

Single Machine Scheduling with Timelag Constraints

by

Gülçin Ermiş

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Industrial Engineering and Operations Management

Koç University

November, 2014

Koç University
Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation By

Gülçin Ermiş

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Prof. CEYDA OĞUZ

Assoc. Prof. E. ALPER YILDIRIM

Assoc. Prof. SERDAR TAŞIRAN

Assoc. Prof. SİBEL SALMAN

Assoc. Prof. BÜLENT ÇATAY

Date: _____

to honesty

ABSTRACT

In a single machine problem a set of tasks has to be processed without preemption on a machine in such a way that at most one task is processed at any time and a given objective function is optimized.

This thesis examines the single machine scheduling problems with timelag constraints. In these problems, precedence relations exist among the tasks that are to be scheduled. In a precedence relation, successor task is not allowed to start before the processing of the predecessor task is completed. Precedence relations may be generalized by adding the timelag constraints. In this thesis, timelags are assumed to be minimal timelags. Timelag constraint necessitates delaying the start time of the successor task after the completion time of the preceding task. Precedence constraints between tasks that have to be respected in every feasible schedule generally increase the computational complexity of a scheduling problem. Occasionally, their introduction may turn a problem that is solvable within polynomial time into an NP-complete one (Lenstra and Rinnoy Kan, 1978). Generalizing the problems with precedence constraints by introducing the timelag constraint makes problems harder. In addition to precedence and timelag constraints, the release time constraint is considered in this thesis. Even though there exists a wide variety of results on the complexity of scheduling problems with precedence and timelag constraints, there are still some problems for which the complexity status remains open.

In this thesis, two scheduling problems with open complexity status are proven to be polynomially solvable. Also, a scheduling problem which is the generalization of two strongly NP-hard problems is considered and a branch and bound algorithm is proposed. Furthermore, for each problem, integer programming model is given. Finally, computational results are provided.

ÖZET

Tek makine çizelgeleme probleminde, bir grup iş, ilgili amaç fonksiyonunu en iyileyecek ve eşzamanlı olmayacak şekilde kesintisiz olarak işlenmelidir.

Bu tez erteleme kısıtlı tek makine çizelgeleme problemlerini konu almaktadır. Bu problemlerde, çizelgelenecek işler arasında öncelik ilişkileri bulunmaktadır. Bir öncelik ilişkisinde ardıl iş öncül iş tamamlanmadan işlenmeye başlayamaz. Öncelik ilişkileri, erteleme kısıtları eklenerek genelleştirilebilir. Bu tezde, erteleme sürelerinin, farklı işlerin bitiş ve başlangıçları arasında bırakılması gereken en küçük zaman boşlukları oldukları varsayılmaktadır. Erteleme kısıtı, öncül iş tamamlandıktan sonra ardıl işin başlama zamanını ertelemeyi gerektirir. Genelde, olurlu çözümde uyulması gereken öncelik kısıtları, çizelgeleme probleminin karmaşıklık derecesini arttırmaktadır. Bazen, öncelik kısıtlarının eklenmesi polinom sürede çözülebilir olan bir problemi NP-tam probleme dönüştürebilir (Lenstra and Rinnoy Kan, 1978). Öncelik kısıtlı problemler, erteleme kısıtı eklenerek genelleştirildiklerinde daha zor problemlere dönüşmüş olurlar. Bu tezde, öncelik ve erteleme kısıtlarına ek olarak, işlerin sisteme bırakılma zamanları ile ilgili kısıtlar dikkate alınmaktadır. Geçmiş çalışmalarda öncelik ve erteleme kısıtlı problemlerin karmaşıklıkları konusunda çok sayıda sonuca ulaşılmış olsa da, karmaşıklık durumu bilinmeyen bazı problemler bulunmaktadır.

Bu tezde, karmaşıklık derecesi bilinmeyen iki ayrı çizelgeleme probleminin polinom sürede çözülebilir olduğu gösterilmektedir. Ayrıca, iki farklı güçlü NP-zor problemin genelleştirilmiş şekli olan bir çizelgeleme problemi incelenmekte ve bir dal ve sınır algoritması önerilmektedir. Çalışmada incelenen her problem için ilgili tamsayılı programlama modeli sunulmaktadır. Son olarak, önerilen tüm çözüm yöntemleri için uygulama sonuçları değerlendirilmektedir.

ACKNOWLEDGMENTS

Foremost, I would like to thank my supervisor Professor Ceyda Oğuz. I would also like to express my profound appreciation to my dissertation committee members Professor Serdar Taşiran, Professor Emre Alper Yıldırım, Professor Sibel Salman, Professor Bülent Çatay for their invaluable comments and critical remarks.

I gratefully acknowledge Professor Süleyman Özekici, Professor Serpil Sayın, Professor Lerzan Örmeci, Professor Metin Türkay, Professor Zeynep Akşin, Professor Onur Kaya for their endless support and inspiring questions during my qualifying examination.

It is a pleasure to thank all the professors for the excellent teaching and guidance expanding my knowledge and skills.

I am also grateful to the non-academic staff at Koç University who were always kind and helpful.

I would like to show my gratitude to my former advisor Professor Yusuf İlker Topçu for his positive attitude and supportive communication style which gave me the motivation to start my PhD study.

I would also like to thank TÜBİTAK BİDEB for providing financial assistance.

I expand my thanks to all my officemates for the peaceful environment and all my friends who supported me in any respect during the completion of my thesis.

All things would not be possible without the patience and love of my family.

TABLE OF CONTENTS

List of Tables	ix
List of Figures	xi
Nomenclature	xiii
Chapter 1: Introduction	1
Chapter 2: Literature Review	7
Chapter 3: A Polynomial Algorithm for Scheduling Chains with Time-lags	16
3.1 Problem definition of $1 chains(l < p); p_i = p; r_i C_{max}$	16
3.2 Optimality conditions for $1 chains(l < p); p_i = p; r_i C_{max}$	17
3.3 Complexity of the proposed algorithm for $1 chains(l < p); p_i = p; r_i C_{max}$	57
3.4 Integer programming model for $1 chains(l < p); p_i = p; r_i C_{max}$	60
3.5 Computational results for $1 chains(l < p); p_i = p; r_i C_{max}$	65
Chapter 4: A Polynomial Algorithm for Scheduling Tasks with Out-tree Precedence Graph and Timelags	76
4.1 Problem definition of $1 outtree(l < p); p_i = p; r_i C_{max}$	76
4.2 Optimality conditions for $1 outtree(l < p); p_i = p; r_i C_{max}$	79
4.3 Complexity of the proposed algorithm for $1 outtree(l < p); p_i = p; r_i C_{max}$	101
4.4 Integer programming model for $1 outtree(l < p); p_i = p; r_i C_{max}$	104
4.5 Computational results for $1 outtree(l < p); p_i = p; r_i C_{max}$	109

Chapter 5:	A Branch and Bound Algorithm for Scheduling Chains with Timelags	115
5.1	Problem definition of $1 chains(l); r_i \sum C_i$	115
5.2	Branch and bound algorithm for $1 chains(l); r_i \sum C_i$	116
5.2.1	Branching scheme	117
5.2.2	Partial solutions	118
5.2.3	Lower bounds	120
5.2.4	Upper bounds	122
5.2.5	Exploring the solution space	125
5.3	Integer programming model for $1 chains(l); r_i \sum C_i$	126
5.4	Computational results for the branch and bound and integer programming model for $1 chains(l); r_i \sum C_i$	128
5.5	A warm start integer programming model for $1 chains(l); r_i \sum C_i$	135
5.6	Computational results for warm start integer programming model for $1 chains(l); r_i \sum C_i$	138
5.7	Implications	140
Chapter 6:	Conclusion	145
	Bibliography	148

LIST OF TABLES

3.1	Computational results for $1 chains(l < p); p_i = p; r_i C_{max}$ for $n=100$ and $\tau_1 \in \{0.2, 0.5, 0.8\}$	70
3.2	Computational results for $1 chains(l < p); p_i = p; r_i C_{max}$ for $n=500$ and $\tau_1 \in \{0.2, 0.5, 0.8\}$	71
3.3	Computational results for $1 chains(l < p); p_i = p; r_i C_{max}$ for $n=2500$, $n=12500$ and $\tau_1 \in \{0.2, 0.5, 0.8\}$	72
3.4	Computational results for $1 chains(l < p); p_i = p; r_i C_{max}$ for $n=100$ and $\tau_1 \in \{2, 5, 8\}$	73
3.5	Computational results for $1 chains(l < p); p_i = p; r_i C_{max}$ for $n=500$ and $\tau_1 \in \{2, 5, 8\}$	74
3.6	Computational results for $1 chains(l < p); p_i = p; r_i C_{max}$ for $n=2500$, $n=12500$ and $\tau_1 \in \{2, 5, 8\}$	75
4.1	Computational results for $1 outtree(l < p); p_i = p; r_i C_{max}$	113
5.1	Preliminary tests for branch and bound algorithm for $1 chains(l); p_i; r_i \sum C_i$ for $n=11$ and $K=3$	130
5.2	Preliminary tests for branch and bound algorithm for $1 chains(l); p_i; r_i \sum C_i$ for $n=22$ and $K=4$	130
5.3	Preliminary tests for branch and bound algorithm for $1 chains(l); p_i; r_i \sum C_i$ for $n=20$ and $K=6$	131
5.4	Comparison of computational results for branch and bound and integer programming model for $n=11$ and $K=3$	132

5.5	Comparison of computational results for branch and bound and integer programming model for $n=22$ and $K=4$	132
5.6	Comparison of computational results for branch and bound and integer programming model for $n=20$ and $K=6$	133
5.7	Comparison of computational results for branch and bound and integer programming model for $n=32$ and $K=4$	133
5.8	Comparison of computational results for branch and bound and integer programming model for $n=39$ and $K=5$	134
5.9	Results for warm start integer programming model for $n = 100$	141
5.10	Results for warm start integer programming model for $n = 500$	142
5.11	Results for warm start integer programming model for $n = 1000$	143

LIST OF FIGURES

3.1	Conflicting and supporting examples to Proposition 1 for $1 chains(l < p); p_i = p; r_i C_{max}$	23
3.2	Alternative sequences when there is a violation on the constraint (iv) of CASE 1 - Condition 1 - Subcondition 1.1 for Proposition 1 for $1 chains(l < p); p_i = p; r_i C_{max}$	24
3.3	Conflicting and supporting examples to Proposition 2 for $1 chains(l < p); p_i = p; r_i C_{max}$	28
3.4	Conflicting and supporting examples to Proposition 3 for $1 chains(l < p); p_i = p; r_i C_{max}$	30
3.5	Conflicting and supporting examples to Proposition 4 for $1 chains(l < p); p_i = p; r_i C_{max}$	35
3.6	Conflicting and supporting examples to Proposition 5 for $1 chains(l < p); p_i = p; r_i C_{max}$	37
3.7	Conflicting and supporting examples to LEMMA 10 for $1 chains(l < p); p_i = p; r_i C_{max}$	49
3.8	Example to LEMMA 10 when $2p + l = r_{\beta,1} - r_{\beta,0}$	50
3.9	Network representation for an instance of $1 chains(l < p); p_i = p; r_i C_{max}$	65
3.10	Optimal solution for the instance of $1 chains(l < p); p_i = p; r_i C_{max}$.	66
4.1	Precedence graph and grouping for $1 outtree(l < p); p_i = p; r_i C_{max}$. .	77
4.2	Labeling the groups and tasks for $1 outtree(l < p); p_i = p; r_i C_{max}$. .	78
4.3	Leveling for $1 outtree(l < p); p_i = p; r_i C_{max}$	79
4.4	Alternative sequences with respect to CASE 1, CASE 1 - Condition 1 for $1 outtree(l < p); p_i = p; r_i C_{max}$	90

4.5	Alternative sequences with respect to CASE 1, CASE 1 - Condition 2 for $1 outtree(l < p); p_i = p; r_i C_{max}$	91
4.6	Alternative sequences when (vii) in CASE 1 - Condition 2 for $1 outtree(l < p); p_i = p; r_i C_{max}$ is violated	92
4.7	Alternative sequences with respect to CASE 1, CASE 1 - Condition 3 for $1 outtree(l < p); p_i = p; r_i C_{max}$	93
4.8	Alternative sequences with respect to CASE 1, CASE 1 - Condition 4 for $1 outtree(l < p); p_i = p; r_i C_{max}$	96
4.9	Alternative sequences with respect to CASE 1, CASE 1 - Condition 5 for $1 outtree(l < p); p_i = p; r_i C_{max}$	98
4.10	Alternative sequences with respect to CASE 1, CASE 1 - Condition 6 for $1 outtree(l < p); p_i = p; r_i C_{max}$	99
4.11	General structure of the proposed algorithm for $1 outtree(l < p); p_i = p; r_i C_{max}$	103
4.12	Scheduling problem instance -precedence graph and release times for $1 outtree(l < p); p_i = p; r_i C_{max}$ -	109
4.13	Instance for integer programming model for $1 outtree(l < p); p_i = p; r_i C_{max}$	110
4.14	Solution for the instance for $1 outtree(l < p); p_i = p; r_i C_{max}$	111
5.1	Instance of <i>chains</i> and the branching scheme for $1 chains(l); r_i \sum C_i$	119

NOMENCLATURE

P	class of polynomial problems
NP	class of nondeterministic polynomial problems
A	set of arcs in a given graph
E	set of edges in a given graph
V	set of vertices in a given graph
C_{max}	makespan
$\sum C_i$	sum of completion times
$\sum w_i C_i$	weighted sum of completion times
L_{max}	maximum lateness
$\sum T_i$	total tardiness
$\sum U_i$	number of tardy tasks
l_{ij}	timelag between tasks i and j
n	size of the instance
n_k	number of tasks in chain k
p_i	processing time of task i
r_i	release time of task i
S_i	start time of task i
C_i	completion time of task i
x_{ij}	binary variable that is equal to 1 if task j is scheduled right after task i

Chapter 1

INTRODUCTION

Scheduling requires allocating the available resources to tasks so as to optimize a given objective function. Application of scheduling methods is vital for manufacturing and service industries. Due to the proliferation of competitive organizations, resources must be used efficiently and production cycles must be shorter and cheaper to satisfy the customer needs while preserving the financial stability. Machine scheduling, vehicle routing, project planning, crew scheduling, airline scheduling, sports scheduling can be given as examples to the problems where scheduling methods are widely used. In these problems resources are in the forms of machines, vehicles, people, facilities, etc. The aim is usually to minimize the cost or to minimize the lateness of tasks so that the customer is satisfied.

In this thesis, machine scheduling is studied. In a machine scheduling problem, tasks are scheduled on a number of machines respecting the given constraints to optimize the objective function. These constraints and objective functions may be in various forms. To denote a scheduling problem, three field notation, $\alpha|\beta|\gamma$, that was introduced by Graham et. al.(1979) is used. In general, in the α field, number of machines and production environment is defined. If there are several machines, production environment may be designed as open shop, flowshop, parallel machines, etc. In the β field, problem specific constraints are shown. These constraints may be the release times, which are the earliest start times for tasks; precedence relations, which are the restrictions on the processing order of tasks; timelag constraints, which are the generalized case of precedence constraints; restrictions on the size of processing times; due dates, which are the latest times the tasks have to be completed. Prece-

dence relations may be in the form of *chains*, *outtree*, *intree*, *tree*, *prec*. In *chains*, each task has at most one immediate predecessor and at most one immediate successor. In *intree*, each task has at most one immediate successor. In *outtree*, each task has at most one immediate predecessor. *tree* is a connected graph with no cycles and it is the generalized case of *intree* and *outtree*. In *prec*, precedence relations are arbitrary. In the γ field, objective function of the scheduling problem is represented. This may be to minimize the makespan, C_{max} , sum of completion times of tasks, $\sum C_i$, weighted sum of completion times of tasks, $\sum w_i C_i$, maximum lateness, L_{max} , total tardiness, $\sum T_i$, number of tardy tasks, $\sum U_i$, total weighted tardiness, $\sum w_i T_i$, weighted number of tardy tasks, $\sum w_i U_i$, etc. Makespan is the maximum of completion times of tasks in the schedule. Sum of completion times is obtained by adding the completion time of the currently scheduled task to the completion time of the task in the former position of the schedule till reaching the task at the last position of the schedule. Lateness of a task is equal to completion time of that task minus the due date for that task. Lateness may be positive or negative. Maximum lateness is the maximum of latenesses of tasks. If the lateness is negative, tardiness is zero. If the lateness is positive, tardiness is equal to lateness. Total tardiness is the sum of tardiness values for all tasks. Number of tardy tasks is the total number of tasks which are completed after their due dates.

This thesis studies the single machine scheduling problems with precedence constraints and nonnegative timelags. Timelag constraint necessitates delaying the start time of a task after the completion time of its immediate predecessor. Given the precedence graph, if task j is the successor of task i , task j cannot be started before the completion time of task i . Moreover, due to the timelag constraint, if task j is an immediate successor of task i , task j cannot be started earlier than the completion time of task i plus the nonnegative timelag duration which is the case in this thesis. In general, timelag relations may be in the forms of start to start or finish to start. Timelag durations may be defined within a range or as exact values. Considering the case where timelag is not exact, if the timelag, l_{ij} , between the tasks i and j is defined

as a nonnegative finish to start lag, then task j cannot be started earlier than l_{ij} time units after the completion of job i . This is also called as the minimal timelag. In the opposite case where the timelag, l_{ij} , is negative, task j cannot start earlier than $|l_{ij}|$ time units before the completion time of job i , or, equivalently, job i cannot finish later than $|l_{ij}|$ time units after the start of job j which defines the maximal timelag. On the other hand, when the timelag is exact, start time of the successor task is not allowed to be in a time window defined by maximal or minimal timelags. It must be a fixed time point related to completion of the preceding task and the exact timelag value. All the rules are also valid for the start to start lags. Such problems may exist in modeling the radar controllers, painting, steel industry, chemical production and food production, etc.

In this thesis, timelag relations are defined as finish to start and the timelags, $l_i = l$, are nonnegative. That is, if i is the immediate predecessor of j , task j cannot start earlier than l time units passed after the completion time of task i . Single machine scheduling problems with precedence constraints and timelags considered in this thesis can be represented as $1|chains(l < p); p_i = p; r_i|C_{max}$, $1|outtree(l < p); p_i = p; r_i|C_{max}$, $1|chains(l); r_i|\sum C_i$, respectively. As explained earlier, first field shows the number of machines which is one in the considered problems. In the second field, *chains* or *outtree* is the structure of the precedence graph, l is the duration of timelag, r_i denotes the release time of task i , p_i is the duration of the processing time of task i which is constant in first two problems and arbitrary in the last one. Third field contains the objective function that is makespan in both $1|chains(l < p); p_i = p; r_i|C_{max}$ and $1|outtree(l < p); p_i = p; r_i|C_{max}$, and sum of completion times in $1|chains(l); r_i|\sum C_i$.

The scheduling problems $1|chains(l < p); p_i = p; r_i|C_{max}$ and $1|outtree(l < p); p_i = p; r_i|C_{max}$ have the same constraints except for the precedence graph. In both problems, the tasks, having identical processing times and timelags but arbitrary release times, are scheduled on a single machine to minimize the makespan. Size of the timelag is bounded by the processing time. Both timelags and processing

times are greater than or equal to 1. In the first problem, the precedence graph is in the form of *chains* while in the second problem it is defined as *outtree*. In *chains*, each task has at most one immediate predecessor and at most one immediate successor. On the other hand, in the *outtree* there is at most one immediate predecessor for each task. In the problem, $1|chains(l); r_i| \sum C_i$, size of the timelag is not dominated by the processing time. That is, timelag is same for all tasks but can take any value. Release times and processing times are arbitrary. The aim is to minimize the sum of completion times of the tasks to be scheduled.

Determining the computational complexity of scheduling problems is an important avenue in scheduling research. The complexity theory is pioneered by Cook(1971), Karp(1972) and Karp(1975) and the problems are classified as either easy or hard. For easy problems, which belong to class P, optimal solution of the problem can be found by a polynomial time algorithm. Harder problems, which belong to class NP, are unlikely to be solved by polynomial algorithms. Problems in class NP can be reduced to each other by polynomial algorithms. NP-completeness of a problem shows that it is the most difficult problem in NP. The class NP contains the class P although the opposite case is unknown. If a polynomial algorithm had been obtained for any NP-complete problem, that algorithm would have been used for every problem in NP and this would show $P = NP$. Whether $P = NP$ or not is still an open question in complexity theory. Binary integer programming problem, traveling salesman problem, and the jobshop scheduling problem are some examples of NP-complete problems. A problem is NP-hard, if and only if an NP-complete problem is reducible to that problem.

The computational complexity of the problems $1|chains(l < p); p_i = p; r_i|C_{max}$ and $1|outtree(l < p); p_i = p; r_i|C_{max}$ are open. That is, there is neither a polynomial algorithm nor an NP- hardness proof. This thesis contributes to literature by proving that the problems $1|chains(l < p); p_i = p; r_i|C_{max}$ and $1|outtree(l < p); p_i = p; r_i|C_{max}$ are polynomially solvable. Problems $1|chains(l < p); p_i = p; r_i|C_{max}$ and $1|outtree(l < p); p_i = p; r_i|C_{max}$ deserve attention since there is no proven com-

plexity result in the literature although there exists an extensive work on complexity classes of scheduling problems with precedence constraints and timelags. The problem $1|chains(1); p_i = p; r_i|C_{max}$, where the timelag is unit, is an easier case of the problem $1|chains(l < p); p_i = p; r_i|C_{max}$. The problem $1|chains(1); p_i = p; r_i|C_{max}$ is classified as minimal open in scheduling theory. Minimal open problems are the problems for which the complexity status is not known, but all easier cases are known to be polynomially solvable. Since the problem $1|chains(1); p_i = p; r_i|C_{max}$, is classified as one of the problems having an open complexity, providing a polynomial time algorithm for the harder problem $1|chains(l < p); p_i = p; r_i|C_{max}$ contributes to literature. Likewise, the problem $1|outtree(l); p_i = p; r_i|C_{max}$ is classified as maximal open in scheduling theory. Maximal open problems are the problems for which the complexity status is not known, but all harder cases are known to be NP-hard. This thesis examines the subcase where $l < p$ and contributes to literature by showing that a subcase of a problem, which is likely to be NP-hard, belongs to class P. In this thesis, to show that the problems $1|chains(l < p); p_i = p; r_i|C_{max}$ and $1|outtree(l < p); p_i = p; r_i|C_{max}$ are polynomial, first, optimality conditions are developed; second, optimality proofs are presented based on proposed optimality conditions; third, complexity of the proposed algorithms is analyzed.

This thesis also deals with the problem $1|chains(l); r_i| \sum C_i$, which is the generalization of strongly NP-hard scheduling problems $1|chains(l)| \sum C_i$ and $1|r_i| \sum C_i$. Due to its complexity status, to solve the problem to optimality, a branch and bound algorithm is developed. Branch and bound algorithm is an intelligent enumeration, which is used to solve an NP-hard problem. Although there are some branch and bound methods applied to solve the scheduling problems with timelags in the literature, there is a lack of research considering the objective function, sum of completion times, together with the release time, precedence and timelag constraint as in the addressed problem.

All proposed algorithms were coded in C programming language and tested on a 2.5 GHz PC with Intel Core i5 processor 4GB of RAM. Furthermore, for each

scheduling problem, corresponding integer programming model was formulated and implemented in GAMS where ILOG CPLEX 12.1.0 was used.

In this thesis, existing results for scheduling problems with timelags are given in Chapter 2. Chapter 3 includes the research on the problem of scheduling chains with nonnegative timelags to minimize the makespan, that is $1|chains(l < p); p_i = p; r_i|C_{max}$. Optimality proofs, pseudo code and complexity analysis for the suggested algorithm and the related integer programming model are given in this chapter. Also, the computational results are presented. Chapter 4 considers the problem of scheduling tasks with nonnegative timelags and outtree precedence graph to minimize the makespan, that is $1|outtree(l < p); p_i = p; r_i|C_{max}$. Proposed algorithm with its pseudo code is explained, optimality of the algorithm is proven and complexity is analyzed. Integer programming model is provided. Performance of all models are tested. In Chapter 5, the branch and bound algorithm that is designed for the problem of scheduling chains with nonnegative timelags to minimize the sum of completion times, that is $1|chains(l); r_i|\sum C_i$, and the corresponding integer programming model are described and later the computational experiments are given. Chapter 6 is devoted to conclusion.

Chapter 2

LITERATURE REVIEW

This thesis considers the single machine scheduling problems with timelag constraints. Minimum and maximum timelags are the time restrictions used to define production processes in food production, chemical production or steel industry (Brucker et al., 1999; Sheen and Liao, 2007). To prevent or reduce food contamination during production process or absorption of airborne particulates during chemical production process, waiting times are limited. Also, time bounds are needed to process the heated steel before its temperature decreases too much (Sheen and Liao, 2007). Maximum timelags are also needed in biotechnology. For example, the extraction of specific blood components from live blood cells require sterile conditions and the risk of contamination is likely to increase when the waiting time becomes too large (Fondreville et al., 2006). Minimal timelags are used to model painting operations. Consider the case where there are several objects to paint, those objects have several different colors and the colors must be painted in a fixed order. A new color cannot be painted while the previous one is wet. Thus the problem becomes a scheduling problem with precedence and timelag constraints (Munier and Sourd, 2003). Furthermore, radar controllers which receive signals after a delay since the pulse emission, some health care treatments that must follow delivery patterns with strictly defined timelags, distributed computing where the master processor organizes data transmission and the timelags between transmissions of input and output files correspond to the execution stage by a slave processor are other examples to applications of different kinds of timelags (Condotta and Shakhlevich, 2012). Additionally, most of the classical scheduling problems can be polynomially reduced to single machine problems with timelags (Brucker et al., 1999). These scheduling problems are general shop problems,

problems with multiprocessor tasks, problems with multi-purpose machines, and single machine problems with sequence-dependent changeover times.

One of the scheduling problems considered in this thesis is denoted by $1|outtree(l < p); p_i = p; r_i|C_{max}$. In this problem, a group of tasks with precedence, timelag and release time constraints must be scheduled on a single machine to minimize the makespan. The precedence graph is in the form of outtree where each task has at most one immediate predecessor. Timelag, l , is identical for all tasks and defined as nonnegative finish to start lag. That is, if task j is the immediate successor of task i , timelag between the completion time of the preceding task and start time of the successor task must be at least equal to l . This kind of timelag is also named as minimum timelag. Note that the timelag, l , is allowed to be nonunit but assumed to be smaller than the given processing time, p , which is also the same for all tasks. This problem has an open complexity status and a polynomial algorithm is proposed in this thesis. Also, the generalized case of the problem where timelag is not dominated by the processing time is maximal open.

Existing research includes several studies considering the outtree structure for precedence constrained scheduling problems and providing complexity results. One of the earliest studies is single machine scheduling problem, $1|outtree(l); p_i = 1; r_i|C_{max}$, which has the outtree precedence graph, constant timelags, unit processing times, arbitrary release times with the objective of minimizing the makespan. The problem was proven to be polynomially solvable by Bruno et al. (1980). It should be pointed out that in this problem all processing times are unit contrary to nonunit processing times in the problem $1|outtree(l < p); p_i = p; r_i|C_{max}$, which is addressed in this thesis. When the processing times are all unit, scheduling problem becomes easier compared to the same problem having nonunit processing times. Brucker and Knust (1999) analyzed the scheduling problem $1|outtree(l); p_i = 1; r_i|\sum C_i$. This problem differs from the problem that is studied in Bruno et al. (1980) by its objective function. In this case, objective function is the sum of completion times which makes the problem more difficult than the problem with makespan objective of Bruno et al. (1980).

