Kumanan, G. Jegan Jose, and K. Raja, "Multi-project scheduling using an heuristic and a genetic algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 31, no. 3-4, pp. 360–366, May 2006.

[28] Hartmann, "Project scheduling with multiple modes: A genetic algorithm based approach," *Annals of Operations Research*, pp. 1–23, 2001.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[30] R. Kolisch and S. Hartmann, "7 heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis," J. Weglarz, Ed. Springer Netherlands, 1999, p. 147.

[31] V. Van Peteghem and M. Vanhoucke, "A genetic algorithm for the multi-mode resource-constrained project scheduling problem," *Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium*, 2008.

[32] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 127, no. 2, pp. 394–407, 2000.

[33] J. Alcaraz, C. Maroto, and R. Ruiz, "Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms," *Journal of the Operational Research Society*, vol. 54, no. 6, pp. 614–626, 2003.

[34] F. Şerifoğlu, "A new uniform order-based crossover operator for genetic algorithm applications to multi-component combinatorial optimization problems," Ph.D. dissertation, Boğaziçi University, Istanbul, 1997.

[35] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource-constrained project scheduling problems," *Management Science*, vol. 41, pp. 1693–1703, 1995.

[36] R. Kolisch and A. Sprecher, "PSPLIB - a project scheduling problem library," *European Journal of Operational Research*, vol. 96, pp. 205–216, 1996.

[37] C. Grosan, M. Oltean, and D. Dumitrescu, "Performance metrics for multiobjective optimization evolutionary algorithms," in *Proceedings of Conference on Applied and Industrial Mathematics (CAIM), Oradea*, 2003.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.