Brucker and Knust (1999) was only able to find the feasible schedule in $O(n^2)$ time. This result is important since sometimes finding even a feasible schedule is not so easy in scheduling problems with timelag constraints. They also claimed that the single machine scheduling problem $1|outtree(l); p_i = 1|L_{max}$ is NP-hard in the strong sense. In this problem, the aim is to minimize the maximum lateness and the release time constraint is removed. In all of the problems Bruno et al.(1980) and Brucker and Knust (1999) studied, time delay is in the form of positive finish to start timelag. In other words, authors analyzed the minimal timelags. A polynomial time algorithm was proposed by Brucker et al. (2002) to solve the problem $P|p_j = 1; r_j; outtree| \sum C_j$ to optimality. In this scheduling problem, timelag constraint is removed and parallel machine environment is considered to minimize the total completion time. Release times are arbitrary and processing times are unit as in the previous cases. Later, Huo and Leung (2005) proposed an $O(n \log n)$ implementation of the algorithm Brucker et al. (2002) developed.

Another scheduling problem, $1|chains(l < p); p_i = p; r_i|C_{max}$, that is investigated in this thesis has an open complexity, since an easier scheduling problem, $1|chains(1); p_i = p; r_i|C_{max}$, where the timelag is unit is classified as minimal open in terms of complexity. In this study, it has been shown that the problem $1|chains(l < p); p_i = p; r_i|C_{max}$ is polynomially solvable. In this problem all the constraints are the same as the ones in $1|outtree(l < p); p_i = p; r_i|C_{max}$, except for the precedence graph. The precedence graph is in the form of chains where each task has at most one immediate predecessor and one immediate successor.

The earlier papers providing complexity results for the problems with chain like precedence constraints are summarized as follows. Munier and Sourd (2003) suggested an $O(n)$ algorithm for $1|chains(l); p_i = p|C_{max}$, an $O(n \log n)$ algorithm for $1|chains(l_{ij} \leq p_k)|C_{max}$ and a quadratic time algorithm for $1|chains(l_{ij} \leq p); p_j = p| \sum C_i$. When $1|chains(l); p_i = p|C_{max}$ is compared with $1|chains(l < p); p_i = p; r_i|C_{max}$, which is addressed in this study, it is seen that the release time constraint is not active in the former problem. In $1|chains(l_{ij} \leq p_k)|C_{max}$, timelags and

processing times are arbitrary as opposed to the case in $1|chains(l); p_i = p|C_{max}$ and $1|chains(l < p); p_i = p; r_i|C_{max}$, where the processing times and timelags are the same for all tasks. In $1|chains(l_{ij} \leq p); p_j = p|\sum C_i$, the objective function is different but the processing times are identical. Also, the timelags are arbitrary but their size can be at most equal to the constant processing time value. Brucker et al. (2006) showed that $1|chains(l); p_i = p|\sum C_i$ is polynomially solvable while $1|chains(l)|\sum C_i$ was proven to be NP-hard by Brucker and Knust (1999). The problem, $1|chains(l); p_i = p|\sum C_i$ is a harder version of the problem which was studied by Munier and Sourd (2003), because the objective function is sum of completion times whereas it was the makespan in Munier and Sourd (2003). Thus Brucker et al. (2006) were able to polynomially solve a harder case. If $1|chains(l)|\sum C_i$ is compared with $1|chains(l); p_i = p|\sum C_i$, a small difference is seen. In the former problem, processing times are arbitrary while they are constant in the second one. However, this small change caused the problem to be in class NP instead of being in class P. Yu (1996) and Yu et al. (2004) proved that $1|chains(l_{ij}); p_i = 1|C_{max}$ is NP-hard in the strong sense. In this case, the timelag, l_{ij} , is arbitrary which means that it is different for each task pair having a precedence relation. The problems $1|chains(1); p_i = 1|\sum w_i C_i$ and $1|chains(l)|C_{max}$ were proven to be NP-hard by Tanaev et al. (1994) and Wikum et al. (1994), respectively.

Complexity results for the scheduling problems with different kinds of precedence relations are presented in various papers. Bruno et al. (1980) claimed that $1|intree(l); p_i = 1|C_{max}$ is polynomially solvable. Note that, *intree* is another kind of precedence graph where each task has at most one immediate successor. Brucker and Knust (1999) showed that $1|intree(l); p_i = 1; r_i|C_{max}$ is NP-hard in the strong sense. In the problem $1|intree(l); p_i = 1; r_i|C_{max}$, release time constraints are introduced to the problem $1|intree(l); p_i = 1|C_{max}$. Release time constraints have changed the complexity class of the problem from P to NP. Finta and Liu (1996) found that $1|prec(l_{ij} \geq 1); p_j = 1|C_{max}$ is NP-hard and $1|prec(l_{ij} = 1); p_j \in N + 1|C_{max}$ is polynomially solvable. In these problems, *prec* denotes the arbitrary precedence relations. In the first problem, timelags are greater than or equal to one while in the second

problem processing times are greater than or equal to one. It should be emphasized that the problem with unit processing times and arbitrary timelags is NP-hard whereas the problem with unit timelags and arbitrary processing times is polynomial. Timkovsky (1998) proved that $1|prec(l); p_i = 1||C_{max}$ is NP-hard. This problem is an easier case of $1|prec(l_{ij} \geq 1); p_j = 1|C_{max}$ in Finta and Liu (1996), since the timelags are constant instead of being arbitrary. However, the problem is NP-hard even in this simpler case. Lawler(1978) showed that $1|prec; p_i = 1|\sum w_i C_i$ and $1|prec|\sum C_i$ are NP-hard. Also, Lawler(1973) suggested polynomial algorithms for the problems $1|prec; r_i|C_{max}$ and $1|prec|L_{max}$. In the problems $1|prec; p_i = 1|\sum w_i C_i$, $1|prec|\sum C_i$ and $1|prec; r_i|C_{max}$, the timelag constraint was not taken into consideration. Furthermore, Bruno et al. (1980) proposed polynomial algorithms for the scheduling problems, $1|tree(l); p_i = 1|C_{max}$, $1|intree(l); p_i = 1|L_{max}$ and $1|intree(l); p_i = 1; r_i|L_{max}$. The precedence graph *tree* is the generalized case of the *chains*, *outtree* and *intree*.

The third problem that was examined in this thesis is $1|chains(l); r_i|\sum C_i$ which is strongly NP-hard. This problem is the generalization of the problems $1|chains(l)|\sum C_i$ and $1|r_i|\sum C_i$ which were proven to be strongly NP-hard (Brucker and Knust, 1999; Lenstra et. al., 1977). In the problem $1|chains(l); r_i|\sum C_i$, a group of tasks with chain-like precedence relations, timelag and release time constraints is scheduled on a single machine to minimize the sum of completion times. Unlike the two other problems which were dealt with in this thesis, in this problem, there is no limitation on the size of the timelag, l . In this thesis a branch and bound algorithm was developed to solve the problem.

Literature includes some applications of branch and bound method on NP-hard scheduling problems with timelag constraints. Brucker et al.(1998) studied the problem of scheduling tasks with positive and negative timelags where timelags, l_{ij} , and processing times, p_{ij} were arbitrary and timelags were defined by start to start relations. In this problem, when there is a timelag constraint for the tasks i and j , if the timelag is positive, task j cannot start earlier than the related timelag, l_{ij} , passed after the start time of task i and if the timelag is negative, task j cannot start

earlier than the associated absolute timelag value, $|l_{ij}|$, before the start time of task j . These are defined as the minimum and maximum timelags, respectively. Authors showed that general shop problems, problems with multi-processor tasks, problems with multipurpose machines, problems with sequence-dependent changeover times and combinations of these problems can be polynomially reduced to the addressed single machine scheduling problem with timelag constraints. Thus, they proposed a branch and bound algorithm for the addressed problem to minimize the makespan and tested their algorithm on job shop and open shop problems which are transformed into single machine scheduling problems with timelag constraints due to the lack of benchmark instances. Sheen and Liao (2007) applied the branch and bound method on a scheduling problem which is a more generalized case of the problem of Brucker et al.(1998). They considered both the minimum and maximum timelag constraint, while Brucker et al.(1998) had observed the case where any timelag was either minimal or maximal. That is, Sheen and Liao (2007) defined the timelags within a time window and they analyzed the case where the direction of the arcs were not known. Accordingly, for any two tasks, i and j , minimum and maximum timelags are given for the directions from i to j and from j to i . Start time of task j must remain within the associated minimum and maximum timelag passed after the completion time of task i or start time of task i must remain within the associated minimum and maximum timelag passed after the completion time of task j . The branch and bound algorithm they proposed was able to either find the optimal makespan or show the infeasibility for instances up to 250 jobs within a reasonable time. Fondrevelle et al.(2006) proposed a branch and bound algorithm to minimize the makespan for permutation flow shop scheduling problem with minimal and maximal timelags. They considered finish to start timelags and scheduled the tasks on m machines. Accordingly, given a sequence of jobs, when a task is completed on a machine, the following machine can start to process that task within the time range defined by the minimum and maximum timelags passed after the completion time of the task. Authors tested the instances with 12 jobs and 5 or 10 machines. They

implemented several lower bounds and used some constructive heuristics to provide initial upper bounds. The instances were generated regarding different parameters. Accordingly, the instances with minimal and maximal timelags were more difficult to solve than the instances with maximal timelags only, problems were harder when minimal and maximal time lags were closer. Also, the heterogeneity of the job processing times or the homogeneity of the machine loads contributed to the difficulty of the problems. Moreover, Fondrevelle et al.(2008) developed a branch and bound algorithm for permutation flow shop scheduling with minimal and maximal time lags with the objective of minimizing the weighted sum of machine completion times. The problem has the same constraints as the one in the study of Fondrevelle et al.(2006) even though the objective function is different. The objective function, weighted sum of machine completion times (Fondrevelle et al., 2008) is also different from the classical objective function, weighted sum of completion times. Machine completion time is the completion time of the last operation processed on machine and weighted sum of machine completion times is the weighted sum of machine completion times of m machines. This criterion is associated with resource utilization. To solve the problem, Fondrevelle et al. (2008) developed a branch and bound algorithm based on the algorithm of Fondrevelle et al. (2006). As a lower bound, Fondrevelle et al. (2008) used an adaptation of the best bound among the bounds proposed by Fondrevelle et al. (2006) for the makespan minimization. Fondrevelle et al.(2008) ran the branch and bound algorithm on randomly generated instances for the combinations of 5, 10, 15 machines and 5, 10, 13 jobs. They generated three main groups of instances. First group includes the problems with minimal and maximal timelags, second group includes the classical problems without timelags and third group includes the problems with no-wait case. With respect to the computational results, problems without timelags were solved more easily than problems with timelags or no-wait constraints.

Other studies concerning different solution approaches for the scheduling problems with precedence constraints or timelags are as follows: Condotta and Shakhlevich (2012) proposed a tabu search algorithm for scheduling coupled-operation jobs with

exact timelags to minimize the makespan. In the problem, each job has two operations that are to be scheduled on a single machine. Second operation should start exactly at the time that corresponds to the sum of the completion time of first operation and the timelag. It should be pointed out that exact timelag is not the same as the minimum or maximum timelags. Minimum and maximum timelags are flexible time ranges defined for the required time delay although the exact timelag is fixed. Condotta and Shakhlevich (2012) developed tabu search heuristic based on efficient job reinsertion which outperformed the earlier algorithms. Wang et al. (2011) were able to minimize the weighted sum of squared completion times for deteriorating jobs with chain precedence constraints in polynomial time. In the study, the processing time of a job is a simple linear function of its starting time. These jobs are called deteriorating jobs. Ecker and Tanas (2012) found that the relaxation of the single processor coupled tasks scheduling problem with unit processing times, constant timelags and strict precedence constraints is polynomially solvable although the problem itself is NP-hard in the strong sense. Coupled tasks scheduling is used for controlling radar devices. Authors assumed that the precedence graph is in the form of chains and suggested an $O(n \log n)$ algorithm to solve the problem. Baptiste and Sadykov (2010) suggested time indexed formulations for the single machine scheduling problem with chain like precedence constraints and constant processing times to produce lower bounds on optimal solutions. The application of the problem is airborne radars scheduling. In the model Baptiste and Sadykov (2010) considered, the radar is a single machine and a radar task is a job to be scheduled on the machine where a job consists of a chain of operations with identical processing times. The operations of the same job should be scheduled with a given frequency and any deviation from this frequency is penalized using a V shape function. The objective is to minimize the total penalty. The approaches given in the paper were based on time indexed mixed integer programming formulations which were supposed to provide tight lower bounds when integrality constraints are relaxed. However, due to the experimental results, this was not the case for the addressed problem when the instances were

randomly generated. Contrary to randomly generated instances, many real life instances of moderate size could be solved to optimality. Liu (2010) used an efficient hybrid approach combining the branch and bound method with several heuristics to solve the problem $1|prec; r_j|L_{max}$. The problem is to find a nonpreemptive schedule for n jobs on a single machine to minimize the maximum lateness subject to given release dates and precedence constraints. The problem $1|prec; r_j|L_{max}$ does not have timelag constraints. However, it contains arbitrary release time and precedence constraints. Problem consists of several subproblems for many other more complicated scheduling problems and can be used to obtain useful bounds or optimal schedules directly. It is closely related to jobshop scheduling. In a jobshop, the sequence of machines for each job is prescribed, and each machine can process at most one job at a time. Processing a job on a machine implies release dates and precedence relations. Potts and Whitehead (2007) presented several heuristic algorithms for the coupled operation single machine scheduling problem with time delays. In the problem, each job consists of two operations that are separated by a time delay which lies within a specified range. Aim is to minimize the makespan. The problem is strongly NP-hard. In the paper, a deterministic and randomized construction algorithm, three descent algorithms and two reactive tabu search algorithms were developed. Ambühl et al. (2006) proposed an approximation algorithm for $1|prec|\sum_j w_j C_j$ which is strongly NP-hard (Lawler 1978; Lenstra et al. 1978). The problem is to schedule precedence constrained jobs on a single machine to minimize the weighted sum of completion times. Authors examined the relationship between the dimension theory of partial orders and the scheduling problem. They proved that the vertex cover graph associated to the scheduling problem is equivalent to the graph of incomparable pairs defined in dimension theory. Due to this conclusion, using the known results in dimension theory, approximation algorithms were derived to solve the scheduling problem. This study shows how the arbitrary precedence graph is related to dimension theory which helps to find approximated solutions for scheduling problems with precedence constraints.

Chapter 3

A POLYNOMIAL ALGORITHM FOR SCHEDULING CHAINS WITH TIMELAGS

In this chapter, the scheduling problem which is denoted by $1|chains(l); p_i = p; r_i|C_{max}$ is considered. The problem has an open complexity status and the aim of this study is to establish the complexity status of this problem. The problem requires scheduling n identical tasks with constant processing times, $p_i = p$, arbitrary release times, r_i , chain like precedence relations and constant timelags, l , on a single machine. Due to the timelag restriction, if there is a precedence relation among two tasks, there has to be a minimal time window, l , between the completion time of the predecessor and start time of the successor. Simultaneous processing of tasks on the machine and preemption are not allowed. The objective of the scheduling problem is to minimize the makespan. In this study, an $O(n^2)$ algorithm that obtains the optimal solution for the case where $1 \leq l < p$ is proposed. Hence it is established that the problem is polynomially solvable. The easier scheduling problem, $1|chains(1); p_i = p; r_i|C_{max}$, is also open. Thus, showing that the harder problem, $1|chains(l < p); p_i = p; r_i|C_{max}$, is polynomially solvable, contributes to literature.

3.1 Problem definition of $1|chains(l < p); p_i = p; r_i|C_{max}$

The scheduling problem can be represented as $1|chains(l < p); p_i = p; r_i|C_{max}$ with the three field notation. The problem is to schedule n tasks on a single machine where the tasks cannot be processed simultaneously and preemption is prohibited. Each task is released into the system at a different time, r_i . Processing time, p_i , and timelag, l , are identical for all tasks. l is greater than or equal to one and less than the processing time. Precedence graph is in the form of chains where each task has at

most one immediate predecessor and at most one immediate successor. When $l < p$, timelag restriction necessitates delaying the start time of a task if it is scheduled right after its immediate predecessor. Thus, time delays occur depending on the precedence graph and sorting order of tasks. The aim is minimizing the makespan C_{max} .

In this thesis, we use the additional notation for $1|chains(l < p); p_i = p; r_i|C_{max}$ as follows: Let K be the total number of chains and n_k be the number of tasks in chain k where $k = 0, \dots, K - 1$. $r_{k,i}$ and $p_{k,i} = p$ denote the release time and processing time of task i , $i = 0, \dots, n_k - 1$, in chain k , $k = 0, \dots, K - 1$, respectively. Regarding this representation, in any chain k , the task shown by the smallest i has the highest priority and priority degree decreases as the value of the index i increases. Subject to the given constraints, makespan C_{max} is minimized.

3.2 Optimality conditions for $1|chains(l < p); p_i = p; r_i|C_{max}$

In this section, an $O(n^2)$ algorithm which is able to obtain the optimal solution of the problem $1|chains(l < p); p_i = p; r_i|C_{max}$ is presented. The proposed algorithm takes the precedence graph and the release time values as input for the first iteration and changes the input for each remaining iteration by dynamically updating the precedence graph and the release time values. At each iteration, the precedence graph is recreated by removing the task that is chosen to be scheduled. The earliest start time of any candidate task is not only determined by its original release time and the completion time of the task that is removed from the precedence graph but also by the occurrence of the time delay which is dependent on the assignment order. This dependency gives the opportunity to eliminate excessive time delays by constructing alternative sequences given the constraints, because, successive assignment of the task pairs which are linked by the precedence relation necessitates delaying the start time of the successor task while successive assignment of the tasks pairs with no precedence link does not. To find the optimal solution, algorithm compares the alternative assignment orders of the tasks which are temporarily created by choosing different candidates for the current position of the schedule and making the appro-

priate assignments to the forthcoming positions up to some predetermined positions. By the proposed method, the candidate task that will not lead to the construction of any partial sequence with excessive time delays before reaching to predetermined positions is scheduled at the current position of the sequence. The remaining members of the best partial sequence are not assigned at this phase as they are temporarily constructed. Predetermined positions are located at a further time point which is obtained by the proposed procedures. This time point is used as a *decision point* within the suggested algorithm. The *decision point* is computed in such a way that none of the possible sequences of the tasks which cannot be assigned into the schedule before that time point has a controversial effect on the best selection strategy of the corresponding iteration. This time point is updated at every iteration so that at each decision phase, the release time and timelag restrictions for a group of unassigned tasks are analyzed and any selection that will definitely eliminate the possibility of obtaining better sequences is prevented.

To summarize, the proposed algorithm works as follows: The task that is to be assigned into the first position of the schedule is chosen due to the *decision point* that is obtained by using the original data of the given instance. Remaining tasks are assigned into the following positions of the schedule by repeating a modified procedure. The modified procedure updates the original problem data and recalculates the *decision point* on the updated data at each iteration. Modifications are needed to consider the completion time of the assigned task and its effect on the possible start times of the candidate tasks by taking the timelag and release time constraints into account and to keep track of the remaining number of tasks in different chains and precedence levels of tasks which may change as the assigned tasks are removed from the graph.

Optimality conditions which define the *decision point* of the first iteration with respect to the original problem data are explained in the following paragraphs and the details of the modified procedure are given later.

Let α , β , θ and ζ be the indexes of the chains which contain the tasks with the

following release times, respectively:

$$r_{\alpha,0} = \min_{0 \leq k \leq K-1} r_{k,0},$$

$$r_{\beta,0} = \min_{0 \leq k \leq K-1} r_{k \neq \alpha,0},$$

$$r_{\theta,0} = \min_{0 \leq k \leq K-1} r_{(k \neq \alpha \& k \neq \beta),0},$$

$$r_{\zeta,0} = \min_{0 \leq k \leq K-1} r_{(k \neq \alpha \& k \neq \beta \& k \neq \theta),0}.$$

In other words, if we order the release times of the first tasks of all chains in the nondecreasing order, the task that has the minimum release time belongs to chain α , the task with the next smallest release time belongs to chain β , the task which is third in the order belongs to chain θ and the task which is the fourth in the order belongs to chain ζ . Note that while labeling the chains, if there are several chains containing the tasks with equal release time values which meet the required conditions, the longest ones are labeled.

Necessary conditions for different propositions are defined as follows:

CASE 1. This case defines the situation where the difference between the release times of the first tasks of chain β and chain α is smaller than the timelag value. For the first iteration, original release time values are used although for the remaining iterations updated earliest start times will be used for this case. This case is not enough to make a decision for scheduling a task at any iteration since the final schedule may result in a nonoptimal solution. Thus some conditions and subconditions are defined.

$$(i) \ r_{\beta,0} - r_{\alpha,0} < l.$$

CASE 1 - Condition 1. Together with CASE 1, this condition mainly helps to find the task, positioned after the first time space at further time points, following a partial block of tasks, in the event that the schedule is started by the first task of chain α . This task is the task g of chain α .

$$(i) \ \exists g \in [1, \min(n_\alpha, n_\beta) - 1] : g = i + 1 \text{ such that it is the first task in chain } \alpha \text{ for which } (2i + 2)p + l \leq r_{\alpha,i+1} - r_{\alpha,0} < (2i + 3)p + l \text{ holds where } i = 0, \dots, \min(n_\alpha, n_\beta) - 2,$$

$$(ii) \ r_{\alpha,i+1} - r_{\alpha,0} \leq (2i+2)p \text{ for all } i = 0, \dots, g-2,$$

$$(iii) \ r_{\beta,i+1} - r_{\beta,0} \leq (2i+2)p \text{ for all } i = 0, \dots, g-2.$$

CASE 1 - Condition 1 - Subcondition 1. In this subcondition, the release times of task g of chain β and the first task of chain θ are checked since they may be alternative choices to task g of chain α after the partial block of tasks. For example, by starting the schedule by the first task of chain β instead of α , a shorter schedule may be found depending on the release time of task g of chain β .

$$(i) \ r_{\theta,0} \geq r_{\alpha,0} + 2gp + l,$$

$$(ii) \ r_{\beta,g} - r_{\beta,0} < 2gp + l,$$

$$(iii) \ r_{\beta,g} < r_{\alpha,g}.$$

CASE 1 - Condition 1 - Subcondition 1.1. In this subcondition, the release times of the tasks which have the indexes greater than g are checked to see where they can be positioned in alternative sequences till reaching a particular group of tasks. Possible time bounds on the start times of the tasks in alternative schedules are determined.

(i) $a \neq 0$ where a is a counter of the following inequality till it becomes invalid:

$$r_{\theta,0} > start_{i+1}^{\beta} = \max(start_i^{\beta} + 2p, r_{\beta,i+1}) \text{ for } i = g, \dots, n_{\beta} - 2 \text{ and } start_g^{\beta} = \max(r_{\beta,0} + 2gp, r_{\beta,g}),$$

(ii) $b = a + 1$ where b is a counter of the following inequality:

$$\min(r_{\theta,0}, r_{\beta,i+1}) < start_{i+1}^{\alpha} = \max(start_i^{\alpha} + 2p, r_{\alpha,i+1}) < start_i^{\alpha} + 2p + l \text{ for } i = g, \dots, g + a \text{ and } start_g^{\alpha} = \max(r_{\alpha,0} + 2gp, r_{\alpha,g}),$$

$$(iii) \ \max_{g \leq i \leq g+a-1} |(start_{i+1}^{\alpha} + 2p) - (start_{i+1}^{\beta} + 2p)| < l,$$

$$(iv) \min(r_{\beta,g+a+1}, r_{\theta,1}, r_{\zeta,0}, \max[\max(start_{g+a}^{\beta} + 2p, r_{\theta,0}) + 2p + l; r_{\alpha,g+a+2}]) < start_{g+a+1}^{\alpha} + 2p.$$

Proposition 1. If all the restrictions defined by CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 1 and CASE 1 - Condition 1 - Subcondition 1.1 are met, the optimal schedule starts at time $r_{\beta,0}$, by the task 0 of chain $k = \beta$.

Proof. Suppose there are K chains in accordance with the conditions given in Proposition 1. For a contradiction, assume that optimal schedule starts at time $r_{\alpha,0}$. Then, there exists a partial sequence σ^1 with $r_{\alpha,0} + (2g+2a+5)p + l \leq C_{max}(\sigma^1) < r_{\alpha,0} + (2g+2a+6)p + (a+2)l$. By swapping the task 0 of chain α with the task 0 of chain β in sequence σ^1 and assigning the remaining tasks accordingly, a new sequence σ^2 starting at time $r_{\beta,0}$ and with $r_{\beta,0} + (2g+2a+5)p \leq C_{max}(\sigma^2) < r_{\alpha,0} + (2g+2a+6)p + (a+2)l$ is found. Although the upper bounds are equal, the lower bound of $C_{max}(\sigma^2)$ is less than the lower bound of $C_{max}(\sigma^1)$ since $r_{\beta,0} + (2g+2a+5)p < r_{\alpha,0} + (2g+2a+5)p + l$ due to the inequality $r_{\beta,0} - r_{\alpha,0} < l$. Therefore, initiating the schedule at time $r_{\alpha,0}$ by task 0 of chain α is not optimal. The proof is valid for any value of g and a .

Example 1. Let g and a be equal to 2 and let the left hand side value in item (iv) of CASE 1 - Condition 1 - Subcondition 1.1 be denoted by r_w . With respect to the restrictions for Proposition 1, the data set is rewritten as $r_{\beta,0} - r_{\alpha,0} < l$, $4p + l \leq r_{\alpha,2} - r_{\alpha,0} < 5p + l$, $r_{\theta,0} \geq r_{\alpha,0} + 4p + l$, $r_{\beta,2} - r_{\beta,0} < 4p + l$, $r_{\beta,2} < r_{\alpha,2}$, $r_{\alpha,1} - r_{\alpha,0} \leq 2p$, $r_{\beta,1} - r_{\beta,0} \leq 2p$. As $a = 2$ and $b = 3$, it is concluded that, $start_4^{\beta} < r_{\theta,0} \leq start_5^{\beta}$ where $start_4^{\beta} = \max[\max[\max(r_{\beta,0} + 4p, r_{\beta,2}) + 2p, r_{\beta,3}] + 2p, r_{\beta,4}]$ and $start_5^{\beta} = \max[start_4^{\beta} + 2p, r_{\beta,5}]$. Additionally, $r_w = \min(r_{\beta,5}, r_{\theta,1}, r_{\zeta,0}, \max[\max(start_4^{\beta} + 2p, r_{\theta,0}) + 2p + l, r_{\alpha,6}])$. From (ii) of CASE 1 - Condition 1 - Subcondition 1.1, $r_{\beta,3} < start_3^{\alpha} < start_2^{\alpha} + 2p + l$, $r_{\beta,4} < start_4^{\alpha} < start_3^{\alpha} + 2p + l$, $r_{\theta,0} < start_5^{\alpha} < start_4^{\alpha} + 2p + l$, $r_w < start_5^{\alpha} + 2p$ where $start_2^{\alpha} = \max(r_{\alpha,0} + 4p, r_{\alpha,2})$. Assuming that optimal solution is obtained by starting at time $r_{\alpha,0}$ and subject to given constraints, alternative sequences are constructed as in Figure 3.1.a, b, d, f, g and represented by σ^1 . These sequences set the bounds for the makespan as $r_{\alpha,0} + 13p + l \leq C_{max}(\sigma^1) < r_{\alpha,0} + 14p + 4l$. By swapping the tasks, new alternative sequences are constructed as in Figure 3.1.c, e,

h and shown by σ^2 . All alternative assignments considered, possible solutions for sequence σ^2 remain in the interval $r_{\beta,0} + 13p \leq C_{max}(\sigma^2) < r_{\alpha,0} + 14p + 4l$. This is an example of a contradiction.

Note that Figure 3.1.a illustrates the partial schedule which sets the lower bound for σ^1 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain α is assigned after task 1 of chain β . Figure 3.1.b shows the partial schedule which sets the lower bound for σ^1 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain β . Similarly, Figure 3.1.c sets the lower bound for σ^2 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain α ; Figure 3.1.d sets the lower bound for σ^1 when $r_{\alpha,2} = r_{\alpha,0} + 4p + l$ and task 2 of chain α is assigned after task 1 of chain β ; Figure 3.1.e sets the lower bound for σ^2 when $r_{\alpha,2} = r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain α ; Figure 3.1.f sets the upper bound for σ^1 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain α is assigned after task 1 of chain β ; Figure 3.1.g sets the upper bound for σ^1 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain β ; Figure 3.1.h sets the upper bound for σ^2 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain α .

Remark. When the restriction on task w is removed, due to the release time value of task 6 of chain α for which there is no information and which may correspond to task w , partial sequence σ^1 may become the better alternative compared to the partial sequence σ^2 which conflicts with Proposition 1. An illustration of this situation is shown in Figure 3.2. Note that when the release time of task 2 of chain α is large enough, that task can be assigned after task 1 of chain α in sequence σ^2 with no additional time delay. In that case, the result in sequence σ^1 can be achieved in sequence σ^2 and optimal schedule is found in any case. Except for this situation, the assignments in Figure 3.2.a, d, f have the potential of generating better solutions when the constraint for task w is violated. This situation is formally defined in Corollary 1.

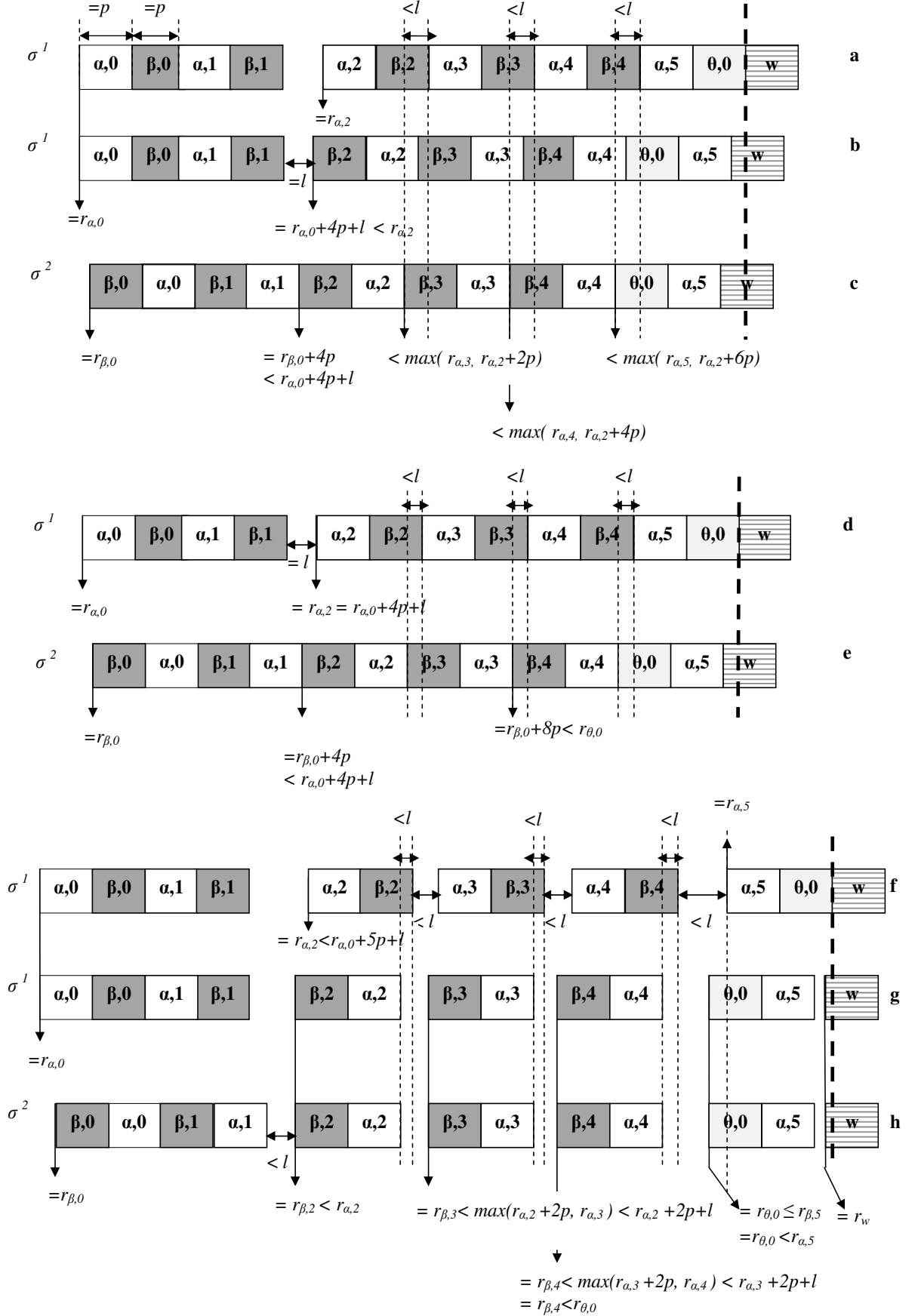


Figure 3.1: Conflicting and supporting examples to Proposition 1 for $1|chains(l < p); p_i = p; r_i|C_{max}$

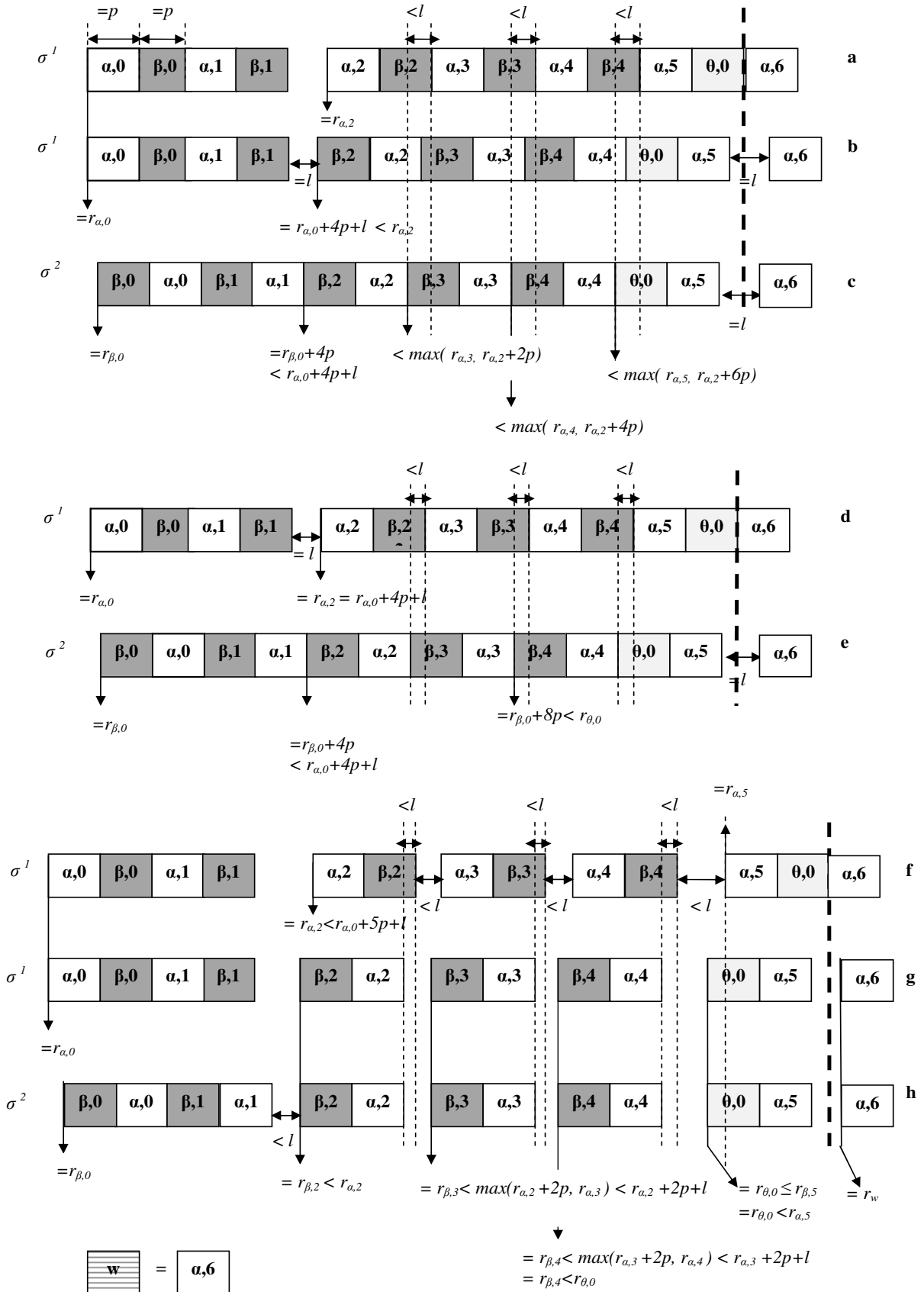


Figure 3.2: Alternative sequences when there is a violation on the constraint (iv) of CASE 1 - Condition 1 - Subcondition 1.1 for Proposition 1 for $1|chains(l < p); p_i = p; r_i | C_{max}$

CASE 1 - Condition 1 - Subcondition 1.2.

- (i) The same as (i) in CASE 1 - Condition 1 - Subcondition 1.1,
- (ii) $b \neq a + 1$ where b is defined by (ii) in CASE 1 - Condition 1 - Subcondition 1.1,
- (iii) The same as (iii) in CASE 1 - Condition 1 - Subcondition 1.1,
- (iv) $c = a + 1$ where c is a counter of the following inequality:

$$\max[\min(r_{\theta,0}, r_{\beta,i+1}), \max(start_i^\alpha + 2p, r_{\alpha,i+1})] < start_i^\alpha + 2p + l \text{ for } i = g, \dots, g + a \text{ and } start_g^\alpha = \max(r_{\alpha,0} + 2gp, r_{\alpha,g}),$$
- (v) $start_{g+a}^\alpha < start_{g+a}^\beta$ & $\max(start_{g+a}^\beta + 2p, r_{\theta,0}) \leq r_{\theta,0} < start_{g+a+1}^\alpha$,
- (vi) The same as (iv) in CASE 1 - Condition 1 - Subcondition 1.1.

Proposition 2. If all the restrictions defined by CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 1 and CASE 1 - Condition 1 - Subcondition 1.2 are met, the optimal schedule starts at time $r_{\beta,0}$.

Proof. Suppose there are K chains having the properties in Proposition 2. Contrary to the fact given by Proposition 2, assume that optimal schedule starts at time $r_{\alpha,0}$. Then, there exists a partial sequence σ^1 containing the task 0 of chain α in its first position. Since $start_{g+a}^\alpha < start_{g+a}^\beta$ & $\max(start_{g+a}^\beta + 2p, r_{\theta,0}) \leq r_{\theta,0} < start_{g+a+1}^\alpha$, for the tightest sequence $\max(start_{g+a}^\beta + 2p, r_{\theta,0}) = r_{\theta,0} = start_{g+a}^\beta + 2p > r_{\alpha,g} + 2(a+1)p = r_{\alpha,0} + 2(g+a+1)p + l$ while for the worst case $start_{g+a}^\beta + 2p < \max(start_{g+a}^\beta + 2p, r_{\theta,0}) = r_{\theta,0} < start_{g+a+1}^\alpha < r_{\alpha,g} + 2(a+1)p + (a+1)l < r_{\alpha,0} + (2g+2a+3)p + (a+2)l$. Thus, $r_{\alpha,0} + (2g+2a+5)p + l < C_{max}(\sigma^1) < r_{\alpha,0} + (2g+2a+6)p + (a+2)l$. By swapping the task 0 of chain α with the task 0 of chain β in sequence σ^1 , a new sequence σ^2 is constructed. As the length of the sequence σ^2 depends directly on $start_{g+a}^\beta$, $r_{\alpha,0} + (2g+2a+5)p + l < C_{max}(\sigma^2) < r_{\alpha,0} + (2g+2a+6)p + (a+2)l$. The upper and lower bounds are equal for both sequencing options. Proposition 2 remains

valid since there is no conflicting result although an alternative sequence having the same length exists. The proof is valid for any value of g and a .

Example 2: Let g and a be equal to 2 and let the left hand side value in item (vi) of CASE 1 - Condition 1 - Subcondition 1.2 be denoted by r_w . Subject to the constraints for Proposition 2, the data set is rewritten as follows: $r_{\beta,0} - r_{\alpha,0} < l$, $4p + l \leq r_{\alpha,2} - r_{\alpha,0} < 5p + l$, $r_{\theta,0} \geq r_{\alpha,0} + 4p + l$, $r_{\beta,2} - r_{\beta,0} < 4p + l$, $r_{\beta,2} < r_{\alpha,2}$, $r_{\alpha,1} - r_{\alpha,0} \leq 2p$, $r_{\beta,1} - r_{\beta,0} \leq 2p$. As $a = 2$, $c = 3$. Note that $start_4^\beta < r_{\theta,0} \leq start_5^\beta$ as defined in Example 1. Moreover, $start_4^\alpha < start_4^\beta$ and $max(start_4^\beta + 2p, r_{\theta,0}) \leq r_{\theta,0} < start_5^\alpha$. Also, r_w has the same meaning. Regarding the rule in the item (iv) of CASE 1 - Condition 1 - Subcondition 1.2, let the following inequalities be valid: $max(start_2^\alpha + 2p, r_{\alpha,3}) < r_{\beta,3} < start_2^\alpha + 2p + l$, $max(start_3^\alpha + 2p, r_{\alpha,4}) < r_{\beta,4} < start_3^\alpha + 2p + l$, $r_{\theta,0} < max(start_4^\alpha + 2p, r_{\alpha,5}) < start_4^\alpha + 2p + l$, $r_w < start_5^\alpha + 2p$ where $start_2^\alpha = max(r_{\alpha,0} + 4p, r_{\alpha,2})$. Assuming that optimal solution is obtained by starting at time $r_{\alpha,0}$ and subject to given constraints, alternative sequences are constructed as in Figure 3.3.a, b, d, f, g and represented by σ^1 . These sequences set the bounds for the makespan as $r_{\alpha,0} + 13p + l < C_{max}(\sigma^1) < r_{\alpha,0} + 14p + 4l$. By swapping the tasks, new alternative sequences are constructed as in Figure 3.3.c, e, h and shown by σ^2 . All alternative assignments considered, possible solutions for sequence σ^2 remain in the interval $r_{\alpha,0} + 13p + l < C_{max}(\sigma^2) < r_{\alpha,0} + 14p + 4l$.

Note that the partial schedule in Figure 3.3.a sets the lower bound for σ^1 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain α is assigned after task 1 of chain β . Likewise, Figure 3.3.b sets the lower bound for σ^1 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain β ; Figure 3.3.c sets the lower bound for σ^2 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain α ; Figure 3.3.d sets the lower bound for σ^1 when $r_{\alpha,2} = r_{\alpha,0} + 4p + l$ and task 2 of chain α is assigned after task 1 of chain β ; Figure 3.3.e sets the lower bound for σ^2 when $r_{\alpha,2} = r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain α ; Figure 3.3.f sets the upper bound for σ^1 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain α is assigned after task 1 of chain β ; Figure 3.3.g sets the upper bound for σ^1 when

$r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain β ; Figure 3.3.h sets the upper bound for σ^2 when $r_{\alpha,2} > r_{\alpha,0} + 4p + l$ and task 2 of chain β is assigned after task 1 of chain α .

CASE 1 - Condition 1 - Subcondition 1.3.

- (i) $a = 0$ where a is defined by (i) in CASE 1 - Condition 1 - Subcondition 1.1,
- (ii) $r_{\theta,0} < \max(r_{\alpha,g+1}, \text{start}_g^\alpha + 2p)$ where $\text{start}_g^\alpha = \max(r_{\alpha,0} + 2gp, r_{\alpha,g})$,
- (iii) $|(\text{start}_g^\alpha + 2p) - (\text{start}_g^\beta + 2p)| < l$,
- (iv) $\min(r_{\beta,g+1}, r_{\theta,1}, r_{\zeta,0}, \max[\max(\text{start}_g^\beta + 2p, r_{\theta,0}) + 2p + l; r_{\alpha,g+2}]) < \text{start}_{g+1}^\alpha + 2p$.

Proposition 3. If all the restrictions defined by CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 1 and CASE 1 - Condition 1 - Subcondition 1.3 are met, the optimal schedule starts at time $r_{\beta,0}$.

Proof. Suppose there are K chains with the restrictions defined by Proposition 3. Suppose that $r_{\alpha,3} < r_{\alpha,2} + 2p + l$ as in the other case the proof is trivial. For a contradiction, assume that optimal schedule starts at time $r_{\alpha,0}$. Then, there exists a partial sequence σ^1 with $r_{\alpha,0} + (2g + 5)p + l \leq C_{\max}(\sigma^1) < r_{\alpha,0} + (2g + 6)p + 2l$. By swapping the task 0 of chain α with the task 0 of chain β in sequence σ^1 and updating the remaining tasks with respect to given restrictions, a new sequence σ^2 starting at time $r_{\beta,0}$ and with $r_{\beta,0} + (2g + 5)p \leq C_{\max}(\sigma^2) < r_{\alpha,0} + (2g + 6)p + 2l$ is found. Thus, any schedule starting by task 0 of chain α is not optimal since a better schedule with $C_{\max}(\sigma^2) < C_{\max}(\sigma^1)$ is found because of the following inequality: $r_{\beta,0} + (2g + 5)p < r_{\alpha,0} + (2g + 5)p + l$. The proof is valid for any value of g .

Example 3.

Let g and a be equal to 2 and let the left hand side value in item (iv) of CASE 1 - Condition 1 - Subcondition 1.3 be denoted by r_w . Subject to the constraints for Proposition 3, the data set is rewritten as follows: $r_{\beta,0} - r_{\alpha,0} < l$, $4p + l \leq r_{\alpha,2} - r_{\alpha,0} < 5p + l$, $r_{\theta,0} \geq r_{\alpha,0} + 4p + l$, $r_{\beta,2} - r_{\beta,0} < 4p + l$, $r_{\beta,2} < r_{\alpha,2}$, $r_{\alpha,1} - r_{\alpha,0} \leq$

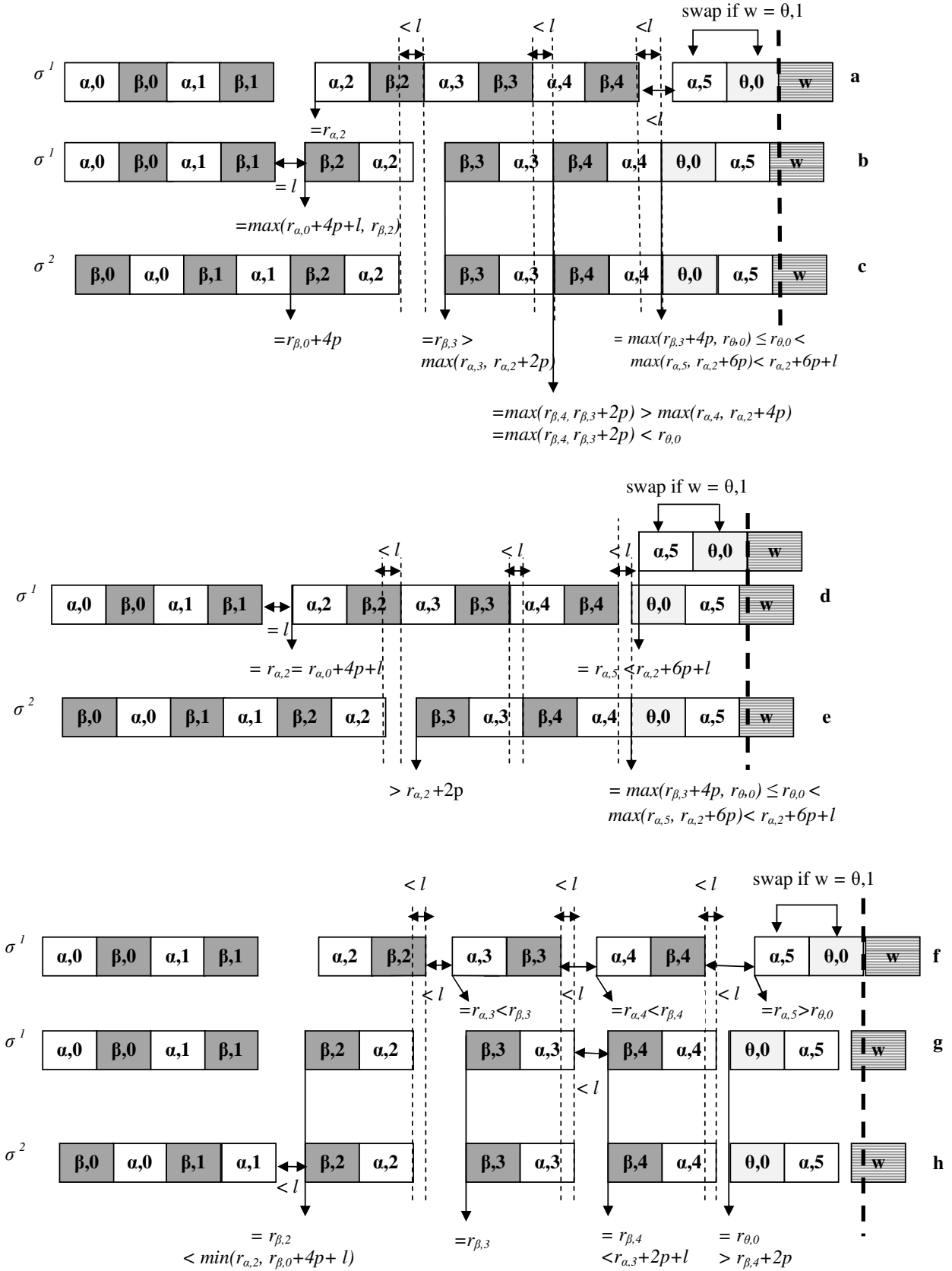


Figure 3.3: Conflicting and supporting examples to Proposition 2 for $1|chains(l < p); p_i = p; r_i|C_{max}$

$2p$, $r_{\beta,1} - r_{\beta,0} \leq 2p$. As $a=0$, $r_{\theta,0} \leq start_{g+1}^\beta = \max(start_g^\beta + 2p, r_{\beta,g+1})$ where $start_g^\beta = \max(r_{\beta,0} + 2gp, r_{\beta,g})$. That is, $r_{\theta,0} \leq start_3^\beta = \max(start_2^\beta + 2p, r_{\beta,3})$ where $start_2^\beta = \max(r_{\beta,0} + 4p, r_{\beta,2})$. From (ii) in CASE 1 - Condition 1 - Subcondition 1.3, $r_{\theta,0} < \max(r_{\alpha,3}, start_2^\alpha + 2p)$ where $start_2^\alpha = \max(r_{\alpha,0} + 4p, r_{\alpha,2})$. Also, $|(start_2^\alpha + 2p) - (start_2^\beta + 2p)| < l$. Moreover, $r_w = \min(r_{\beta,3}, r_{\theta,1}, r_{\zeta,0}, \max[\max(start_2^\beta + 2p, r_{\theta,0}) + 2p + l; r_{\alpha,4}]) < start_3^\alpha + 2p$. When $r_{\alpha,3} < r_{\alpha,2} + 2p + l$, assume that optimal schedule starts at time $r_{\alpha,0}$. The partial sequence constructed by assigning task 0 of chain α into the first position is shown by σ^1 . Figure 3.4.a, b, d, f, g demonstrates the alternative assignments for σ^1 . Accordingly, $r_{\alpha,0} + 9p + l \leq C_{max}(\sigma^1) < r_{\alpha,0} + 10p + 2l$. Now, swap the positions of task 0 of chain α and chain β in sequence σ^1 and assign the remaining tasks appropriately to find a new sequence σ^2 . For σ^2 , alternative partial schedules are possible given the restrictions. These alternatives are shown in Figure 3.4.c, e, h which leads to the bounds on $C_{max}(\sigma^2)$ i.e. $r_{\beta,0} + 9p \leq C_{max}(\sigma^2) < r_{\alpha,0} + 10p + 2l$. This completes the proof.

LEMMA 1. If one of the following subconditions is satisfied in addition to CASE1, CASE 1 - Condition 1 and CASE 1 - Condition 1 - Subcondition 1, the optimal schedule starts at time $r_{\beta,0}$:

- (i) CASE 1 - Condition 1 - Subcondition 1.1,
- (ii) CASE 1 - Condition 1 - Subcondition 1.2,
- (iii) CASE 1 - Condition 1 - Subcondition 1.3.

Note that *decision point* corresponds to $start_{g+a+1}^\alpha + 2p$ for LEMMA 1.

Proof. Proof follows from Proposition 1, Proposition 2 and Proposition 3.

Corollary 1. When the constraint (iv) in CASE 1 - Condition 1 - Subcondition 1.1 or (vi) in CASE 1 - Condition 1 - Subcondition 1.2 or (iv) in CASE 1 - Condition 1 - Subcondition 1.3 is violated, the following strategies are proposed: If the left hand side value is strictly greater than the right hand side value, starting at time $r_{\alpha,0}$ is the best option. On the other hand if there is an equality of left and right hand sides,

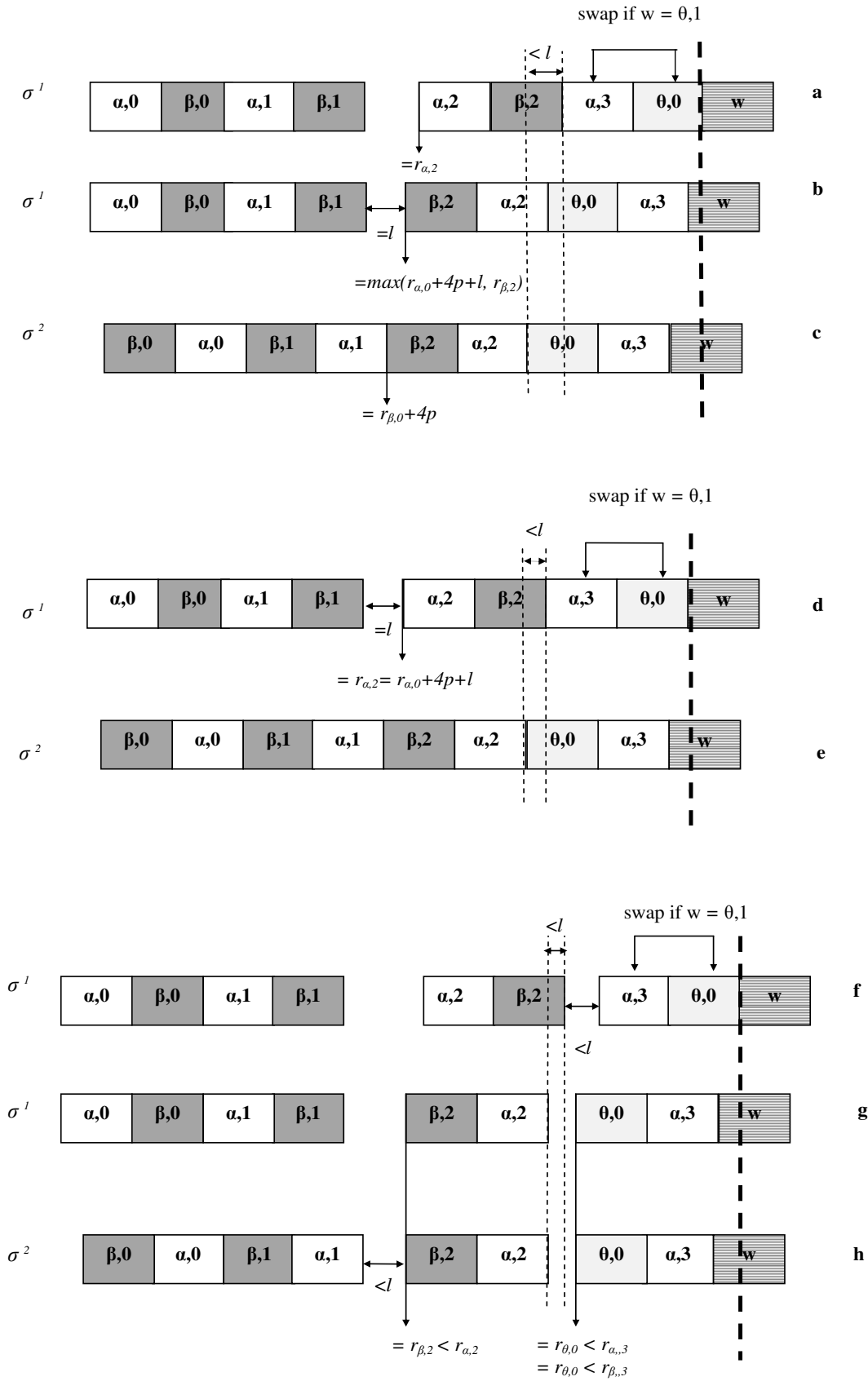


Figure 3.4: Conflicting and supporting examples to Proposition 3 for $1|chains(l < p); p_i = p; r_i|C_{max}$

comparing the number of remaining tasks in chain α and β and initiating the schedule by the task of the longer chain gives the optimal schedule.

Corollary 2. If the constraint (ii) in CASE 1 - Condition 1 - Subcondition 1.1 is violated, the constraint (ii) in CASE 1 - Condition 1 - Subcondition 1.2 is met and the appropriate strategies are as follows: If the constraint (iv) in CASE 1 - Condition 1 - Subcondition 1.2 is not satisfied i.e. $c \neq a + 1$, optimal schedule starts at time $r_{\beta,0}$ when the restrictions in (v) and (vi) are met. If the restriction in (vi) in addition to the one in (iv) is violated, the result in Corollary 1 is valid. Also Corollary 3.a is checked for any violation in (v) in addition to the one in (iv). Although it contains the results for the situation where (iv) is not violated i.e. $c = a + 1$, the same results are valid for the case where $c \neq a + 1$.

Corollary 3.

- (a) When the second restriction of (v) in CASE 1 - Condition 1 - Subcondition 1.2 is destroyed while (i), (ii), (iii), (iv) and the first restriction in (v) are valid, optimal solution is obtained by initiating the schedule at time $r_{\alpha,0}$. When the restrictions in (v) of CASE 1 - Condition 1 - Subcondition 1.2 are replaced by $start_{g+a}^{\alpha} \geq start_{g+a}^{\beta}$ and $max(start_{g+a}^{\beta} + 2p, r_{\theta,0}) < start_{g+a+1}^{\alpha}$, respectively, optimal schedule starts at time $r_{\beta,0}$ and if there is a violation of $max(start_{g+a}^{\beta} + 2p, r_{\theta,0}) < start_{g+a+1}^{\alpha}$ optimal schedule starts at time $r_{\alpha,0}$.
- (b) If there is a violation of $r_{\theta,0} < max(r_{\alpha,g+1}, start_g^{\alpha} + 2p)$ in (ii) of CASE 1 - Condition 1 - Subcondition 1.3, optimal solution is obtained by assigning the task of chain α at time $r_{\alpha,0}$.
- (c) Let the first $(start_{i+1}^{\alpha} + 2p) - (start_{i+1}^{\beta} + 2p)$ that satisfies the condition $(start_{i+1}^{\alpha} + 2p) - (start_{i+1}^{\beta} + 2p) \geq l$ or the condition $(start_{i+1}^{\alpha} + 2p) - (start_{i+1}^{\beta} + 2p) \leq -l$ where $i = g, \dots, g+a-1$ be stored in the memory of the variable *controlalpha*. When $max_{g \leq i \leq g+a-1} |(start_{i+1}^{\alpha} + 2p) - (start_{i+1}^{\beta} + 2p)| \geq l$, which violates the restriction of (iii) in CASE 1 - Condition 1 - Subcondition 1.1 and in CASE 1 - Condition 1 - Subcondition 1.2, if *controlalpha* $\geq l$, optimal schedule starts at

$r_{\beta,0}$ and if $controlalpha \leq -l$, optimal schedule starts at $r_{\alpha,0}$. When $|(start_g^\alpha + 2p) - (start_g^\beta + 2p)| \geq l$, which violates the restriction of (iii) in CASE 1 - Condition 1 - Subcondition 1.3, optimal schedule starts at time $r_{\beta,0}$.

- (d) Note that, as the chosen tasks are removed from the graph iteratively, some of the tasks with the information needed to set the conditions may be lacking at some iterations. Optimal strategies are also determined for those cases. One example is that, when there is no task corresponding to the task having the earliest start time defined by the left hand side value in (iv) of CASE 1 - Condition 1 - Subcondition 1.1, (vi) of CASE 1 - Condition 1 - Subcondition 1.2 and (iv) of CASE 1 - Condition 1 - Subcondition 1.3, optimal schedule starts at time $r_{\beta,0}$.

CASE 1 - Condition 1 - Subcondition 2.

- (i) $r_{\theta,0} \geq r_{\alpha,0} + 2gp + l$,
- (ii) $r_{\beta,g} - r_{\beta,0} < 2gp + l$,
- (iii) $r_{\beta,g} \geq r_{\alpha,g}$.

CASE 1 - Condition 1 - Subcondition 2.1.

- (i) $d \neq 0$ where d is a counter of the following inequality till it becomes invalid:

$$r_{\theta,0} > start_{i+1}^\alpha = \max(start_i^\alpha + 2p, r_{\alpha,i+1}) \text{ for } i = g, \dots, n_\alpha - 2 \text{ and } start_g^\alpha = \max(r_{\alpha,0} + 2gp, r_{\alpha,g}),$$

- (ii) $e = d + 1$ where e is a counter of the following inequality:

$$\min(r_{\theta,0}, r_{\alpha,i+1}) < start_{i+1}^\beta = \max(start_i^\beta + 2p, r_{\beta,i+1}) < start_i^\beta + 2p + l \text{ for } i = g, \dots, g + d \text{ and } start_g^\beta = \max(r_{\beta,0} + 2gp, r_{\beta,g}),$$

- (iii) $\max_{g \leq i \leq g+d-1} |(start_{i+1}^\beta + 2p) - (start_{i+1}^\alpha + 2p)| < l$,

$$(iv) \min(r_{\alpha,g+d+1}, r_{\theta,1}, r_{\zeta,0}, \max[\max(start_{g+d}^{\alpha} + 2p, r_{\theta,0}) + 2p + l; r_{\beta,g+d+2}]) < start_{g+d+1}^{\beta} + 2p.$$

Proposition 4. If all the restrictions defined by CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 2 and CASE 1 - Condition 1 - Subcondition 2.1 are met, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. Suppose there are K chains that meet the constraints given in Proposition 4. For a contradiction, assume that optimal schedule starts at time $r_{\beta,0}$. Then, the minimum value partial sequence $C_{max}(\sigma^1)$ can take is equal to $r_{\alpha,0} + 2gp + l + (2d + 5)p$ when $r_{\beta,g} = r_{\alpha,g} = r_{\alpha,0} + 2gp + l$ with respect to given restrictions and because of the fact that $r_{\beta,g} = r_{\alpha,0} + 2gp + l > r_{\beta,0} + 2gp$. The maximum value $C_{max}(\sigma^1)$ can take is smaller than $r_{\beta,0} + 2gp + l + (2d + 5)p + (d + 1)l$ when $r_{\alpha,0} + 2gp + l < r_{\alpha,g} \leq r_{\beta,g} < r_{\beta,0} + 2gp + l$. When the task 0 of chain β is swapped by the task 0 of chain α in sequence σ^1 a new sequence σ^2 starting at time $r_{\alpha,0}$ is constructed subject to given constraints. The smallest value of $C_{max}(\sigma^2)$ is the same as the one of $C_{max}(\sigma^1)$ since $r_{\alpha,g}$ is not allowed to start before the time $r_{\alpha,0} + 2gp + l$. The upper bound is also the same as the one of $C_{max}(\sigma^1)$. To sum up, $r_{\alpha,0} + (2g + 2d + 5)p + l \leq C_{max}(\sigma^1) < r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$ as well as $r_{\alpha,0} + (2g + 2d + 5)p + l \leq C_{max}(\sigma^2) < r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$. Although the upper and lower bounds do not change for both scheduling options, a shorter makespan for $C_{max}(\sigma^2)$ is possible for smaller values of $r_{\theta,0}$ and r_w in the schedule created for setting the upper bound. Because of this conflicting structure, starting with task 0 of chain β does not give the optimal solution (Example 4).

Example 4. Let g and d be equal to 2. From CASE 1, $r_{\beta,0} - r_{\alpha,0} < l$. From CASE 1 - Condition 1, $r_{\alpha,2} \geq r_{\alpha,0} + 4p + l$, $r_{\alpha,2} < r_{\alpha,0} + 5p + l$, $r_{\alpha,1} - r_{\alpha,0} \leq 2p$, $r_{\beta,1} - r_{\beta,0} \leq 2p$. From CASE 1 - Condition 1 - Subcondition 2, $r_{\theta,0} \geq r_{\alpha,0} + 4p + l$, $r_{\beta,2} - r_{\beta,0} < 4p + l$, $r_{\beta,g} \geq r_{\alpha,g}$. From CASE 1 - Condition 1 - Subcondition 2.1.(i), since $d = 2$, $r_{\theta,0} > start_4^{\alpha} = \max(start_3^{\alpha} + 2p, r_{\alpha,4})$ where $start_3^{\alpha} = \max(start_2^{\alpha} + 2p, r_{\alpha,3})$ and $start_2^{\alpha} = \max(start_0^{\alpha} + 4p, r_{\alpha,2})$. From CASE 1 - Condition 1 - Subcondition 2.1.(ii), as $e = 3$, $\min(r_{\theta,0}, r_{\alpha,5}) < start_5^{\beta} = \max(start_4^{\beta} + 2p, r_{\beta,5}) < start_4^{\beta} + 2p + l$,

$\min(r_{\theta,0}, r_{\alpha,4}) < start_4^\beta = \max(start_3^\beta + 2p, r_{\beta,4}) < start_3^\beta + 2p + l$, $\min(r_{\theta,0}, r_{\alpha,3}) < start_3^\beta = \max(start_2^\beta + 2p, r_{\beta,3}) < start_2^\beta + 2p + l$ where $start_2^\beta = \max(start_0^\beta + 4p, r_{\beta,2})$. From CASE 1 - Condition 1 - Subcondition 2.1.(iii), $\max_{2 \leq i \leq 3} |(start_{i+1}^\beta + 2p) - (start_{i+1}^\alpha + 2p)| < l$. From CASE 1 - Condition 1 - Subcondition 2.1.(iv), $\min(r_{\alpha,5}, r_{\theta,1}, r_{\zeta,0}, \max[\max(start_4^\alpha + 2p, r_{\theta,0}) + 2p + l; r_{\beta,6}]) < start_5^\beta + 2p$. In this inequality, let the left hand side value be denoted by r_w . Given these restrictions, for a contradiction to Proposition 4, suppose that optimal schedule starts at time $r_{\beta,0}$. Then a sequence σ^1 is constructed as in Figure 3.5.a and c. In the best possible assignment shown in Figure 3.5.a, $\max(r_{\beta,0} + 4p, r_{\beta,2}) = r_{\beta,2} = r_{\alpha,2} = r_{\alpha,0} + 4p + l$ and all the remaining tasks are available when needed. Therefore, $C_{max}(\sigma^1) \geq r_{\alpha,0} + 13p + l$. On the other hand, if the release time values take the latest possible values as illustrated in Figure 3.5.c, $\max(r_{\beta,0} + 4p, r_{\beta,2}) = r_{\beta,2} < r_{\beta,0} + 4p + l$, $\max(r_{\beta,2} + 2p, r_{\beta,3}) = r_{\beta,3} < r_{\beta,2} + 2p + l$, $\max(r_{\beta,3} + 2p, r_{\beta,4}) = r_{\beta,4} < r_{\beta,3} + 2p + l$, $\max(r_{\beta,4} + 2p, r_{\theta,0}) = r_{\theta,0} < r_{\beta,5} < r_{\beta,4} + 2p + l$, $r_w < \max(r_{\beta,4} + 2p, r_{\beta,5}) + 2p = r_{\beta,5} + 2p$. Thus $C_{max}(\sigma^1) < r_{\beta,0} + 13p + 4l$. When the task 0 of chain β is swapped by the task 0 of chain α in sequence σ^1 a new sequence σ^2 starting at time $r_{\alpha,0}$ is constructed as in Figure 3.5.b and d. Figure 3.5.b shows the partial schedule when the tasks have the earliest release times given the restrictions. Since $r_{\alpha,2} = r_{\alpha,0} + 4p + l$, $C_{max}(\sigma^2) \geq r_{\alpha,0} + 13p + l$. Figure 3.5.d shows the partial schedule when the tasks have the latest release times given the restrictions. In this case, $r_{\alpha,2} = r_{\beta,2} < r_{\beta,0} + 4p + l$, $r_{\alpha,3} < r_{\beta,3} < r_{\beta,2} + 2p + l$, $r_{\alpha,4} < r_{\beta,4} < r_{\beta,3} + 2p + l$, $r_{\theta,0} < r_{\beta,5} < r_{\beta,4} + 2p + l$, $r_w < r_{\beta,5} + 2p$. Accordingly, $C_{max}(\sigma^2) < r_{\beta,0} + 13p + 4l$. To sum up the lower and upper bounds are equal for sequence σ^1 and σ^2 . However, better solutions of sequence σ^2 can be found for smaller values of $r_{\theta,0}$ and r_w by simply shifting the positions of task 0 of chain θ and task w to left in Figure 3.5.d. Because of this structure, starting with task 0 of chain β does not give the optimal solution.

CASE 1 - Condition 1 - Subcondition 2.2.

- (i) The same as (i) in CASE 1 - Condition 1 - Subcondition 2.1,

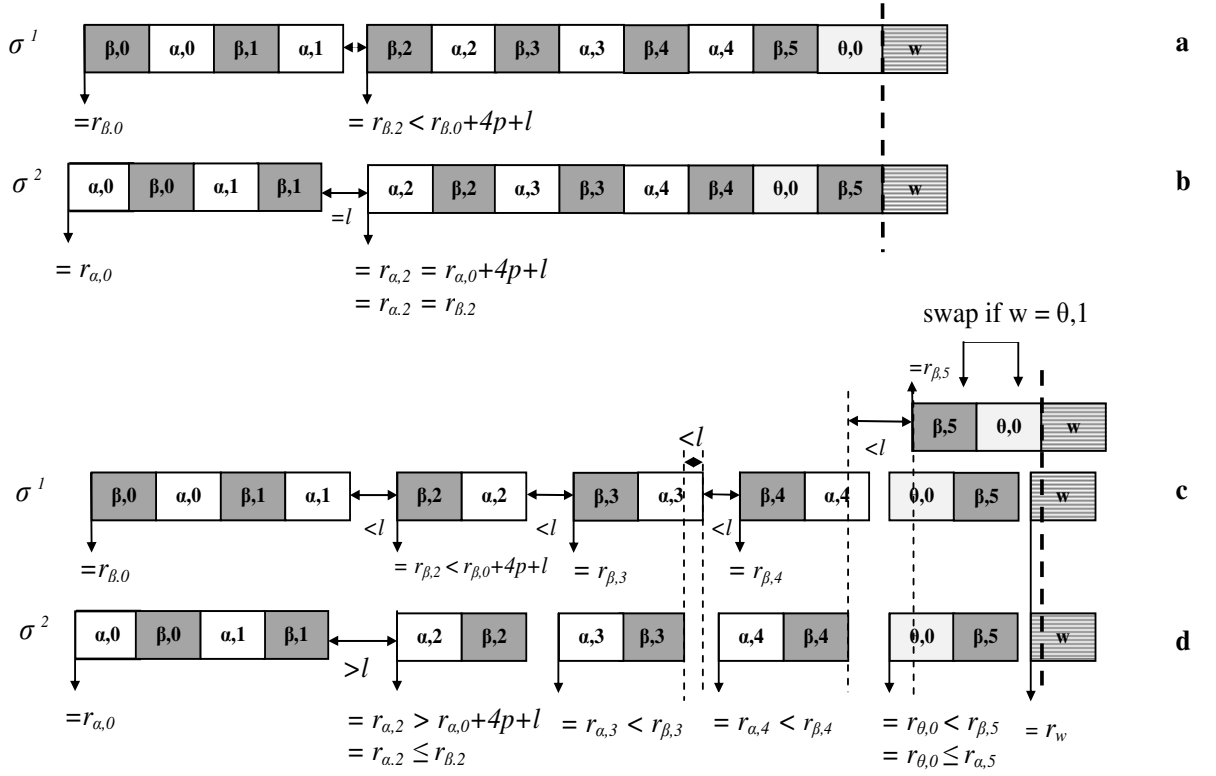


Figure 3.5: Conflicting and supporting examples to Proposition 4 for $1|chains(l < p); p_i = p; r_i|C_{max}$

(ii) $e \neq d + 1$ where e is defined by (ii) in CASE 1 - Condition 1 - Subcondition 2.1,

(iii) The same as (iii) in CASE 1 - Condition 1 - Subcondition 2.1,

(iv) $f = d + 1$ where f is a counter of the following inequality:

$$\max[\min(r_{\theta,0}, r_{\alpha,i+1}), \max(\text{start}_i^\beta + 2p, r_{\beta,i+1})] < \text{start}_i^\beta + 2p + l \text{ for } i = g, \dots, g + d \text{ and } \text{start}_g^\beta = \max(r_{\beta,0} + 2gp, r_{\beta,g}),$$

(v) $\text{start}_{g+d}^\beta < \text{start}_{g+d}^\alpha$ & $\max(\text{start}_{g+d}^\alpha + 2p, r_{\theta,0}) \leq r_{\theta,0} < \text{start}_{g+d+1}^\beta$,

(vi) The same as (iv) in CASE 1 - Condition 1 - Subcondition 2.1.

Proposition 5. If all the restrictions defined by CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 2 and CASE 1 - Condition 1 - Subcondition 2.2 are met, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. Suppose there are K chains as defined in Proposition 5. Assume that optimal schedule starts at time $r_{\beta,0}$. Then, there exists a partial sequence σ^1 with $r_{\alpha,0} + (2g + 2d + 5)p + l < C_{max}(\sigma^1) < r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$. By swapping the first two tasks in sequence σ^1 and scheduling the remaining tasks according to given conditions, a new sequence σ^2 with $r_{\alpha,0} + (2g + 2d + 5)p + l < C_{max}(\sigma^2) < r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$ is found. As $C_{max}(\sigma^1) = C_{max}(\sigma^2)$ in both sequences, accuracy of Proposition 5 is still valid (Example 5).

Example 5. Let g and d be equal to 2. The constraints of CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 2, CASE 1 - Condition 1 - Subcondition 2.2.(i), CASE 1 - Condition 1 - Subcondition 2.1.(iii) which are given in Example 4 are also valid for this example. With respect to CASE 1 - Condition 1 - Subcondition 2.1.(ii), as $e \neq 3$. From CASE 1 - Condition 1 - Subcondition 2.1.(iv), $f = d + 1 = 3$. Considering CASE 1 - Condition 1 - Subcondition 2.1.(ii) and (iv), let the following inequalities be the bounds for release times for this example : $\min(r_{\theta,0}, r_{\alpha,5}) < start_5^\beta = \max(start_4^\beta + 2p, r_{\beta,5}) < start_4^\beta + 2p + l$, $start_4^\beta = \max(start_3^\beta + 2p, r_{\beta,4}) < \min(r_{\theta,0}, r_{\alpha,4}) < start_3^\beta + 2p + l$, $start_3^\beta = \max(start_2^\beta + 2p, r_{\beta,3}) < \min(r_{\theta,0}, r_{\alpha,3}) < start_2^\beta + 2p + l$ where $start_2^\beta = \max(start_0^\beta + 4p, r_{\beta,2})$. Additionally, from CASE 1 - Condition 1 - Subcondition 2.1.(v), $start_4^\beta < start_4^\alpha$ and $\max(start_4^\alpha + 2p, r_{\theta,0}) \leq r_{\theta,0} < start_5^\beta$ and from (vi) $r_w < start_5^\beta + 2p$. Assume that optimal schedule starts with task 0 of chain β . Then, there exists a sequence σ^1 as in Figure 3.6.a and c. The minimum value $C_{max}(\sigma^1)$ can take is greater than $r_{\alpha,0} + 13p + l$ (Figure 3.6.a). The largest value of $C_{max}(\sigma^1)$ is smaller than $r_{\beta,0} + 13p + 4l$ (Figure 3.6.c). That is $r_{\alpha,0} + 13p + l < C_{max}(\sigma^1) < r_{\beta,0} + 13p + 4l$. By swapping the first two tasks in sequence σ^1 a new sequence σ^2 is constructed as in Figure 3.6.b and d. Accordingly, $r_{\alpha,0} + 13p + l < C_{max}(\sigma^2) < r_{\beta,0} + 13p + 4l$. As the lower and upper bounds are the same and $C_{max}(\sigma^1) = C_{max}(\sigma^2)$ in any assignment under the current restrictions, Proposition 5 preserves its validity.

CASE 1 - Condition 1 - Subcondition 2.3.

- (i) $d = 0$ where d is defined by (i) in CASE 1 - Condition 1 - Subcondition 2.1,

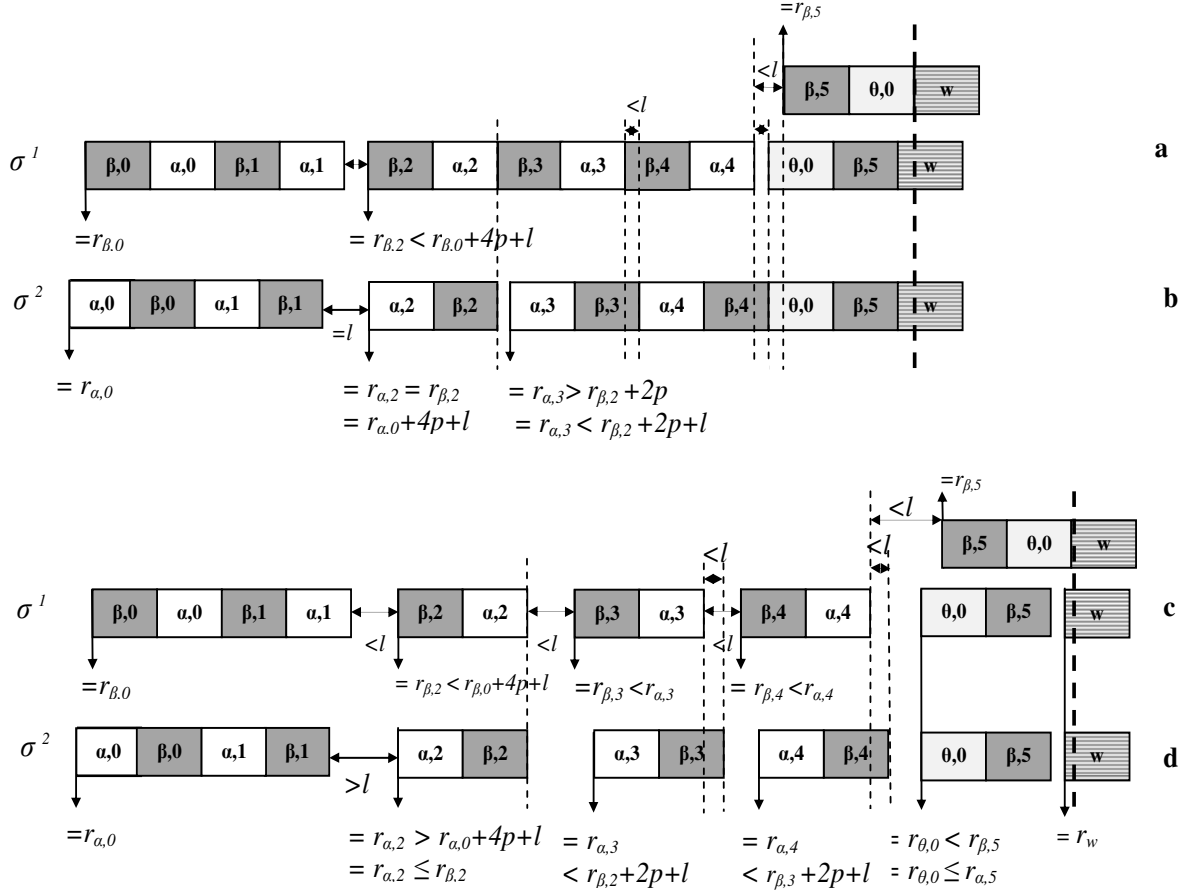


Figure 3.6: Conflicting and supporting examples to Proposition 5 for $1|chains(l < p); p_i = p; r_i|C_{max}$

(ii) $r_{\theta,0} < \max(r_{\beta,g+1}, start_g^\beta + 2p)$ where $start_g^\beta = \max(r_{\beta,0} + 2gp, r_{\beta,g})$,

(iii) $|(start_g^\beta + 2p) - (start_g^\alpha + 2p)| < l$,

(iv) $\min(r_{\alpha,g+1}, r_{\theta,1}, r_{\zeta,0}, \max[\max(start_g^\alpha + 2p, r_{\theta,0}) + 2p + l; r_{\beta,g+2}]) < start_{g+1}^\beta + 2p$.

Proposition 6. If the constraints in CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 2 and CASE 1 - Condition 1 - Subcondition 2.3 are satisfied, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. Suppose there are K chains with the restrictions defined in Proposition 6. Suppose that $r_{\beta,g+1} < r_{\beta,g} + 2gp + l$ since in the other case proof is trivial. Consider

the case when $r_{\theta,0} > r_{\alpha,0} + 2gp + l$. For a contradiction, suppose that there exists a sequence σ^1 that starts at time $r_{\beta,0}$ and assume that it leads to the optimal solution. Given the restrictions, $r_{\alpha,0} + (2g+5)p + l \leq C_{max}(\sigma^1) < r_{\beta,0} + (2g+5)p + 2l$. When the first two tasks are swapped, a sequence σ^2 with a better solution or in the extreme cases an equal solution is found. Better solution may be obtained for a range of values of $r_{\theta,0}$ which remain in the interval $r_{\beta,g} + 2p > r_{\theta,0} > r_{\alpha,g} > r_{\alpha,0} + 2gp + l$ when $r_{\beta,g} > r_{\alpha,g}$. If also the other values $r_{\theta,0}$ can take are considered, it is found that $r_{\alpha,0} + (2g+5)p + l \leq C_{max}(\sigma^2) < r_{\beta,0} + (2g+5)p + 2l$.

When $r_{\theta,0}$ takes its smallest value which is equal to $r_{\alpha,0} + 2gp + l$, there is no need to check the alternative assignment times for the task g of chain β and the task g of chain α since they are allowed to be scheduled in any combination after the assignment of task 0 of chain θ at time $r_{\theta,0} + 2gp + l$. In this case $C_{max}(\sigma^1) = C_{max}(\sigma^2)$.

To summarize, upper and lower bounds for the makespan of both scheduling alternatives are equal. However, when $r_{\beta,g} > r_{\alpha,g}$, a sequence σ^2 with $C_{max}(\sigma^2) < C_{max}(\sigma^1)$ is possible. These facts conflict with the assumption that starting at $r_{\beta,0}$ generates the optimal schedule.

LEMMA 2. In addition to the restrictions of CASE1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 2, if all of the restrictions defined by any one of the following constraint groups are satisfied, starting at time $r_{\alpha,0}$ gives the optimal solution:

- (i) CASE 1 - Condition 1 - Subcondition 2.1,
- (ii) CASE 1 - Condition 1 - Subcondition 2.2,
- (iii) CASE 1 - Condition 1 - Subcondition 2.3.

Note that *decision point* corresponds to $start_{g+d+1}^{\beta} + 2p$ for LEMMA 2.

Proof. Proof follows from Proposition 4, Proposition 5 and Proposition 6.

Corollary 4. When the constraint (iv) in CASE 1 - Condition 1 - Subcondition 2.1 or (vi) in CASE 1 - Condition 1 - Subcondition 2.2 or (iv) in CASE 1 - Condition

1 - Subcondition 2.3 is violated, the following propositions are valid: If the left hand side value is strictly greater than the right hand side value, starting at time $r_{\beta,0}$ is the best option. On the other hand if there is an equality of left and right hand sides, comparing the number of remaining tasks in chain α and β and initiating the schedule by the task of the longer chain gives the optimal schedule.

Corollary 5. If the constraint (ii) in CASE 1 - Condition 1 - Subcondition 2.1 is violated, the constraint (ii) in CASE 1- Condition 1 - Subcondition 2.2 is met and the appropriate strategies are as follows: If the constraint (iv) in CASE 1- Condition 1 - Subcondition 2.2 is not satisfied i.e. $f \neq d + 1$, optimal schedule starts at $r_{\alpha,0}$ when the restrictions in (v) and (vi) are met. If the restriction in (vi) in addition to the one in (iv) is violated, the result in Corollary 4 is valid. Also, Corollary 6.a is checked for any violation in (v) in addition to the one in (iv). Although it contains the results for the situation where (iv) is not violated i.e. $f = d + 1$, the same results are valid for the case where $f \neq d + 1$.

Corollary 6.

- (a) When the second restriction of (v) in CASE 1 - Condition 1 - Subcondition 2.2 is destroyed while (i), (ii), (iii), (iv) and the first restriction in (v) are valid, optimal solution is obtained by initiating the schedule at time $r_{\beta,0}$. When the restrictions of (v) in CASE 1 - Condition 1 - Subcondition 2.2 are replaced by $start_{g+d}^{\beta} \geq start_{g+d}^{\alpha}$ and $max(start_{g+d}^{\alpha} + 2p, r_{\theta,0}) < start_{g+d+1}^{\beta}$, respectively, optimal schedule starts at time $r_{\alpha,0}$ and if there is a violation of $max(start_{g+d}^{\alpha} + 2p, r_{\theta,0}) < start_{g+d+1}^{\beta}$ optimal schedule starts at time $r_{\beta,0}$.
- (b) If there is a violation of $r_{\theta,0} < max(r_{\beta,g+1}, start_g^{\beta} + 2p)$ in (ii) of CASE 1 - Condition 1 - Subcondition 1.3, optimal solution is obtained by assigning the task of chain β at time $r_{\beta,0}$.
- (c) Let the first difference with $(start_{i+1}^{\beta} + 2p) - (start_{i+1}^{\alpha} + 2p) \geq l$ or $(start_{i+1}^{\beta} + 2p) - (start_{i+1}^{\alpha} + 2p) \leq -l$ where $i = g, \dots, g + d - 1$ be stored by the variable *controlbeta*. When $max_{g \leq i \leq g+d-1} |(start_{i+1}^{\beta} + 2p) - (start_{i+1}^{\alpha} + 2p)| \geq l$, which

violates the restriction of (iii) in CASE 1 - Condition 1 - Subcondition 2.1 and in CASE 1 - Condition 1 - Subcondition 2.2, if $controlbeta \geq l$, optimal schedule starts at time $r_{\alpha,0}$ and if $controlbeta \leq -l$, optimal schedule starts at time $r_{\beta,0}$. When $|(start_g^\beta + 2p) - (start_g^\alpha + 2p)| \geq l$, which violates the restriction of (iii) in CASE 1 - Condition 1 - Subcondition 2.3, optimal schedule starts at $r_{\alpha,0}$.

- (d) When some of the tasks with the information needed to set the conditions do not exist, appropriate strategies are applied. For instance, if there is no task corresponding to the task having the earliest start time defined by the left hand side value in (iv) of CASE 1 - Condition 1 - Subcondition 2.1, (vi) of CASE 1 - Condition 1 - Subcondition 2.2 or (iv) of CASE 1 - Condition 1 - Subcondition 2.3, optimal schedule starts at time $r_{\alpha,0}$.

CASE 1 - Condition 1 - Subcondition 3.

- (i) $r_{\theta,0} \geq r_{\alpha,0} + 2gp + l$,
- (ii) $r_{\beta,g} - r_{\beta,0} \geq 2gp + l$.

LEMMA 3. If the data set meets all of the restrictions defined by CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 3, starting at time $r_{\alpha,0}$ gives the optimal solution.

Proof. Suppose there are K chains with the restrictions of CASE 1, CASE 1 - Condition 1, CASE 1 - Condition 1 - Subcondition 3. For a contradiction, assume that there exists a sequence σ^1 of the tasks $0, \dots, g$ of chain β and chain α starting at time $r_{\beta,0}$. In the best assignment, the makespan of sequence σ^1 is equal to $r_{\beta,0} + (2g+2)p + l$. In the worst sequence, as $r_{\alpha,g} < r_{\alpha,0} + (2g+1)p + l$, task g of chain α is assigned after the task $g-1$ of chain α which was completed at time $r_{\beta,0} + 2gp$ before task g of chain β with $r_{\beta,g}$ which is unbounded from above. Note that the latest possible release time under the bound $r_{\alpha,0} + (2g+1)p + l$ is loose enough to contain the compulsory timelag value. Task g of chain β is assigned at a later time with respect to its release time.

Thus, $r_{\beta,0} + (2g + 2)p + l \leq C_{max}(\sigma^1) \leq r_{\beta,g} + p$. If the first two tasks of sequence σ^1 are swapped, a new sequence σ^2 is found. The smallest value $C_{max}(\sigma^2)$ can take is equal to $r_{\alpha,0} + (2g + 2)p + l$. The upper bound is the same as the one for sequence σ^1 since the order of tasks after the task $g - 1$ of chain α or chain β are exactly the same in both sequences. Therefore, $r_{\alpha,0} + (2g + 2)p + l \leq C_{max}(\sigma^2) \leq r_{\beta,g} + p$. As the lower bound of $C_{max}(\sigma^2)$ is smaller than the lower bound of $C_{max}(\sigma^1)$, starting with task 0 of chain β is not optimal. Note that, depending on its release time, task 0 of chain θ can be located anywhere in both partial sequences without creating a conflicting result. Also, since $r_{\beta,g} \geq r_{\beta,0} + 2gp + l > r_{\alpha,0} + 2gp + l$, the release times of tasks after g in chains α and β are not checked. Because, this structure allows to assign task g of chain β instead of task g of chain α in sequence σ^2 without creating a new timelag, if necessary. Therefore, there is no need to make swaps on the preceding tasks in sequence σ^2 to find a shorter partial sequence when there is an opportunity.

CASE 1 - Condition 1 - Subcondition 4.

- (i) $r_{\theta,0} < r_{\alpha,0} + 2gp + l$,
- (ii) $r_{\theta,0} > r_{\beta,0} + 2gp$,
- (iii) $\min(r_{\beta,g}, r_{\theta,1}, r_{\zeta,0}, \max[\max(r_{\beta,0} + 2gp, r_{\theta,0}) + 2p + l; r_{\alpha,g+1}]) < \max(r_{\alpha,0} + 2gp, r_{\alpha,g}) + 2p$.

LEMMA 4. If CASE 1 - Condition 1 - Subcondition 4 is satisfied in addition to the constraints of CASE 1, CASE 1 - Condition 1, starting at time $r_{\beta,0}$ is optimal.

Proof. Suppose there are K chains with the conditions of LEMMA 4.

When $r_{\beta,g} < r_{\theta,0}$, assume that optimal solution is obtained by assigning task 0 of chain α into first position in sequence σ^1 . Task g of chain β cannot be assigned before the time $r_{\alpha,0} + 2gp + l$ because of the timelag constraint, although it has an earlier release time $r_{\beta,g} < r_{\theta,0} < r_{\beta,0} + 2gp + l$. Also since $r_{\alpha,g} \geq r_{\alpha,0} + 2gp + l$, task 0 of chain θ is initiated within the open interval $(r_{\beta,0} + 2gp, r_{\alpha,0} + 2gp + l)$ in σ^1 . Therefore, $r_{\beta,0} + (2g + 3)p < C_{max}(\sigma^1) < r_{\alpha,0} + (2g + 3)p + l$. By swapping the

task 0 of chain β and task 0 of chain α , a new sequence σ^2 with $r_{\beta,0} + (2g + 3)p \leq C_{max}(\sigma^2) < r_{\alpha,0} + (2g + 3)p + l$ is constructed. Since this is a contradiction, starting at time $r_{\alpha,0}$ does not give the optimal solution when $r_{\beta,g} < r_{\theta,0}$.

When $r_{\beta,g} \geq r_{\theta,0}$, assume that optimal sequence σ^1 starts with task 0 of chain α . As $r_{\theta,0} > r_{\beta,0} + 2gp$ and since the lower bound is equal to $r_{\theta,0} + 3p$, $C_{max}(\sigma^1) > r_{\beta,0} + (2g + 3)p$. Likewise, according to (iii) there exists a task with the release time of $\min(r_{\beta,g}, r_{\theta,1}, r_{\zeta,0}, \max[\max(r_{\beta,0} + 2gp, r_{\theta,0}) + 2p + l; r_{\alpha,g+1}]) < r_{\alpha,g} + 2p$ which is assigned after task 0 of chain θ and task g of chain α in sequence σ^1 and this means that $C_{max}(\sigma^1)$ is bounded by $r_{\alpha,g} + 3p$ from above when $r_{\beta,g} \geq r_{\theta,0}$. Thus, $r_{\beta,0} + (2g + 3)p < C_{max}(\sigma^1) < r_{\alpha,g} + 3p$. When the first two tasks are swapped, another sequence σ^2 with $r_{\beta,0} + (2g + 3)p < C_{max}(\sigma^2) < r_{\alpha,g} + 3p$ is found. As, LEMMA 4 is approved by the proof for the assumption that $r_{\beta,g} < r_{\theta,0}$, and no conflicting result is found when $r_{\beta,g} \geq r_{\theta,0}$ optimal schedule starts at time $r_{\beta,0}$.

Corollary 7. When the constraint given in (iii) of CASE 1 - Condition 1 - Subcondition 4 is violated by the inequality $\min(r_{\beta,g}, r_{\theta,1}, r_{\zeta,0}, \max[\max(r_{\beta,0} + 2gp, r_{\theta,0}) + 2p + l; r_{\alpha,g+1}]) > \max(r_{\alpha,0} + 2gp, r_{\alpha,g}) + 2p$, optimal schedule starts at time $r_{\alpha,0}$. If it is violated by the equality, then optimal schedule starts by the task of the longer chain.

CASE 1 - Condition 1 - Subcondition 5.

(i) $r_{\theta,0} < r_{\alpha,0} + 2gp + l$,

(ii) $r_{\theta,0} \leq r_{\beta,0} + 2gp$.

LEMMA 5. Under the restrictions of CASE1, CASE 1- Condition 1 and CASE 1 - Condition 1 - Subcondition 5, initiating the schedule at time $r_{\alpha,0}$ gives the optimal solution.

Proof. The proof is trivial, therefore it is omitted.

CASE 1 - Condition 2.

(i) $\exists g \in [1, \min(n_\alpha, n_\beta) - 1] : g = i + 1$ such that it is the first task in chain

β for which $(2i + 2)p + l \leq r_{\beta,i+1} - r_{\beta,0} < (2i + 3)p + l$ is true where $i = 0, \dots, \min(n_\alpha, n_\beta) - 2$,

(ii) $r_{\alpha,i+1} - r_{\alpha,0} \leq (2i + 2)p$ for all $i = 0, \dots, g - 2$,

(iii) $r_{\beta,i+1} - r_{\beta,0} \leq (2i + 2)p$ for all $i = 0, \dots, g - 2$.

CASE 1 - Condition 2 - Subcondition 1. The definition of g is given in Condition 2.

(i) $r_{\theta,0} \geq r_{\beta,0} + 2gp + l$,

(ii) $r_{\alpha,g} - r_{\alpha,0} < 2gp + l$,

(iii) $r_{\alpha,g} < r_{\beta,g}$.

LEMMA 6. If all the constraints in CASE 1, CASE 1 - Condition 2 and CASE 1 - Condition 2 - Subcondition 1 are met, assigning the task 0 of chain α at time $r_{\alpha,0}$ is optimal.

Proof. The proof is trivial. Suppose there are K chains as appropriate to the structure of the conditions for LEMMA 6. In the sequence σ^1 which starts by the task 0 of chain β at time $r_{\beta,0}$ none of the available tasks can be assigned before time $r_{\beta,0} + 2gp + l$ after the completion time of the task $g - 1$ of chain α which corresponds to $r_{\beta,0} + 2gp$. Because, due to given restrictions, $r_{\theta,0} \geq r_{\beta,0} + 2gp + l$, $r_{\beta,0} \geq r_{\beta,0} + 2gp + l$ and start time of the task g of chain α is greater than or equal to $r_{\beta,0} + 2gp + l$ subject to the timelag constraint. However, in the sequence σ^2 which starts by the task 0 of chain α at time $r_{\alpha,0}$, task g of chain α can start as soon as it is released into the system after the preceding tasks are completed since the time delay is eliminated in this sequence. What is more, as $r_{\alpha,0} + 2gp + l < r_{\beta,0} + 2gp + l$, any permutation on the positions of the tasks which will follow the task $g - 1$ in sequence σ^1 is also allowed in sequence σ^2 without creating a new time delay. Therefore, there is no need for further control on the release times of remaining tasks as they will not provide with a result conflicting with LEMMA 6. This completes the proof.

CASE 1 - Condition 2 - Subcondition 2. The definition of g is given in Condition 2.

$$(i) \ r_{\theta,0} \geq r_{\beta,0} + 2gp + l,$$

$$(ii) \ r_{\alpha,g} - r_{\alpha,0} < 2gp + l,$$

$$(iii) \ r_{\alpha,g} \geq r_{\beta,g}.$$

This is not a consistent group of constraints since $r_{\alpha,g} < r_{\alpha,0} + 2gp + l < r_{\beta,0} + 2gp + l \leq r_{\beta,g}$.

CASE 1 - Condition 2 - Subcondition 3. The definition of g is given in Condition 2.

$$(i) \ r_{\theta,0} \geq r_{\beta,0} + 2gp + l,$$

$$(ii) \ r_{\alpha,g} - r_{\alpha,0} \geq 2gp + l.$$

LEMMA 7. If all of the constraints in CASE 1, CASE 1 - Condition 2, CASE 1 - Condition 2 - Subcondition 3 are satisfied, optimal schedule is found by starting at time $r_{\alpha,0}$.

Proof. The proof is trivial. Any appropriate task that is to be assigned after the task $g - 1$ of chain α cannot start before time $r_{\beta,0} + 2gp + l$ in sequence σ^1 which has the task 0 of chain β on its first position. Likewise, any appropriate task that is to be assigned after the task $g - 1$ of chain β cannot start before time $r_{\alpha,0} + 2gp + l$ in sequence σ^2 which has the task 0 of chain α on its first position. Noting also that $r_{\alpha,0} + 2gp < r_{\beta,0} + 2gp$ obtaining a better solution is always possible in sequence σ^2 and any different arrangement of the tasks after the task $g - 1$ of chain β in sequence σ^2 does not cause an extra time delay compared to sequence σ^1 . Therefore, initializing the schedule at time $r_{\alpha,0}$ is optimal.

CASE 1 - Condition 2 - Subcondition 4: The definition of g is given in Condition 2.

$$(i) \quad r_{\theta,0} < r_{\beta,0} + 2gp + l.$$

LEMMA 8. Under the restrictions of CASE1, CASE 1 - Condition 2, CASE 1 - Condition 2 - Subcondition 4, initializing the schedule at time $r_{\alpha,0}$ is optimal.

Proof. Proof is trivial and done by using a similar argument. Therefore it is omitted.

CASE 1 - Condition 3.

$$(i) \quad \exists g \in [1, \min(n_\alpha, n_\beta) - 1] : g = i + 1 \text{ such that it is the first task in chain } \alpha \text{ for which } (2i + 2)p + l > r_{\alpha,i+1} - r_{\alpha,0} \text{ is valid where } i = 0, \dots, \min(n_\alpha, n_\beta) - 2,$$

$$(ii) \quad r_{\alpha,i+1} - r_{\alpha,0} \leq (2i + 2)p \text{ for all } i = 0, \dots, g - 2,$$

$$(iii) \quad r_{\beta,i+1} - r_{\beta,0} \leq (2i + 2)p \text{ for all } i = 0, \dots, g - 2.$$

CASE 1 - Condition 3 - Subcondition 1. g is defined in Condition 3.

$$(i) \quad r_{\beta,g} \geq r_{\alpha,g},$$

$$(ii) \quad r_{\beta,g} - r_{\beta,0} < 2gp + l.$$

CASE 1 - Condition 3 - Subcondition 1.1. The same as CASE 1 - Condition 1 - Subcondition 2.1 but the index g defined by Condition 3 is used instead of the one of Condition 1.

Proposition 7. If CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 1, CASE 1 - Condition 3 - Subcondition 1.1 are completely satisfied, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. Suppose there are K chains meeting the conditions in Proposition 7. With respect to given restrictions, assume that optimal sequence σ^1 starts at time $r_{\beta,0}$. The lower bound of the makespan of sequence σ^1 is equal to $r_{\beta,0} + (2g + 2d + 5)p$ while the upper bound is strictly smaller than $r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$. If the task 0 of chain α and the task 0 of chain β are swapped a new sequence σ^2 is found. The lower bound of the makespan of sequence σ^2 is equal to $r_{\alpha,0} + (2g + 2d + 5)p$. The upper bound of

the makespan of sequence σ^2 is strictly smaller than $r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$. As the lower bound of $C_{max}(\sigma^2)$ is smaller than the lower bound of $C_{max}(\sigma^1)$, starting with task 0 of chain β is not optimal.

CASE 1 - Condition 3 - Subcondition 1.2. The same as CASE 1 - Condition 1 - Subcondition 2.2 but the index g defined by Condition 3 is used instead of the one by Condition 1.

Proposition 8. If CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 1, CASE 1 - Condition 3 - Subcondition 1.2 are completely satisfied, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. Given the restrictions, assume that optimal sequence σ^1 starts at time $r_{\beta,0}$. In this assignment, $r_{\beta,0} + (2g + 2d + 5)p < C_{max}(\sigma^1) < r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$. By swapping the task 0 of chain β and task 0 of chain α in sequence σ^1 , a new sequence σ^2 with $r_{\beta,0} + (2g + 2d + 5)p < C_{max}(\sigma^2) < r_{\beta,0} + (2g + 2d + 5)p + (d + 2)l$ is constructed. There is no conflicting result for the proposition that starting with task 0 of chain α is optimal, since the boundaries are equal. However, starting with task 0 of chain β may become optimal when there is a violation of the restriction (vi) in CASE 1 - Condition 3 - Subcondition 1.2 which is not a concern for Proposition 8 but is stated in the corollaries.

CASE 1 - Condition 3 - Subcondition 1.3. The same as CASE 1 - Condition 1 - Subcondition 2.3 but the index g defined by Condition 3 is used instead of the one by Condition 1.

Proposition 9. If in addition to preconditions defined by CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 1, the constraints in CASE 1 - Condition 3 - Subcondition 1.3 are satisfied, optimal schedule starts at time $r_{\alpha,0}$.

Proof. Assume that optimal sequence σ^1 starts at time $r_{\beta,0}$. In this assignment, $r_{\beta,0} + (2g + 5)p \leq C_{max}(\sigma^1) < r_{\beta,0} + (2g + 5)p + 2l$. Swap the first two tasks in sequence σ^1 to obtain the alternative sequence σ^2 . In sequence σ^2 , $r_{\alpha,0} + (2g + 5)p \leq C_{max}(\sigma^2) < r_{\beta,0} + (2g + 5)p + 2l$, which is a contradiction.

LEMMA 9. If all the constraints given by one of the following subconditions

are satisfied in addition to the ones in CASE1, CASE 1 - Condition 3 and CASE 1 - Condition 3 - Subcondition 1, the optimal schedule starts at time $r_{\alpha,0}$:

- (i) CASE 1 - Condition 3 - Subcondition 1.1,
- (ii) CASE 1 - Condition 3 - Subcondition 1.2,
- (iii) CASE 1 - Condition 3 - Subcondition 1.3,

Proof: Proof follows from Proposition 7, 8, 9.

Corollary 8: Check Corollary 4. Substitute CASE 1 - Condition 3 - Subcondition 1.1 by CASE 1 - Condition 1 - Subcondition 2.1, substitute CASE 1 - Condition 3 - Subcondition 1.2 by CASE 1 - Condition 1 - Subcondition 2.2, substitute CASE 1 - Condition 3 - Subcondition 1.3 by CASE 1 - Condition 1 - Subcondition 2.3.

Corollary 9: Check Corollary 5. Substitute CASE 1 - Condition 3 - Subcondition 1.1 by CASE 1 - Condition 1 - Subcondition 2.1, substitute CASE 1 - Condition 3 - Subcondition 1.2 by CASE 1 - Condition 1 - Subcondition 2.2. Substitute Corollary 8 and Corollary 10.a by Corollary 4 and Corollary 6.a.

Corollary 10: Check Corollary 6. Substitute CASE 1 - Condition 3 - Subcondition 1.2, CASE 1 - Condition 3 - Subcondition 1.3 and CASE 1 - Condition 3 - Subcondition 1.1 by CASE 1 - Condition 1 - Subcondition 2.2, CASE 1 - Condition 1 - Subcondition 2.3 and CASE 1 - Condition 1 - Subcondition 2.1, respectively.

CASE 1 - Condition 3 - Subcondition 2. Check Condition 3 for g .

- (i) $r_{\beta,g} \geq r_{\alpha,g}$,
- (ii) $r_{\beta,g} - r_{\beta,0} \geq 2gp + l$.

LEMMA 10. If CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 2 are completely satisfied, optimal schedule starts at time $r_{\alpha,0}$. There is no need to check the release times of tasks $g + 1, \dots, g + 1 + d + 1$ whether $d = 0$ or not where d is a counter as defined in CASE 1 - Condition 1 - Subcondition 2.1.

Because, the decision that is based on the restrictions for the tasks $0, \dots, g$ of chain β and chain α is strong enough to determine the optimal schedule.

Proof. For a contradiction, assume that starting at time $r_{\beta,0}$ generates the optimal solution. Ignoring the tasks with $i > g$ in chain α and chain β , the lower and upper bounds on the makespan are checked for the intervals $2gp + l \leq r_{\beta,g} - r_{\beta,0} < (2g+1)p + l$ and $(2g+1)p + l \leq r_{\beta,g} - r_{\beta,0}$. When $2gp + l \leq r_{\beta,g} - r_{\beta,0} < (2g+1)p + l$, $C_{max}(\sigma^1) = r_{\beta,0} + (2g+2)p + l$ and $r_{\alpha,0} + (2g+2)p \leq C_{max}(\sigma^2) < r_{\beta,0} + (2g+2)p + l$. As the upper bound of $C_{max}(\sigma^2)$ is smaller than $C_{max}(\sigma^1)$, starting at $r_{\beta,0}$ is not optimal. When $(2g+1)p + l \leq r_{\beta,g} - r_{\beta,0}$, $r_{\beta,0} + (2g+2)p + l \leq C_{max}(\sigma^1)$ and $r_{\beta,0} + (2g+2)p + l \leq C_{max}(\sigma^2)$. This completes the proof since there is no conflicting result. Note that task 0 of chain θ can be assigned in any position due to its release time and this assignment does not change the result.

Figure 3.7 illustrates the case where $g = 1$. Figure 3.7.a shows the lower bound for σ^1 when $2p + l \leq r_{\beta,1} - r_{\beta,0} < 3p + l$. Figure 3.7.b shows the lower bound for σ^2 when $2p + l \leq r_{\beta,1} - r_{\beta,0} < 3p + l$. Figure 3.7.c shows the upper bound for σ^1 when $2p + l \leq r_{\beta,1} - r_{\beta,0} < 3p + l$. Figure 3.7.d shows the upper bound for σ^2 when $2p + l \leq r_{\beta,1} - r_{\beta,0} < 3p + l$. Figure 3.7.e shows the lower bound for σ^1 when $3p + l \leq r_{\beta,1} - r_{\beta,0}$. Figure 3.7.f shows the lower bound for σ^2 when $3p + l \leq r_{\beta,1} - r_{\beta,0}$. Figure 3.7.g shows the upper bound for σ^1 when $3p + l \leq r_{\beta,1} - r_{\beta,0}$. Figure 3.7.h shows the upper bound for σ^2 when $3p + l \leq r_{\beta,1} - r_{\beta,0}$. Also any different order of tasks after the tasks $g - 1$ of two chains in sequence σ^1 can also be constructed in sequence σ^2 without causing an extra time delay. Figure 3.8 shows the case where $g = 1$ and $2p + l = r_{\beta,1} - r_{\beta,0}$. Figure 3.8.a shows the sequence σ^1 where task 1 of chain β is assigned after task 0 of chain α . Figure 3.8.b shows the sequence σ^2 where task 1 of chain β is assigned after task 0 of chain β . Therefore, searching a better solution that sequence σ^1 may create is meaningless.

CASE 1 - Condition 3 - Subcondition 3. Condition 3 contains the definition of g .

- (i) $r_{\beta,g} < r_{\alpha,g}$,

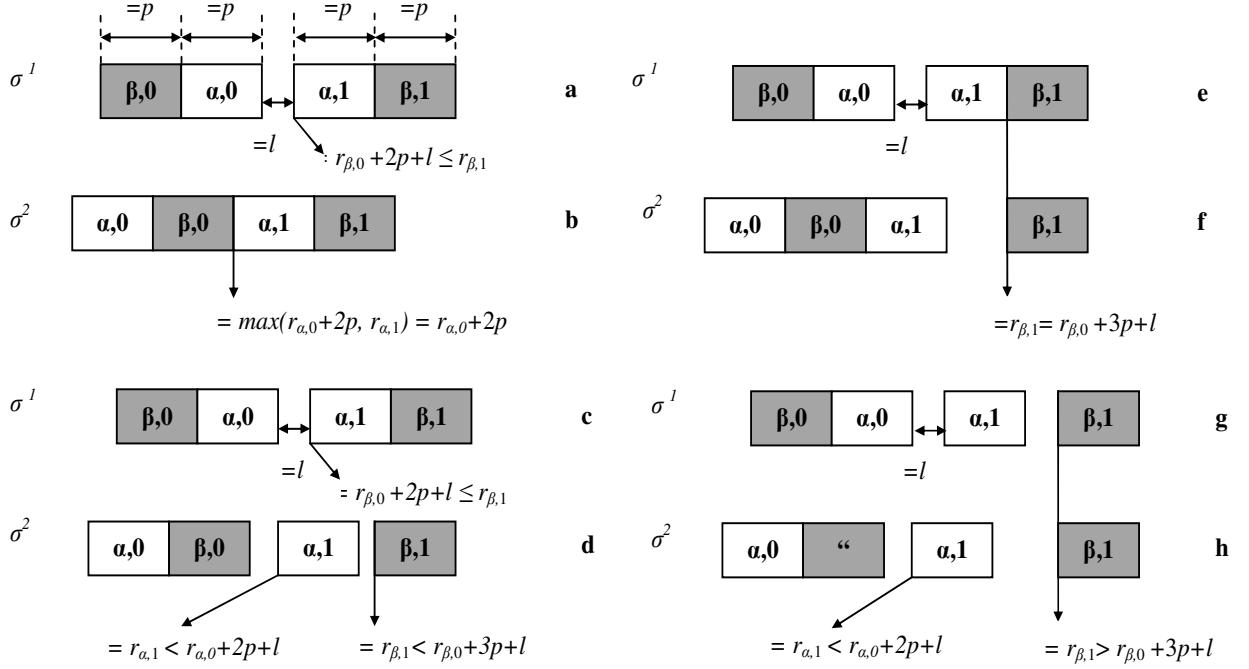


Figure 3.7: Conflicting and supporting examples to LEMMA 10 for $1|chains(l < p); p_i = p; r_i|C_{max}$

(ii) $r_{\alpha,g} - r_{\beta,0} > 2gp$,

(iii) $r_{\theta,0} - r_{\beta,0} > 2gp$.

CASE 1 - Condition 3 - Subcondition 3.1. The same as CASE 1 - Condition 1 - Subcondition 1.1. The only difference is on index g which is determined by Condition 3 for current subcondition.

Proposition 10: If all of the restrictions in CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 3, CASE 1 - Condition 3 - Subcondition 3.1 are met, the optimal schedule starts at time $r_{\beta,0}$.

Proof: Subject to given constraints, assume that optimal solution is obtained by

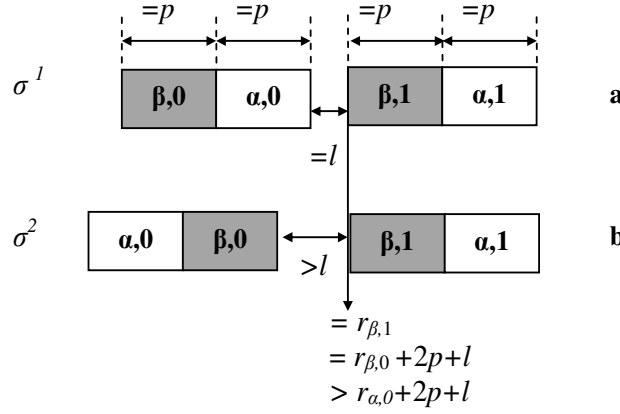


Figure 3.8: Example to LEMMA 10 when $2p + l = r_{\beta,1} - r_{\beta,0}$

starting at time $r_{\alpha,0}$. Then, there exists a sequence σ^1 with $r_{\beta,0} + (2g + 2a + 5)p < C_{max}(\sigma^1) < r_{\alpha,0} + (2g + 2a + 5)p + (a + 2)l$ since $r_{\beta,0} + 2gp < r_{\alpha,g} < r_{\alpha,0} + 2gp + l$. By swapping the task 0 of chain α and task 0 of chain β in sequence σ^1 and assigning the remaining tasks accordingly, a new sequence σ^2 with $r_{\beta,0} + (2g + 2a + 5)p \leq C_{max}(\sigma^2) < r_{\alpha,0} + (2g + 2a + 5)p + (a + 2)l$ is found. When $C_{max}(\sigma^2) = r_{\beta,0} + (2g + 2a + 5)p$, $C_{max}(\sigma^2) < C_{max}(\sigma^1)$ and this is a contradiction.

CASE 1 - Condition 3 - Subcondition 3.2. The same as CASE 1 - Condition 1 - Subcondition 1.2. The only difference is on index g which is determined by Condition 3 for current subcondition.

Proposition 11: If all of the restrictions in CASE 1, CASE 1 - Condition 3, CASE 1- Condition 3 - Subcondition 3, CASE 1- Condition 3 - Subcondition 3.1 are met, the optimal schedule starts at time $r_{\beta,0}$.

Proof: Assume that optimal solution is found by starting at time $r_{\alpha,0}$. Then, there exists a sequence σ^1 with $r_{\beta,0} + (2g + 2a + 5)p < C_{max}\sigma^1 < r_{\alpha,0} + (2g + 2a + 5)p + (a + 2)l$. By swapping the first two tasks in sequence σ^1 and updating the order of remaining tasks, a new sequence σ^2 with $r_{\beta,0} + (2g + 2a + 5)p < C_{max}\sigma^2 < r_{\alpha,0} + (2g + 2a +$

$5)p + (a + 2)l$ is obtained. Neither a worse nor a better solution exists. Thus, the proposition remains valid.

CASE 1 - Condition 3 - Subcondition 3.3. The same as CASE 1 - Condition 1 - Subcondition 1.3. The only difference is on index g which is determined by Condition 3 for current subcondition.

Proposition 12. If CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 3, CASE 1 - Condition 3 - Subcondition 3.3 are completely satisfied, the optimal schedule starts at time $r_{\beta,0}$.

Proof. With respect to given restrictions, sequence σ^1 starting at time $r_{\alpha,0}$ and sequence σ^2 starting at time $r_{\beta,0}$ are compared. Accordingly, optimal schedule does not start with the task 0 of chain α .

LEMMA 11. Under the restrictions of CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 3, if one of the following subconditions is met, assigning the task 0 of chain β into the first position is optimal.

- (i) CASE 1 - Condition 3 - Subcondition 3.1,
- (ii) CASE 1 - Condition 3 - Subcondition 3.2,
- (iii) CASE 1 - Condition 3 - Subcondition 3.3.

Proof. Proof follows from Proposition 10, 11, 12.

Corollary 11. Check Corollary 1. Substitute CASE 1 - Condition 3 - Subcondition 3.1, CASE 1 - Condition 3 - Subcondition 3.2, CASE 1 - Condition 3 - Subcondition 3.3 by CASE 1 - Condition 1 - Subcondition 1.1, CASE 1 - Condition 1 - Subcondition 1.2, CASE 1 - Condition 1 - Subcondition 1.3, respectively.

Corollary 12. Check Corollary 2. Substitute CASE 1 - Condition 3 - Subcondition 3.1 by CASE 1 - Condition 1 - Subcondition 1.1, substitute CASE 1 - Condition 3 - Subcondition 3.2 by CASE 1 - Condition 1 - Subcondition 1.2. Substitute Corollary 11 by Corollary 1 and Corollary 13.a by Corollary 3.a.

Corollary 13. Check Corollary 3. Substitute CASE 1 - Condition 3 - Subcondition 3.2, CASE 1 - Condition 3 - Subcondition 3.3, CASE 1 - Condition 3 - Subcondition 3.1 by CASE 1 - Condition 1 - Subcondition 1.2, CASE 1 - Condition 1 - Subcondition 1.3, CASE 1 - Condition 1 - Subcondition 1.1, respectively.

CASE 1 - Condition 3 - Subcondition 4. Condition 3 contains the definition of g .

- (i) $r_{\beta,g} < r_{\alpha,g}$,
- (ii) $r_{\alpha,g} - r_{\beta,0} > 2gp$,
- (iii) $r_{\theta,0} - r_{\beta,0} \leq 2gp$.

LEMMA 12. Given the restrictions in CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 4, assigning the task 0 of chain α into the first position is optimal.

Proof: Proof is trivial, thus it is omitted.

CASE 1 - Condition 3 - Subcondition 5. Condition 3 contains the definition of g .

- (i) $r_{\beta,g} < r_{\alpha,g}$,
- (ii) $r_{\alpha,g} - r_{\beta,0} \leq 2gp$.

CASE 1 - Condition 3 - Subcondition 5.1. The same as CASE 1 - Condition 1 - Subcondition 2.1. The index g defined by Condition 3 is used for current subcondition.

Proposition 13. If CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 5, CASE 1 - Condition 3 - Subcondition 5.1 are satisfied, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. The sequence σ^1 starting with task 0 of chain β and the sequence σ^2 starting with task 0 of chain α are constructed with respect to the constraints listed

in Proposition 13. As $r_{\beta,0} + (2g + 2d + 5)p \leq C_{max}(\sigma^1) < r_{\beta,0} + (2g + 2d + 5)p + (d + 1)l$ and $r_{\alpha,0} + (2g + 2d + 5)p \leq C_{max}(\sigma^2) < r_{\beta,0} + (2g + 2d + 5)p + (d + 1)l$, assigning task 0 of chain α into first position is optimal.

CASE 1 - Condition 3 - Subcondition 5.2. The same as CASE 1 - Condition 1 - Subcondition 2.2. The index g defined by Condition 3 is used.

Proposition 14. If CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 5, CASE 1 - Condition 3 - Subcondition 5.2 are satisfied, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. The sequence σ^1 starting with task 0 of chain β and the sequence σ^2 starting with task 0 of chain α are constructed with respect to the restrictions of Proposition 14. As $r_{\beta,0} + (2g + 2d + 5)p < C_{max}(\sigma^1) < r_{\beta,0} + (2g + 2d + 5)p + (d + 1)l$ and $r_{\beta,0} + (2g + 2d + 5)p < C_{max}(\sigma^2) < r_{\beta,0} + (2g + 2d + 5)p + (d + 1)l$, proposition remains valid.

CASE 1 - Condition 3 - Subcondition 5.3: The same as CASE 1 - Condition 1 - Subcondition 2.3. The index g defined by Condition 3 is used.

Proposition 15. If CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 5, CASE 1 - Condition 3 - Subcondition 5.3 are satisfied, the optimal schedule starts at time $r_{\alpha,0}$.

Proof. Proof is similar and trivial.

LEMMA 13. In addition to the restrictions of CASE 1, CASE 1 - Condition 3, CASE 1 - Condition 3 - Subcondition 4, if one of the following subconditions are met, assigning the task 0 of chain α into the initial position is optimal.

- (i) CASE 1 - Condition 3 - Subcondition 5.1,
- (ii) CASE 1 - Condition 3 - Subcondition 5.2,
- (iii) CASE 1 - Condition 3 - Subcondition 5.3.

Proof: Proof follows from Proposition 13, 14, 15.

Corollary 14. Check Corollary 4. Substitute CASE 1 - Condition 3 - Subcondition 5.1, CASE 1 - Condition 3 - Subcondition 5.2, CASE 1 - Condition 3 - Subcondition 5.3 by CASE 1 - Condition 1 - Subcondition 2.1, CASE 1 - Condition 1 - Subcondition 2.2, CASE 1 - Condition 1 - Subcondition 2.3, respectively.

Corollary 15. Check Corollary 5. Substitute CASE 1- Condition 3 - Subcondition 5.1 by CASE 1 - Condition 1 - Subcondition 2.1, substitute CASE 1- Condition 3 - Subcondition 5.2 by CASE 1 - Condition 1 - Subcondition 2.2. Substitute Corollary 14 by Corollary 4 and Corollary 16.a by Corollary 6.a.

Corollary 16. Check Corollary 6. Substitute CASE 1 - Condition 3 - Subcondition 5.2, CASE 1 - Condition 3 - Subcondition 5.3, CASE 1 - Condition 3 - Subcondition 5.1 by CASE 1 - Condition 1 - Subcondition 2.B, CASE 1 - Condition 1 - Subcondition 2.C, CASE 1 - Condition 1 - Subcondition 2.A, respectively.

CASE 1 - Condition 4.

- (i) $\exists g \in [1, \min(n_\alpha, n_\beta) - 1] : g = i + 1$ such that it is the first task in chain α for which $r_{\alpha, i+1} - r_{\alpha, 0} \geq (2i + 3)p + l$ is valid where $i = 0, \dots, \min(n_\alpha, n_\beta) - 2$,
- (ii) $r_{\alpha, i+1} - r_{\alpha, 0} \leq (2i + 2)p$ for all $i = 0, \dots, g - 2$,
- (iii) $r_{\beta, i+1} - r_{\beta, 0} \leq (2i + 2)p$ for all $i = 0, \dots, g - 2$.

CASE 1 - Condition 4 - Subcondition 1. Condition 4 defines g .

- (i) $r_{\theta, 0} \geq r_{\alpha, 0} + 2gp + l$,

LEMMA 14. When the constraints defined by CASE 1, CASE 1 - Condition 4, CASE 1 - Condition 4 - Subcondition 1 are satisfied, optimal schedule starts at time $r_{\beta, 0}$.

Proof. Proof is trivial therefore it is omitted.

CASE 1 - Condition 4 - Subcondition 2. Condition 4 contains the definition of g .

$$(i) \ r_{\theta,0} < r_{\alpha,0} + 2gp + l,$$

$$(ii) \ r_{\theta,0} \leq r_{\beta,0} + 2gp.$$

LEMMA 15. When CASE 1 - Condition 4 - Subcondition 2 is satisfied in addition to the constraints defined by CASE 1, CASE 1 - Condition 4, assigning the task 0 of chain α into the first position of the schedule at time $r_{\alpha,0}$ gives the optimal solution.

Proof. Proof is trivial.

CASE 1 - Condition 4 - Subcondition 3. Condition 4 contains the definition of g .

$$(i) \ r_{\theta,0} < r_{\alpha,0} + 2gp + l,$$

$$(ii) \ r_{\theta,0} > r_{\beta,0} + 2gp.$$

LEMMA 16. When CASE 1 - Condition 4 - Subcondition 3 is satisfied in addition to the constraints defined by CASE 1, CASE 1 - Condition 4, assigning the task 0 of chain β into the first position of the schedule at time $r_{\beta,0}$ is optimal.

Proof. Proof is trivial.

CASE 2. $r_{\beta,0} - r_{\alpha,0} \geq l$.

LEMMA 17. If CASE 2 holds, assigning the task 0 of chain α into the initial position of the schedule at time $r_{\alpha,0}$ gives the optimal solution.

Proof: Suppose there are two chains with $r_{\beta,0} - r_{\alpha,0} \geq l$. When $n_{\alpha} > n_{\beta}$, proof is trivial. Consider the case where $n_{\beta} > n_{\alpha}$. Let n_{β} be equal to $n_{\alpha} + h$ where h is a positive integer. For a contradiction, assume that optimal schedule starts at time $r_{\beta,0}$. Then there exists a sequence σ^1 where $C_{max}(\sigma^1) = (n_{\beta} + n_{\alpha}).p + r_{\beta,0} + (h-1)l$ under the assumption of availability of tasks. For sequence σ^2 , $C_{max}(\sigma^2) = (n_{\beta} + n_{\alpha}).p + r_{\alpha,0} + hl$. $C_{max}(\sigma^2) \leq C_{max}(\sigma^1)$ since $r_{\beta,0} \geq r_{\alpha,0} + l$. Note that there will always occur one more timelag if the schedule starts with the task of the shorter chain. However, if chain α is the shorter chain, the excessive timelag in sequence σ^2 is tolerated by the difference

$r_{\beta,0} - r_{\alpha,0} \geq l$ so that starting at time $r_{\alpha,0}$ remains optimal even in the case where chain α has smaller number of tasks.

Modified Procedure:

LEMMA 1, ..., LEMMA 17 define the optimal selection strategy for the first iteration. For the following iterations the same rules are applied in the general manner, however modifications are needed to consider the completion time of the chosen task and its effect on the sequence due to timelag restrictions. Therefore, some additional variables are introduced and used in the modified procedure:

One of the variables defined to serve modified procedure is the completion time of the task that was assigned into the schedule in the previous iteration. This variable is denoted by ct and is equal to the sum of the start time of previously assigned task and the processing time.

Index $\Omega \in [0, K - 1]$ represents the index of chain from which a task was chosen to be assigned into schedule at the previous iteration. n_Ω which is the number of tasks in chain Ω is updated by removing the chosen task from the chain. Also, the remaining tasks in chain Ω are shifted on the direction that will increase their priority levels. By this way, the index i of each task in chain Ω is changed.

When the shifting procedure is completed at any iteration, $r_{\Omega,0}$ denotes the original release time of the unassigned task with highest priority degree in chain Ω . Furthermore, $\max(ct + l, r_{\Omega,0})$ denotes the maximum of completion time of the previously assigned task plus timelag and release time of the candidate task in chain Ω . In other words, $\max(ct + l, r_{\Omega,0})$ is treated as the earliest start time of the task that has the highest priority in chain Ω . It is updated at each iteration depending on the chain that Ω refers.

To integrate $\max(ct + l, r_{\Omega,0})$ into the solution procedure, another dynamic variable set $ar_{k,0}$ where $k = 0, \dots, K - 1$ is defined. The updated release time of task 0 of chain Ω is taken into the new array by the assignment $ar_{\Omega,0} = \max(ct + l, r_{\Omega,0})$ and updated release times of the tasks with the label 0 in all other chains are copied as $ar_{k \neq \Omega,0} = \max(ct, r_{k \neq \Omega,0})$ where $k = 0, \dots, K - 1$. Note that $ar_{k,0}$ temporarily records

the information of the earliest start times of the candidate tasks instead of original release times, for the corresponding iteration.

Let α' , β' , θ' and ζ' be the chains which contain the tasks with the following earliest start times (updated release times), respectively:

$$\begin{aligned} ar_{\alpha',0} &= \min_{0 \leq k \leq K-1} ar_{k,0}, \\ ar_{\beta',0} &= \min_{0 \leq k \leq K-1} ar_{k \neq \alpha',0}, \\ ar_{\theta',0} &= \min_{0 \leq k \leq K-1} ar_{(k \neq \alpha' \& k \neq \beta'),0}, \\ ar_{\zeta',0} &= \min_{0 \leq k \leq K-1} ar_{(k \neq \alpha' \& k \neq \beta' \& k \neq \theta'),0}. \end{aligned}$$

While labeling the chains, if there are several chains having the tasks with equal release times which meet the required conditions, only the longest ones are labeled.

With respect to the iteratively modified information, the following strategies are repeated:

CASE 3. $ar_{\beta',0} - ar_{\alpha',0} < l$

All conditions and subconditions defined for CASE 1 where $r_{\beta,0} - r_{\alpha,0} < l$ are valid for CASE 3 when the variables $r_{\beta,0}, r_{\alpha,0}, r_{\beta,g}, r_{\alpha,g}, n_{\beta}, n_{\alpha}, r_{\theta,0}, r_{\zeta,0}$ are substituted by $ar_{\beta',0}, ar_{\alpha',0}, r_{\beta',g}, r_{\alpha',g}, n_{\beta'}, n_{\alpha'}, ar_{\theta',0}, ar_{\zeta',0}$ (LEMMA 1, . . . , LEMMA 16).

CASE 4. $ar_{\beta',0} - ar_{\alpha',0} \geq l$

Optimal solution is found by assigning the task 0 of chain α' into the current position of the schedule at time $ar_{\alpha',0}$. Proof corresponds to the one for LEMMA 17 which takes its input from CASE 2.

3.3 Complexity of the proposed algorithm for $1|chains(l < p); p_i = p; r_i|C_{max}$

To solve the problem $1|chains(l); p_i = p; r_i|C_{max}$ where $l < p$, an algorithm is designed in conjunction with the properties of the optimal schedule which are proposed and verified in this study. In Section 3.2, those properties were grouped under different propositions and corollaries where the appropriate strategies are suggested for determining the task and the start time of the task that will be assigned into the schedule at current iteration. Pseudo code of the proposed algorithm is as follows:

Pseudo code :

Input: $p, l, r_{k,i}, n_k, K, \alpha, \beta, \theta, \zeta,$

CASE 1: $r_{\beta,0} - r_{\alpha,0} < l$.

When $g = i + 1$ verifies Condition 1

Check whether Condition 1 - Subcondition 1, 2, 3, 4 or 5 is valid

For 1,2, find the *decision point* by related method

Apply the appropriate strategy.

(See LEMMA 1, ..., 5 and Corollary 1, ..., 7)

When $g = i + 1$ verifies Condition 2

Check whether Condition 2 - Subcondition 1, 3 or 4 is valid.

Apply the appropriate strategy

(See LEMMA 6, ..., 8)

When $g = i + 1$ verifies Condition 3

Check whether Condition 3 - Subcondition 1, 2, 3, 4 or 5 is valid

For 1,3,5, find the *decision point* by related method

Apply the appropriate strategy.

(See LEMMA 9, ..., 13 and Corollary 8, ..., 16)

When $g = i + 1$ verifies Condition 4

Check whether Condition 4 - Subcondition 1, 2 or 3 is valid

Apply the appropriate strategy.

(See LEMMA 14, ..., 16)

CASE 2: $r_{\beta,0} - r_{\alpha,0} \geq l$.

Apply directly the appropriate strategy (LEMMA 17)

Intermediary output: start time, Ω .

Update the array $r_{\Omega,i}$ and n_{Ω} and obtain ct and $\max(ct + l, r_{\Omega,0})$.

Create the array $ar_{k,0}$ and find $\alpha', \beta, \theta', \zeta'$ wrt. $ar_{k,0}$

While total number of remaining tasks > 0

Repeat the corresponding procedures by using

$ar_{\beta',0}, ar_{\alpha',0}, r_{\beta',g}, r_{\alpha',g}, n_{\beta'}, n_{\alpha'}, ar_{\theta',0}, ar_{\zeta',0}$

instead of $r_{\beta,0}, r_{\alpha,0}, r_{\beta,g}, r_{\alpha,g}, n_{\beta}, n_{\alpha}, r_{\theta,0}, r_{\zeta,0}$

Update the following variables:

start time, Ω , $r_{\Omega,i}$, n_{Ω} , ct , $\max(ct + l, r_{\Omega,0})$, $ar_{k,0}$, α' , β' , θ' , ζ' .

LEMMA 18. Proposed algorithm is $O(n^2)$.

Proof. The size of the problem is defined by $n = \sum_{k=0}^{K-1} n_k$. Algorithm contains two distinct parts: one for assigning the first task of the schedule and the other one for assigning the remaining tasks. The second stage is the repetition of the modified procedure $n - 1$ times. The repetitive part is dominant in terms of complexity, thus, consider this part for complexity analysis. The following values are found at the beginning of each iteration in the repetitive part: $ar_{\alpha',0} = \min_{0 \leq k \leq K-1} ar_{k,0}$, $ar_{\beta',0} = \min_{0 \leq k \leq K-1} ar_{k \neq \alpha',0}$, $ar_{\theta',0} = \min_{0 \leq k \leq K-1} ar_{(k \neq \alpha' \& k \neq \beta'),0}$, $ar_{\zeta',0} = \min_{0 \leq k \leq K-1} ar_{(k \neq \alpha' \& k \neq \beta' \& k \neq \theta'),0}$. This takes $O((n-1)K)$ time on the whole. Obtaining the $\max(ct + l, r_{\Omega,0})$ which is a comparison of two values takes $O(1)$ time per iteration and $O(n-1)$ time on the whole. By the same way, all comparison processes take $O(1)$ time per iteration and $O(n-1)$ on the whole. Updating $r_{\Omega,i}$ is an $O(n_{\Omega-2})$ procedure and it takes $O((n-1)(n_{\Omega-2}))$ time when repeated $n-1$ times. Additionally, for $0 \leq k < K$, transferring the updated release time values into the array $ar_{k,0}$ takes $O(K)$ time per iteration and $O((n-1)(K))$ for $n-1$ iterations. One of the most complicated parts is tracing the release time values of tasks of any particular chain that has a total number of remaining jobs denoted by n_k , with the expectation of detecting an index g . This search takes $O(n_k)$ time per iteration and $O((n-1)n_k)$ time for $n-1$ iterations. Likewise, other counting procedures on chain α' or β' are $O((n-1)n_k)$ for $n-1$ iterations. Also, at each iteration, $O(n_k)$ time is consumed to compute the entities of the arrays $start_i^{\beta'}$ and $start_i^{\alpha'}$. This takes $O((n-1).n_k)$ for $n-1$ iterations. Checking any subcondition together with its additional constraints to find the *decision point* takes $O(1)$ time per iteration and $O(n-1)$ on the whole.

As a result, complexity of the second stage is shown by the following simplified formulation:

$$O((n-1)K) + O(n-1) + O((n-1)n_k) = O(nK - K) + O(n-1) + O(nn_k - n_k)$$

Complexity of the first stage is as follows:

$$O(K) + O(1) + O(n_k)$$

For the whole algorithm it becomes:

$$O(nK) + O(n) + O(nn_k)$$

As the complexity order is bounded from above by the variable having the largest degree of the polynomial, complexity order of the algorithm is equal to the following statement:

$$\max(O(nK), O(nn_k)) = O(n \cdot \max(K, n_k))$$

In any problem, K or n_k may be larger than the other one since K is the total number of chains, n_k is the number of tasks in any chain and there is no restriction on their comparative sizes. Also, from $K < n$ and $n_k < n$, it follows that $\max(K, n_k) < n$. Thus:

$$O(n \cdot \max(K, n_k)) < O(n^2)$$

which supports the accuracy of the next theorem:

Theorem: $1|chains(l); p_i = p; r_i|C_{max}$ is polynomially solvable for the subcase where $l < p$.

3.4 Integer programming model for $1|chains(l < p); p_i = p; r_i|C_{max}$

An integer programming model that solves problem $1|chains(l); p_i = p; r_i|C_{max}$ to optimality is developed. In this model, the scheduling problem is defined as a network model. Let $G(V, A, E)$ be the given graph where V is the set of vertices, A is the set of directed arcs (i, j) for which the path is allowed to be directed from $i \in V$ to $j \in V$ only, and E is the set of edges (i, j) for which the path has two directions emanating from $i \in V$ or $j \in V$. In order to convert the scheduling problem $1|chains(l); p_i = p; r_i|C_{max}$ into the network model, each task i is represented as a vertex of the graph and two dummy tasks which do not exist in the primary problem are defined as $i = 0 \in V$ and $i = n + 1 \in V$. The set of arcs, A , contains the precedence links between the tasks of a chain, the links emanating from the node 0 and entering into the nodes which are associated with the first tasks in different chains, the links leaving the nodes associated with the last tasks in different chains and are being

directed towards the end node. The set of edges, E is composed of the links between the tasks in different chains where there is no precedence constraint. Given this graph, an integer programming model with disjunctive formulations is presented.

For integer programming model, the solution which is obtained by the polynomial algorithm explained in Section 3.2 is used as an initial solution.

Notation

- Sets and Parameters :

O_i : Set of alternative destinations, $j \in V$ where $(i, j) \in A$ or $(i, j) \in E$, for any $i \in V$.

I_j : Set of alternative sources, $i \in V$ where $(i, j) \in A$ or $(i, j) \in E$, for each $j \in V$.

r_i : Release time for each task $i \in V$. For $i = 0$ or $i = n + 1$, $r_i = 0$. For $i \neq 0$ and $i \neq n + 1$, $r_i \geq 0$.

p_i : Processing time of task i where $i \in V$. If $i \neq 0$ and $i \neq n + 1$, $p_i = p$. When $i = 0$ or $i = n + 1$, $p_i = 0$.

l : Duration of timelag.

Z^0 : Objective function value of the initial solution that is obtained by applying the polynomial algorithm in Section 3.2.

LB^0 : Initial lower bound on the objective function value. This bound is equal to $\min_{0 \leq k < K} r_{k,0} + np$ which is computed by an external program depending on the given instance.

- Decision variables :

x_{ij} : Binary variable for the link between $i \in V$ and $j \in V$. $x_{ij} = 1$ if task j is assigned immediately after task i . $x_{ij} = 0$ in the other case.

S_i : Start time at node $i \in V$.

C_i : Completion time at node $i \in V$.

- Objective function :

C_{n+1} : Completion time at node $n + 1$. It corresponds to the maximum of C_i for all $i \in V$.

Model

$$\text{minimize } C_{n+1} \quad (1)$$

$$C_i = S_i + p_i \quad \text{for each } i \in V \quad (2)$$

$$S_{n+1} \geq C_i \quad \text{for each } i \in V \text{ where } i \neq n+1 \quad (3)$$

$$S_i \geq r_i \quad \text{for each } i \in V \quad (4)$$

$$S_j + M(1 - x_{ij}) \geq C_i \quad \text{for each } (i, j) \in E \text{ and } (i, j) \in A \quad (5)$$

$$S_j \geq C_i + l \quad \text{for each } (i, j) \in A, i \neq 0, j \neq n+1 \quad (6)$$

$$\sum_{j \in O_i} x_{ij} = 1 \quad \text{for each } i \in V \text{ where } i \neq n+1 \quad (7)$$

$$\sum_{i \in I_j} x_{ij} = 1 \quad \text{for each } j \in V \text{ where } j \neq 0 \quad (8)$$

$$x_{ij} + x_{ji} \leq 1 \quad \text{for each } (i, j) \in E \quad (9)$$

$$C_i \leq Z^0 \quad \text{for each } i \in V \quad (10)$$

$$S_i \leq Z^0 \quad \text{for each } i \in V \quad (11)$$

$$C_{n+1} \geq LB^0 \quad \text{for each } i \in V \quad (12)$$

In this model, (1) is the objective function. Aim is minimizing the makespan and in the proposed network makespan corresponds to completion time computed at the node $n+1$.

The constraint set (2) computes the completion time of task i which is equal to sum of start time and processing time of task i .

In the constraint set (3), S_{n+1} , the minimum time the node $n+1$ can start must be greater than or equal to C_i at any node. The node $n+1$ is a dummy node for which processing time is zero. It does not have any outgoing arc since it has to be scheduled after all tasks. Therefore, start time at node i is equal to the makespan and C_{n+1} is also equal to start time.

The constraint set (4) requires that start time of a task must be larger than its release time.

The constraint set (5) is a disjunctive constraint. Accordingly, if task j is scheduled immediately after task i , start time of task j must be larger than the completion time of task i .

The constraint set (6) states that if the link between nodes i and j is a directed

link emanating from i and entering into j , start time of task j must be always larger than the C_i value of node i plus the timelag, l .

The constraint sets (7) and (8) restrict the number of chosen outgoing arcs or incoming arcs by 1 except for the node 0 where the number of incoming arcs is zero and node $n + 1$ where the number of outgoing arcs is zero.

With respect to the constraint (9), for any edge (i, j) , at most one of the directed links which are from i to j and j to i is allowed to be in the optimal path from node 0 to node $n + 1$.

Due to constraint set (10), objective function value of the initial solution is an upper bound for the decision variable, C_i , for each $i \in V$.

Due to constraint set (11), objective function value of the initial solution is an upper bound for the decision variable, S_i , for each $i \in V$.

Due to constraint (12), objective function value cannot be smaller than the lower bound, LB^0 .

Technical notes for the integer programming model

In the integer programming model, the initial feasible solution, which is obtained by the polynomial algorithm given in Section 3.2 is used. Thus, in GAMS/CPLEX option file, mipstart=1 instead of the default value mipstart=0.

Related parameters and assignments are explained:

F^0 : Set of the links, (i, j) , which belong to the path generated by applying the polynomial algorithm in Section 3.2. This path represents the order of scheduled tasks.

ST_i^0 : Start time value of task i which is obtained by applying the polynomial algorithm in Section 3.2.

CT_i^0 : Completion time value of task i which is obtained by applying the polynomial algorithm in Section 3.2.

Z^0 : Objective function value of the initial solution that is obtained by applying the polynomial algorithm in Section 3.2.

x_{ij}^0 : $x_{ij}^0 = 1$ if task j is assigned immediately after task i in the initial solution.

S_i^0 : Start time at node $i \in V$ in the initial solution.

C_i^0 : Completion time at node $i \in V$ in the initial solution.

$$x_{ij}^0 = 1 \quad \text{for each } (i, j) \in F^0 \quad (13)$$

$$S_i^0 = ST_i^0 \quad \text{for each } i \in V \quad (14)$$

$$C_i^0 = CT_i^0 \quad \text{for each } i \in V \quad (15)$$

$$C_{n+1}^0 = Z^0 \quad (16)$$

The constraint set (13) shows the binary variables which are equal to 1 in the initial solution.

The constraint set (14) shows the start time values of tasks with respect to the initial solution.

The constraint set (15) shows the completion time values of tasks with respect to the initial solution.

The constraint (16) shows the objective function value obtained in the initial solution.

Also relative gap optcr and absolute gap optca are set to zero.

Example solution on an instance

In the following example, a scheduling problem instance is given, an illustration of how the given instance is transformed into a network model is provided, and the solution found by applying the proposed integer programming model is demonstrated.

Instance: Figure 3.9 shows the graph created from 3 chains with 6, 6 and 1 tasks, respectively. Let the nodes 0 and $n + 1$ be shown by s and end . According to this graph, $\{(s, 00), (00, 01), (01, 02), (02, 03), (03, 04), (04, 05), (05, end), (s, 10), (10, 11), (11, 12), (12, 13), (13, 14), (14, 15), (15, end), (s, 20), (20, end)\} \in A$ where nodes 00, 01, 02, 03, 04, 05 belong to chain 0, nodes 10, 11, 12, 13, 14, 15 belong to chain 1 and node 20 belong to chain 2. Let l be equal to 2. Also,

$$r_s = 0, r_{00} = 3, r_{01} = 7, r_{02} = 18, r_{03} = 25, r_{04} = 31, r_{05} = 37,$$

$$r_{10} = 4, r_{11} = 8, r_{12} = 16, r_{13} = 19, r_{14} = 20, r_{15} = 23,$$

$$r_{20} = 25,$$

$$r_{end} = 0;$$

$$p_s = p_{end} = 0,$$

and for all other nodes $p_i = p$ where $p = 3$.

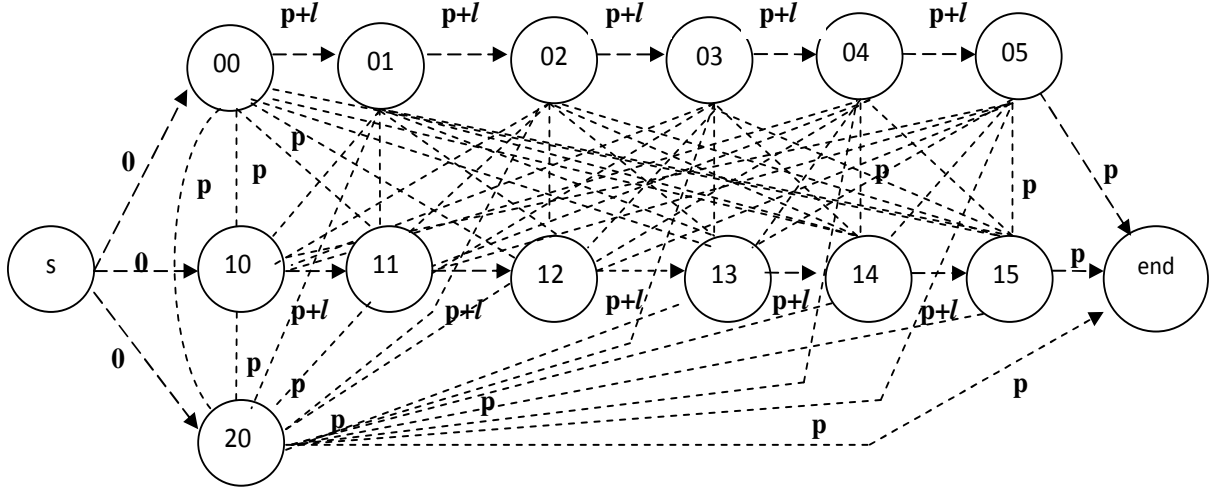


Figure 3.9: Network representation for an instance of $1|chains(l < p); p_i = p; r_i|C_{max}$

Solution of the given instance is shown in Figure 3.10.

3.5 Computational results for $1|chains(l < p); p_i = p; r_i|C_{max}$

The proposed algorithm has been coded in C and implemented on randomly generated problem instances. Solutions of the same instances generated by the proposed algorithm have been set as starting solutions for the integer programming model in GAMS. Moreover, a lower bound for the objective function value of each instance is computed in C and called by GAMS to be integrated with the integer programming model.

Polynomial algorithm and integer programming model are tested on the instances for which $n \in \{100, 500, 2500, 12500\}$ where n is the total number of tasks to be scheduled. While generating the data, for each n , numerical values of parameters of the problem are changed. These parameters are, number of chains, K , number of tasks in the longest chain, $\max_{0 \leq k < K} n_k$, release times, $r_{k,i}$, and timelag, l . When

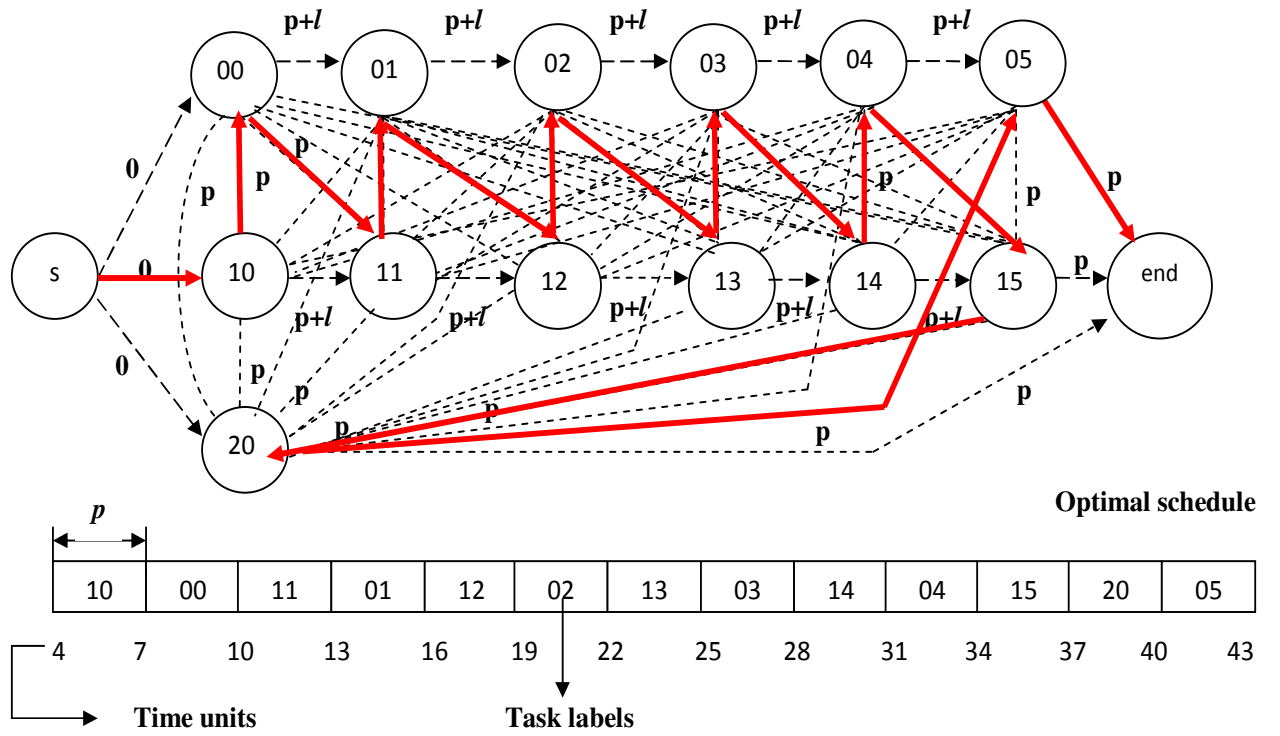


Figure 3.10: Optimal solution for the instance of $1|chains(l < p); p_i = p; r_i|C_{max}$

the problem size, n , is fixed, changing the number of chains, K , or when n and K are fixed, changing the size of the longest chain, $\max_{0 \leq k < K} n_k$, provides with different precedence graphs for the instances with the same size. Formulations and coefficients which are used to generate data are given as follows:

- (i) $K = \lfloor \kappa_1 n \rfloor$ where $\kappa_1 \in \{0.1, 0.2, 0.3\}$,
- (ii) $\max_{0 \leq k < K} n_k = \lceil n/K + \kappa_2[n - K - n/K + 1] \rceil$ where $\kappa_2 \in \{0.1, 0.2\}$,
- (iii) $r_{k,i} \in U[0, \tau_1 np]$ where $p = 20$ and $\tau_1 \in \{0.2, 0.5, 0.8, 2, 5, 8\}$,
- (iv) $l = \tau_2 p$ where $p = 20$ and $\tau_2 \in \{0.1, 0.6\}$.

Due to (i), number of chains depends on the coefficient κ_1 and problem size, n . When n is fixed, if the number of chains is large, chain lengths are short or vice versa. In (ii), number of tasks in the longest chain is computed by considering K and n values and the coefficient κ_2 . When n and K are constant, depending on κ_2 , number of tasks in the longest chain changes. In this case, number of tasks in the longest chain, $\max_{0 \leq k < K} n_k$, affects the number of tasks in the remaining chains. If $\max_{0 \leq k < K} n_k$ is large, number of tasks in the other chains is small. Number of tasks in each remaining chain is randomly generated by an algorithm which ensures that the lengths of the remaining chains are all smaller than $\max_{0 \leq k < K} n_k$ and sum of the lengths of all chains does not exceed the problem size, n . With respect to (iii), release times are uniformly distributed between 0 and $\tau_1 np$ for $\tau_1 \in \{0.2, 0.5, 0.8, 2, 5, 8\}$. When $\tau_1 \in \{0.2, 0.5, 0.8\}$, difference between the release times of tasks is smaller which means that tasks are released into the system more frequently compared to the cases where $\tau_1 \in \{2, 5, 8\}$. Equation (iv) determines the timelag values that have to be smaller than the processing time, p . For data generation, C programming language has been used.

For each combination of different values of parameters, 10 instances are generated. Thus, $3 \times 2 \times 6 \times 2 \times 10 = 720$ instances are tested for each problem size and $720 \times 4 = 2880$ instances are tested on the whole. All tests have been performed on a 2.5 GHz PC with Intel Core i5 processor and 4GB of RAM. GAMS/CPLEX solver based on the Cplex Callable Library from ILOG CPLEX 12.1.0 has been used for solving the integer

programming model.

Results are given in Tables 3.1 - 3.6. Table 3.1, Table 3.2 and Table 3.3 show the computational results for the instances with $\tau_1 \in \{0.2, 0.5, 0.8\}$, while Table 3.4, Table 3.5 and Table 3.6 show the results for the instances with $\tau_1 \in \{2, 5, 8\}$. In the tables, for each different combination of parameter values, the average solution, C_{max} over ten instances; minimum, average and maximum of the CPU times for the solution of ten instances in C; minimum, average and maximum of the CPU times for the solution of ten instances in GAMS; minimum, average and maximum gap at root and termination over ten instances in GAMS are shown. Gaps are the relative gaps which are obtained by the formulation, $(\text{best integer solution} - \text{best bound})/\text{best integer solution}$. C_{max} values obtained after running the polynomial algorithm in C are used as the initial solutions in GAMS. GAMS has been run at most 300 seconds for all instances. With this time limitation, the outputs of the polynomial algorithm were either proven to be optimal with zero gap or solved with small gaps at termination. In any case, C_{max} values which were obtained in C and GAMS were not different for the same instance. Before integrating the lower bound, LB^0 , with the integer programming model, gaps were much larger for the instances with $\tau_1 \in \{0.2, 0.5, 0.8\}$ compared to the gaps for the instances with $\tau_1 \in \{2, 5, 8\}$ since the release times with large values as in the case with $\tau_1 \in \{2, 5, 8\}$ were generating better cuts on the solution space. After introducing the lower bound, LB^0 , there was a significant decrease in the gaps for the instances with $\tau_1 \in \{0.2, 0.5, 0.8\}$. Tables 3.1 - 3.6 contain the solutions for the latter case where the lower bound, LB^0 , was introduced to the integer programming model.

Computational results for the polynomial algorithm shows that the solution time is considerably small for each instance. For the instances with $n = 12500$, when the parameters κ_1 or κ_2 took larger values, computational times were slightly larger which is an anticipated case as the algorithm is $O(n \cdot \max(K, \max_{0 \leq k < K} n_k))$ where K and $\max_{0 \leq k < K} n_k$ are dependent on the parameters κ_1 and κ_2 , respectively. In the worst cases, solution times were 3 or 4 seconds in C. However, overall, for the instances

with $n = 100$, $n = 500$ and $n = 2500$, changing the values of κ_1 or κ_2 , did not lead to notable differences in solution times. Computational results were also satisfying for integer programming model. For $n = 100$ and $n = 500$, the output of the polynomial algorithm was proven to be optimal with zero gap in most of the instances. For the remaining instances, model was interrupted after 300 seconds and the same solution as the output of the polynomial algorithm was obtained with little gaps. For $n = 500$ and $n = 2500$, instances could not be solved in GAMS due the limitations in memory.

Table 3.1: Computational results for $lchains(l < p); p_i = p, r_i | C_{max}$ for $n=100$ and $\tau_1 \in \{0.2, 0.5, 0.8\}$

n	κ ₁	κ ₂	τ ₁	τ ₂	C _{max}		Elapsed time (sec)										Initial gap for IP model						Final gap for IP model					
							O(n ³) algorithm					IP model																
avg	min	avg	max	min	avg	max	min	avg	max	min	avg	max	min	avg	max	min	avg	max										
100	0.1	0.1	0.2	0.1	2003.20	0.00	0.01	0.05	0.77	0.94	1.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00								
				0.6	2003.20	0.00	0.00	0.01	0.80	0.97	1.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00								
				0.5	2013.90	0.00	0.00	0.00	0.81	60.76	300.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.01								
			0.8	0.6	2013.90	0.01	0.01	0.02	0.80	60.71	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01								
				0.1	2029.80	0.01	0.01	0.01	0.86	210.29	300.00	0.00	0.01	0.02	0.00	0.01	0.02	0.00	0.01	0.02								
				0.6	2030.40	0.00	0.00	0.00	0.78	210.28	300.00	0.00	0.01	0.02	0.00	0.01	0.02	0.00	0.01	0.02								
		0.2	0.1	2005.10	0.00	0.01	0.01	0.83	0.91	1.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00									
			0.6	2005.40	0.00	0.00	0.00	0.78	0.89	1.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00									
			0.5	2012.80	0.01	0.01	0.02	0.79	120.54	300.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02									
			0.6	2013.90	0.00	0.00	0.00	0.84	120.61	300.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02									
			0.1	2033.10	0.00	0.00	0.00	0.97	106.88	300.00	0.00	0.01	0.03	0.00	0.01	0.03	0.00	0.01	0.03									
			0.6	2035.30	0.00	0.01	0.01	0.80	240.16	300.00	0.00	0.01	0.03	0.00	0.01	0.03	0.00	0.01	0.03									
0.2	0.1	0.2	0.1	2002.70	0.00	0.01	0.01	0.79	0.92	1.24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.6	2002.70	0.00	0.00	0.00	0.80	0.90	1.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.5	2009.60	0.00	0.00	0.00	0.81	33.62	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
		0.8	0.6	2009.60	0.01	0.01	0.01	4.74	34.47	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.1	2022.30	0.01	0.01	0.04	0.76	90.72	300.00	0.00	0.01	0.03	0.00	0.01	0.03	0.00	0.01	0.03									
			0.6	2022.30	0.00	0.00	0.00	0.75	64.17	300.00	0.00	0.01	0.03	0.00	0.01	0.03	0.00	0.01	0.03									
	0.2	0.2	0.1	2004.70	0.00	0.00	0.00	0.75	0.93	1.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.6	2004.70	0.00	0.01	0.02	0.76	0.83	1.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.5	2004.30	0.01	0.02	0.06	0.77	30.77	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
		0.8	0.6	2004.30	0.00	0.00	0.01	0.78	30.75	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.1	2034.90	0.01	0.01	0.01	0.75	240.15	300.00	0.00	0.01	0.05	0.00	0.01	0.05	0.00	0.01	0.05									
			0.6	2036.60	0.00	0.00	0.00	0.75	240.16	300.00	0.00	0.01	0.05	0.00	0.01	0.05	0.00	0.01	0.05									
0.3	0.1	0.2	0.1	2003.80	0.00	0.00	0.00	0.77	0.84	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.6	2004.80	0.00	0.01	0.01	0.78	0.88	1.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.5	2015.20	0.01	0.01	0.01	0.76	60.66	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
		0.8	0.6	2015.20	0.00	0.00	0.00	0.77	60.68	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.1	2035.00	0.01	0.02	0.03	300.00	300.00	300.00	0.00	0.01	0.02	0.00	0.01	0.02	0.00	0.01	0.02									
			0.6	2035.00	0.00	0.00	0.00	300.00	300.00	300.00	0.00	0.01	0.02	0.00	0.01	0.02	0.00	0.01	0.02									
	0.2	0.2	0.1	2003.40	0.00	0.01	0.03	0.77	0.98	1.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.6	2003.40	0.00	0.00	0.01	0.76	0.83	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00										
			0.5	2014.00	0.00	0.02	0.04	0.76	90.63	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01										
		0.8	0.6	2014.00	0.00	0.00	0.00	0.77	64.10	300.00	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.01										
			0.1	2033.60	0.00	0.01	0.02	0.78	150.43	300.00	0.00	0.01	0.05	0.00	0.01	0.05	0.00	0.01	0.05									
			0.6	2034.40	0.00	0.00	0.00	0.77	150.44	300.00	0.00	0.01	0.05	0.00	0.01	0.05	0.00	0.01	0.05									

Table 3.2: Computational results for $1|chains(l < p); p_i = p; r_i|C_{max}$ for $n=500$ and $\tau_1 \in \{0.2, 0.5, 0.8\}$

n	κ_1	κ_2	τ_1	τ_2	C_{max}	Elapsed time (sec)						Initial gap for IP model						Final gap for IP model			
						$O(n^2)$ algorithm			IP model			min	avg	max	min	avg	max	min	avg	max	
						min	avg	max	min	avg	max										
500	0.1	0.1	0.2	0.1	10005.60	0.02	0.03	0.07	17.84	18.31	18.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
					10005.60	0.00	0.00	0.00	18.19	18.67	19.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
			0.5	0.1	10012.50	0.00	0.00	0.01	17.83	102.86	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
					10013.50	0.02	0.03	0.04	17.88	103.10	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		0.8	0.1	10048.10	0.00	0.02	0.04	17.63	159.08	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01		
				10048.10	0.00	0.00	0.00	17.60	158.94	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01		
			0.2	0.1	10005.20	0.00	0.00	0.00	18.63	19.00	19.76	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
					10005.20	0.02	0.04	0.06	18.73	19.14	19.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		0.5	0.1	10015.10	0.00	0.03	0.05	18.97	131.94	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
				10015.50	0.00	0.00	0.01	18.68	131.65	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
			0.8	0.1	10054.20	0.00	0.00	0.00	19.14	234.83	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	
					10054.20	0.03	0.04	0.07	18.69	243.74	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	
500	0.2	0.1	0.2	0.1	10003.50	0.00	0.00	0.01	17.26	18.16	19.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
					10003.50	0.00	0.01	0.01	17.42	17.88	18.19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
				0.5	0.1	10017.60	0.00	0.00	0.01	18.07	75.81	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
						10017.60	0.00	0.03	0.03	18.11	74.90	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			0.8	0.1	10031.80	0.01	0.02	0.03	17.50	215.46	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	
					10031.80	0.00	0.00	0.01	17.73	20.98	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	
				0.2	0.1	10003.20	0.00	0.00	0.01	17.96	18.61	19.48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
						10003.20	0.01	0.02	0.03	18.09	18.78	19.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			0.5	0.1	10006.90	0.01	0.03	0.04	17.83	103.08	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					10007.90	0.00	0.00	0.01	17.50	102.84	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
				0.8	0.1	10052.50	0.00	0.00	0.01	18.42	243.71	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
						10052.60	0.00	0.02	0.03	18.61	243.74	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
500	0.3	0.1	0.2	0.1	10001.70	0.01	0.02	0.03	17.26	17.92	18.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
					10001.70	0.00	0.01	0.01	17.04	17.93	18.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
				0.5	0.1	10015.80	0.00	0.00	0.01	17.57	102.58	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
						10015.80	0.00	0.03	0.06	17.35	102.53	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			0.8	0.1	10048.00	0.01	0.03	0.04	18.13	243.79	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	
					10049.40	0.00	0.01	0.01	17.57	243.58	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	
				0.2	0.1	10003.30	0.00	0.00	0.01	17.19	18.14	18.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
						10003.30	0.01	0.02	0.03	17.25	18.07	19.66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			0.5	0.1	10015.50	0.01	0.03	0.04	17.27	102.47	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					10015.50	0.00	0.00	0.01	17.90	102.89	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
				0.8	0.1	10034.70	0.00	0.01	0.01	18.13	215.62	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
						10034.70	0.00	0.01	0.03	17.99	215.53	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01

Table 3.3: Computational results for $l|chains(l < p); p_i = p; r_i|C_{max}$ for $n=2500$, $n=12500$ and $\tau_1 \in \{0.2, 0.5, 0.8\}$

κ_1	κ_2	τ_1	τ_2	O(n ²) algorithm (n=2500)				O(n ²) algorithm (n=12500)				
				C _{max}		Elapsed time (sec)		C _{max}		Elapsed time (sec)		
				avg	min	avg	max	avg	min	avg	max	
0.1	0.1	0.2	0.1	50003.90	0.02	0.02	0.03	250002.10	0.91	0.94	1.03	
			0.6	50003.90	0.02	0.05	0.06	250002.10	0.89	0.90	0.92	
		0.5	0.1	50008.90	0.05	0.06	0.10	250008.10	0.95	0.96	0.97	
			0.6	50008.90	0.02	0.03	0.04	250008.10	0.91	0.92	0.95	
		0.8	0.1	50033.00	0.02	0.02	0.04	250055.70	0.93	0.99	1.06	
			0.6	50033.00	0.04	0.06	0.07	250055.70	0.94	0.96	0.98	
	0.2	0.2	0.1	50005.30	0.02	0.07	0.12	250001.60	0.99	1.02	1.09	
			0.6	50005.30	0.02	0.02	0.03	250001.60	0.97	0.99	1.05	
		0.5	0.1	50005.30	0.02	0.02	0.03	250012.60	0.98	1.01	1.06	
			0.6	50005.30	0.04	0.06	0.07	250012.60	0.97	0.99	1.01	
		0.8	0.1	50025.00	0.04	0.06	0.06	250035.90	1.02	1.06	1.17	
			0.6	50026.00	0.02	0.03	0.03	250036.10	0.96	1.02	1.05	
0.2	0.1	0.2	0.1	50001.60	0.04	0.05	0.06	250001.00	2.64	2.76	2.91	
			0.6	50001.60	0.06	0.08	0.14	250001.00	2.61	2.71	2.78	
		0.5	0.1	50013.70	0.06	0.08	0.10	250010.20	2.70	2.78	2.87	
			0.6	50013.70	0.03	0.04	0.05	250010.20	2.66	2.76	2.88	
		0.8	0.1	50031.80	0.04	0.05	0.05	250029.60	2.75	2.87	3.00	
			0.6	50032.30	0.04	0.08	0.11	250029.60	2.72	2.83	2.93	
	0.2	0.2	0.1	50004.30	0.04	0.07	0.11	250004.80	3.25	3.42	3.55	
			0.6	50004.30	0.03	0.05	0.05	250004.80	3.25	3.39	3.48	
		0.5	0.1	50011.30	0.03	0.04	0.05	250004.20	3.32	3.42	3.51	
			0.6	50011.30	0.06	0.07	0.08	250004.20	3.30	3.39	3.65	
		0.8	0.1	50023.80	0.07	0.08	0.11	250034.60	3.38	3.60	3.77	
			0.6	50024.80	0.04	0.05	0.05	250035.60	3.36	3.46	3.67	
0.3	0.1	0.2	0.1	50002.70	0.05	0.06	0.07	250002.80	2.49	2.59	2.68	
			0.6	50002.70	0.08	0.09	0.14	250002.80	2.47	2.55	2.60	
		0.5	0.1	50010.00	0.09	0.11	0.16	250008.60	2.58	2.65	2.69	
			0.6	50010.00	0.06	0.06	0.07	250008.60	2.59	2.64	2.69	
		0.8	0.1	50032.50	0.05	0.07	0.11	250039.90	2.72	2.76	2.82	
			0.6	50032.50	0.07	0.10	0.12	250039.90	2.67	2.72	2.77	
	0.2	0.2	0.1	50003.60	0.06	0.09	0.11	250003.20	2.83	2.91	2.99	
			0.6	50003.60	0.06	0.08	0.11	250003.40	2.82	2.86	2.91	
		0.5	0.1	50010.50	0.06	0.06	0.07	250007.20	2.88	2.94	3.03	
			0.6	50011.30	0.08	0.10	0.11	250007.20	2.85	2.92	2.97	
		0.8	0.1	50037.70	0.08	0.11	0.17	250020.70	2.98	3.10	3.18	
			0.6	50037.70	0.06	0.07	0.08	250020.70	3.03	3.07	3.14	

Table 3.4: Computational results for $1|chains(l < p); p_i = p; r_i|C_{max}$ for $n=100$ and $\tau_1 \in \{2, 5, 8\}$

n	κ_1	κ_2	τ_1	τ_2	C_{max}		Elapsed time (sec)						Initial gap for IP model			Final gap for IP model		
					avg	max	$O(n^2)$ algorithm			IP model			min	avg	max	min	avg	max
							min	avg	max	min	avg	max						
100	0.1	0.1	2	0.1	3985.60	0.00	0.00	0.01	1.00	3.89	20.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	3986.60	0.00	0.00	0.01	1.25	5.01	18.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				5	0.1	9862.10	0.00	0.00	0.01	0.78	1.42	2.35	0.00	0.00	0.00	0.00	0.00	0.00
			5	0.6	9862.10	0.00	0.00	0.01	0.80	1.27	2.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				8	0.1	15843.90	0.00	0.00	0.01	0.80	0.98	2.27	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	15843.90	0.00	0.00	0.02	0.81	0.93	1.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.2	2	0.1	3988.10	0.00	0.00	0.01	0.87	90.84	300.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01
				0.6	3989.30	0.00	0.00	0.01	0.81	60.89	300.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01
				5	0.1	9934.20	0.00	0.00	0.01	0.79	0.95	1.64	0.00	0.00	0.00	0.00	0.00	0.00
			8	0.6	9935.00	0.00	0.00	0.01	0.77	0.97	1.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.1	15924.80	0.00	0.00	0.01	0.80	1.10	1.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	15924.80	0.00	0.00	0.00	0.78	0.89	1.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	0.2	0.1	2	0.1	3998.20	0.00	0.00	0.01	0.73	120.71	300.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02
				0.6	3998.20	0.00	0.00	0.01	0.79	120.78	300.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02
				5	0.1	9876.20	0.00	0.00	0.01	0.72	30.73	300.00	0.00	0.00	0.00	0.00	0.00	0.00
			8	0.6	9876.60	0.00	0.00	0.01	0.7	30.74	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.1	15848.70	0.00	0.00	0.01	0.81	0.98	1.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	15848.70	0.00	0.00	0.01	0.75	0.91	1.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.2	2	0.1	3988.90	0.00	0.00	0.00	0.79	0.97	2.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	3989.90	0.00	0.00	0.01	0.8	0.94	1.85	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				5	0.1	9907.30	0.00	0.00	0.01	0.74	1.00	2.23	0.00	0.00	0.00	0.00	0.00	0.00
			8	0.6	9907.30	0.00	0.00	0.01	0.78	0.94	1.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.1	15892.40	0.00	0.00	0.00	0.74	0.90	1.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	15892.40	0.00	0.00	0.01	0.78	0.95	2.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	0.3	0.1	2	0.1	4009.00	0.00	0.00	0.01	0.8	128.57	300.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02
				0.6	4010.00	0.00	0.00	0.01	0.77	126.60	300.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02
				5	0.1	9978.40	0.00	0.00	0.01	0.8	31.25	300.00	0.00	0.00	0.00	0.00	0.00	0.00
			8	0.6	9981.60	0.00	0.00	0.01	0.78	31.18	300.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.1	15890.70	0.00	0.00	0.01	0.73	1.13	1.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	15890.70	0.00	0.00	0.01	0.77	1.20	2.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.2	2	0.1	3996.70	0.00	0.00	0.00	0.71	0.93	2.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	3997.00	0.00	0.00	0.01	0.76	0.92	2.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				5	0.1	9915.40	0.00	0.00	0.01	0.73	0.82	0.92	0.00	0.00	0.00	0.00	0.00	0.00
			8	0.6	9915.40	0.00	0.00	0.01	0.73	0.83	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.1	15895.20	0.00	0.00	0.01	0.72	0.90	1.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				0.6	15895.20	0.00	0.00	0.00	0.73	0.78	0.78	0.78	0.00	0.00	0.00	0.00	0.00	0.00

Table 3.5: Computational results for $1|chains(l < p); p_i = p; r_i|C_{max}$ for $n=500$ and $\tau_1 \in \{2, 5, 8\}$

n	κ ₁	κ ₂	τ ₁	τ ₂	C _{max}		Elapsed time (sec)						Initial gap for IP model			Final gap for IP model		
					avg	O(n ³) algorithm			IP model									
						min	avg	max	min	avg	max	min	avg	max	min	avg	max	
500	0.1	0.1	2	0.1	19971.90	0.00	0.00	0.01	17.90	91.61	300.00	0.00	0.00	0.00	0.00	0.00	0.00	
				0.6	19971.90	0.00	0.00	0.01	17.70	109.71	300.00	0.00	0.00	0.00	0.00	0.00		
			5	0.1	49930.50	0.00	0.00	0.01	18.24	34.85	180.24	0.00	0.00	0.00	0.00	0.00	0.00	
				0.6	49930.50	0.00	0.00	0.01	17.96	40.44	240.34	0.00	0.00	0.00	0.00	0.00	0.00	
			8	0.1	79779.80	0.00	0.00	0.01	17.80	46.57	300.00	0.00	0.00	0.00	0.00	0.00	0.00	
				0.6	79779.80	0.00	0.00	0.01	18.34	47.10	300.00	0.00	0.00	0.00	0.00	0.00	0.00	
		0.2	2	0.1	19991.70	0.00	0.00	0.01	19.21	132.06	300.00	0.00	0.00	0.01	0.00	0.00	0.01	
				0.6	19991.70	0.00	0.00	0.01	18.66	131.65	300.00	0.00	0.00	0.01	0.00	0.00	0.01	
			5	0.1	49928.90	0.00	0.00	0.01	18.79	40.30	230.02	0.00	0.00	0.00	0.00	0.00	0.00	
				0.6	49928.90	0.00	0.00	0.01	19.06	39.75	220.52	0.00	0.00	0.00	0.00	0.00	0.00	
			8	0.1	79916.00	0.00	0.00	0.01	18.80	96.97	300.00	0.00	0.00	0.00	0.00	0.00	0.00	
				0.6	79916.00	0.00	0.00	0.00	22.94	99.49	300.00	0.00	0.00	0.00	0.00	0.00	0.00	
0.2	0.1	2	0.1	19980.50	0.00	0.00	0.01	21.35	90.84	300.00	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	19980.50	0.00	0.00	0.01	21.03	95.28	300.00	0.00	0.00	0.00	0.00	0.00	0.00		
		5	0.1	49956.60	0.00	0.00	0.01	17.50	40.61	229.08	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	49956.60	0.00	0.00	0.01	17.62	38.10	205.43	0.00	0.00	0.00	0.00	0.00	0.00		
		8	0.1	79843.20	0.00	0.00	0.01	18.01	37.23	166.98	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	79843.20	0.00	0.00	0.01	17.57	36.01	167.43	0.00	0.00	0.00	0.00	0.00	0.00		
	0.2	2	0.1	20013.60	0.00	0.00	0.01	18.11	157.58	300.00	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	20013.60	0.00	0.00	0.01	18.34	148.03	300.00	0.00	0.00	0.00	0.00	0.00	0.00		
		5	0.1	49923.10	0.00	0.00	0.01	18.21	60.91	288.46	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	49923.10	0.00	0.00	0.01	17.89	65.86	289.93	0.00	0.00	0.00	0.00	0.00	0.00		
		8	0.1	79881.50	0.00	0.00	0.01	17.80	70.05	296.25	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	79881.50	0.00	0.00	0.01	17.96	64.68	175.90	0.00	0.00	0.00	0.00	0.00	0.00		
0.3	0.1	2	0.1	19991.10	0.00	0.00	0.01	17.28	180.90	300.00	0.00	0.00	0.01	0.00	0.00	0.01		
			0.6	19991.10	0.00	0.00	0.01	17.17	191.05	300.00	0.00	0.00	0.01	0.00	0.00	0.01		
		5	0.1	49925.40	0.00	0.00	0.01	17.44	17.89	18.53	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	49925.40	0.00	0.00	0.01	17.21	17.77	18.07	0.00	0.00	0.00	0.00	0.00	0.00		
		8	0.1	79751.30	0.00	0.00	0.01	17.56	18.32	19.48	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	79751.30	0.00	0.00	0.01	17.20	18.36	19.44	0.00	0.00	0.00	0.00	0.00	0.00		
	0.2	2	0.1	19968.10	0.00	0.00	0.01	17.76	181.41	300.00	0.00	0.00	0.01	0.00	0.00	0.01		
			0.6	19968.10	0.00	0.00	0.01	18.16	164.08	300.00	0.00	0.00	0.00	0.00	0.00	0.00		
		5	0.1	49946.50	0.00	0.00	0.01	18.02	62.54	208.79	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	49946.50	0.00	0.00	0.01	17.70	78.10	242.74	0.00	0.00	0.00	0.00	0.00	0.00		
		8	0.1	79900.60	0.00	0.00	0.01	17.70	18.23	19.13	0.00	0.00	0.00	0.00	0.00	0.00		
			0.6	79900.60	0.00	0.00	0.01	18.10	18.58	19.37	0.00	0.00	0.00	0.00	0.00	0.00		

Table 3.6: Computational results for $1|chains(l < p); p_i = p; r_i|C_{max}$ for $n=2500$, $n=12500$ and $\tau_1 \in \{2, 5, 8\}$

κ_1	κ_2	τ_1	τ_2	$O(n^2)$ algorithm ($n=2500$)				$O(n^2)$ algorithm ($n=12500$)			
				C_{max}		Elapsed time (sec)		C_{max}		Elapsed time (sec)	
				avg	max	min	avg	avg	max	min	avg
0.1	0.1	2	0.1	100001.00	0.02	0.03	0.03	494981.80	0.93	0.94	0.98
			0.6	100001.00	0.02	0.02	0.03	499981.80	0.88	0.96	0.99
			0.1	249921.50	0.02	0.02	0.03	1249861.40	0.92	0.94	0.96
		5	0.6	249921.50	0.02	0.03	0.04	1249861.40	0.88	0.96	1.00
			0.1	399792.40	0.02	0.03	0.03	1999881.20	0.91	0.93	0.95
			0.6	399792.40	0.02	0.03	0.03	1999881.20	0.93	0.98	1.02
	0.2	2	0.1	99979.00	0.02	0.03	0.03	499989.30	0.98	1.01	1.02
			0.6	99980.00	0.02	0.03	0.05	499991.30	0.98	0.99	1.01
			0.1	249922.40	0.02	0.03	0.03	1249960.40	0.97	0.99	1.01
		5	0.6	249922.40	0.02	0.03	0.03	1249960.40	0.98	0.99	1.01
			0.1	399903.10	0.02	0.03	0.03	1999981.10	0.98	0.99	1.01
			0.6	399903.10	0.05	0.06	0.08	1999981.10	0.97	0.99	1.00
0.2	0.1	2	0.1	100001.70	0.04	0.05	0.06	500007.20	2.50	2.54	2.60
			0.6	100002.70	0.04	0.06	0.09	500007.20	2.46	2.61	3.19
			0.1	249932.00	0.04	0.00	0.06	1249909.50	2.46	2.55	2.67
		5	0.6	249932.00	0.04	0.05	0.09	1249909.50	2.45	2.57	2.75
			0.1	399916.40	0.04	0.05	0.06	1999826.10	2.46	2.54	2.71
			0.6	399916.40	0.04	0.05	0.07	1999826.10	2.46	2.52	2.62
	0.2	2	0.1	99959.00	0.05	0.05	0.06	500008.20	3.05	3.20	3.46
			0.6	99959.00	0.04	0.05	0.08	500008.20	3.03	3.08	3.14
			0.1	249860.20	0.04	0.05	0.05	1249877.40	3.03	3.17	3.52
		5	0.6	249860.20	0.04	0.05	0.08	1249878.40	3.04	3.15	3.32
			0.1	399873.00	0.04	0.05	0.08	1999958.70	3.07	3.24	3.75
			0.6	399873.00	0.04	0.05	0.05	1999958.70	2.81	3.19	3.44
0.3	0.1	2	0.1	99998.40	0.07	0.08	0.09	499997.60	2.72	2.76	2.80
			0.6	99998.40	0.07	0.07	0.08	500002.10	2.72	2.76	2.83
			0.1	249898.90	0.07	0.08	0.09	1249926.30	2.70	2.74	2.77
		5	0.6	249898.90	0.06	0.08	0.10	1249926.30	2.69	2.76	2.95
			0.1	399913.80	0.06	0.08	0.09	1999942.40	2.74	2.78	2.88
			0.6	399913.80	0.06	0.08	0.09	1999942.40	2.68	2.74	2.80
	0.2	2	0.1	99973.00	0.07	0.08	0.09	500002.30	3.01	3.07	3.24
			0.6	99973.00	0.07	0.08	0.09	500002.30	3.01	3.09	3.20
			0.1	249910.80	0.07	0.08	0.09	1249972.30	2.96	3.05	3.12
		5	0.6	249910.80	0.07	0.08	0.10	1249972.30	3.04	3.11	3.18
			0.1	399901.30	0.07	0.08	0.09	1999832.40	2.92	3.05	3.41
			0.6	399901.30	0.07	0.09	0.09	1999832.40	2.99	3.10	3.22

Chapter 4

A POLYNOMIAL ALGORITHM FOR SCHEDULING TASKS WITH OUTTREE PRECEDENCE GRAPH AND TIMELAGS

In this chapter, the scheduling problem which is denoted by $1|outtree(l); p_i = p; r_i|C_{max}$ is analyzed. The problem has an open complexity status, therefore it needs consideration. As the former problem that was examined in Chapter 3, n identical tasks with constant processing times, $p_i = p$, arbitrary release times, r_i , and constant timelags, l , are scheduled on a single machine to minimize the makespan. However, in this case, the precedence graph is in the form of *outtree* where each task has at most one immediate predecessor. In this thesis, it is assumed that $l < p$ and an $O(n^3)$ algorithm that solves the problem to optimality is proposed.

4.1 Problem definition of $1|outtree(l < p); p_i = p; r_i|C_{max}$

The problem is shown by the three-field notation of $1|outtree(l); p_i = p; r_i|C_{max}$. Formally, the problem is to schedule n identical tasks with constant processing times, $p_i = p$, arbitrary release times, r_i , precedence constraints and constant timelags, l , on a single machine. The precedence relations between the tasks are represented by the outtree. In this graph, each task has at most one immediate predecessor. Timelags exist depending on the precedence structure and scheduling order of tasks. To give an example, if there is a precedence link between two tasks on the given graph and if those tasks are to be assigned into the schedule successively, start time of the successor task is delayed. Tasks are not allowed to be processed on the machine simultaneously. Preemption is forbidden. The makespan, C_{max} , is minimized. In this study, timelag is

assumed to be strictly less than the processing time and a polynomial time algorithm is developed. It is also shown that the proposed algorithm finds the optimal solution.

The notation of $1|outtree(l < p); p_i = p; r_i|C_{max}$ is adjusted by a preliminary work which is described below:

Preliminary processing

Figure 4.1.a demonstrates an example of an *outtree*. In the graph the node at the tail of an arrow is the predecessor of the node at the head of that arrow. The graph is taken and transformed by the following procedures:

Grouping : The nodes -or the tasks- are separated into groups with respect to their immediate predecessors on the given graph instance. That is, any group of tasks is composed of all immediate successors of one common task. Also, dummy groups are created for the nodes in the end leaves which in fact have no successors (Figure 4.1.b).

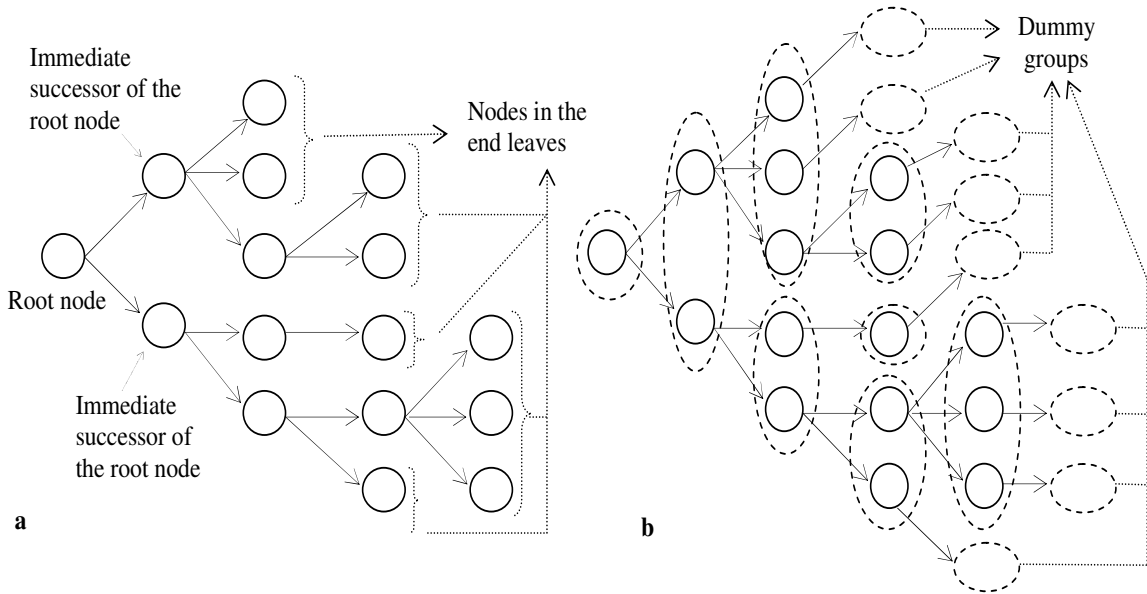


Figure 4.1: Precedence graph and grouping for $1|outtree(l < p); p_i = p; r_i|C_{max}$

Labeling the groups : Both dummy and real groups are labeled. Labeling starts by the number zero at the root group, continues by increasing that number by one following the direction from up to down and left to right and ends by the number

equal to the total number of tasks at the last group. Figure 4.2 shows how the groups are labeled.

Labeling the tasks : Tasks labeling is the same as group labeling except that the task numbers are increased by one instead of the group numbers, dummy groups are skipped and labeling is stopped at the largest number strictly smaller than total number of tasks. Figure 4.2 shows how the tasks are labeled.

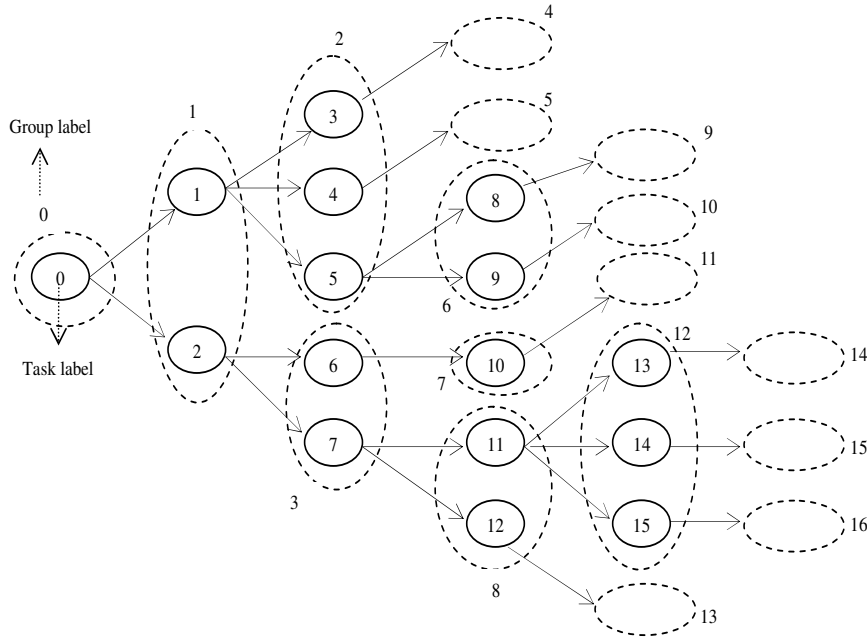


Figure 4.2: Labeling the groups and tasks for $1|outtree(l < p); p_i = p; r_i|C_{max}$

Leveling : Groups are leveled in accordance with their priorities. The more the groups are urgent the larger priority values they take. Figure 4.3 gives an example. The idea comes from Hu(1961), who applied a similar leveling approach for the precedence graph in the form of an intree.

Notation after preliminary processing

Let K be the total number of groups, k be the index of any group and n_k be the number of tasks in group k where $k = 0, \dots, K - 1$. Let i be the index of any task in group k . Then, $r_{k,i}$, $k = 0, \dots, K - 1$, $i = 0, \dots, n_k - 1$, denotes the release time of i^{th}

by $lsg = t_{k,i} + 1$. When k is dummy, $r_{k,i} = +\infty$, $n_k = 1$, $pr_{k,i} = 0$, $t_{k,i} = +\infty$.

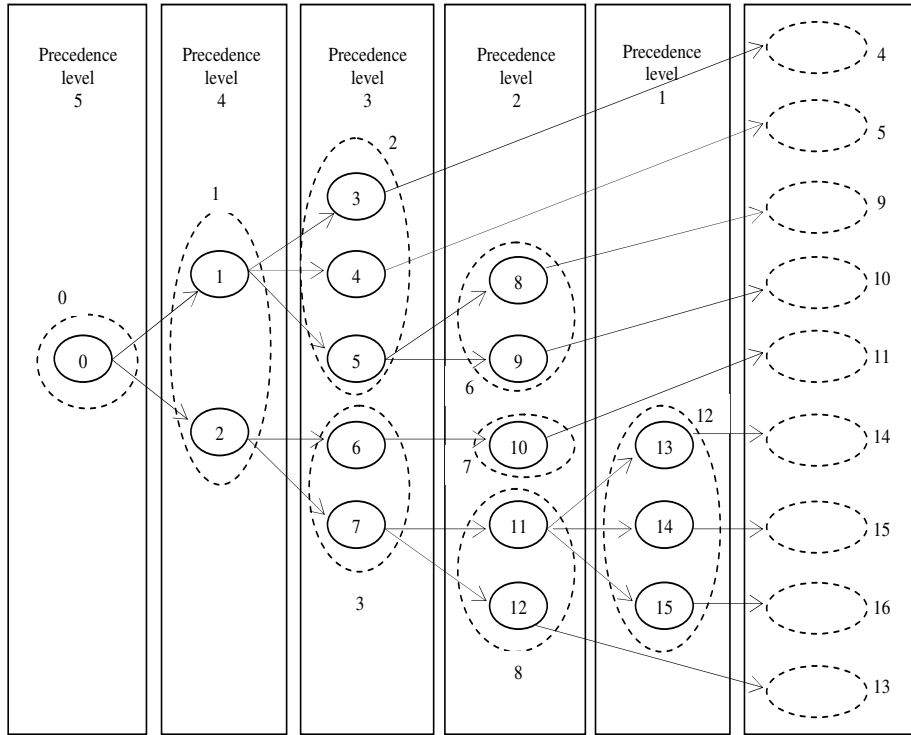


Figure 4.3: Leveling for $1|outtree(l < p); p_i = p; r_i|C_{max}$

4.2 Optimality conditions for $1|_{\text{outtree}(l < p)}; p_i = p; r_i|C_{\max}$

Proposed algorithm decomposes the outtree into chains with respect to the release times of tasks and their relation with p and l at any iteration. According to the precedence restrictions and release time constraints for the tasks which do not belong

to the chains, chains are partially updated. The best partial sequence for a group of tasks in different chains is obtained and the task that leads to the best partial sequence is assigned to the current position of the schedule. Earliest start times for candidate tasks are updated based on the completion time of the task in the former position, release time and timelag constraints of candidates for current position. Procedure is repeated until all tasks are scheduled. In this method, decomposition approach helps to adapt the optimality conditions proposed in Chapter 3 for $1|chains(l < p); p_i = p; r_i|C_{max}$ to the solution procedure of $1|outtree(l < p); p_i = p; r_i|C_{max}$. The outtree precedence graph has a more complicated structure compared to chains. However, in this chapter, hidden chains with desired conditions are created from the outtree graph and optimality conditions are restructured with respect to the properties of the scheduling problem $1|outtree(l < p); p_i = p; r_i|C_{max}$.

In the proposed algorithm, the following variables are updated at the beginning of each iteration:

ct : The completion time of the task that was chosen to be processed in the former iteration. It is equal to $st + p$ where st is the start time of the formerly scheduled task.

Ω : The index of the group $k \in [0, K - 1]$, from which a task was chosen to be scheduled at the former iteration.

δ : The index of the formerly chosen task $i \in [0, n_\Omega - 1]$ in group Ω .

lsg^* : The label of the group containing the immediate successors of formerly chosen task, $t_{\Omega, \delta}$. It is obtained by the equation, $lsg^* = t_{\Omega, \delta} + 1$.

$r_{\Omega, \delta}$: The release time of $t_{\Omega, \delta}$. It is replaced by $+\infty$ to prevent the task $t_{\Omega, \delta}$ from being a candidate task in the remaining iterations.

$m_{lsg^*, i}$: The earliest start time of task i in group lsg^* . $m_{lsg^*, i} = \max(ct + l, r_{lsg^*, i})$ for each $i = 0, \dots, n_{lsg^*} - 1$.

$pr_{k, i}^*$: Updated priority degree of task i in group k . $pr_{lsg^*, i}^* = +\infty$ for each $i = 0, \dots, n_{lsg^*} - 1$. When $k = \Omega$ & $i \neq \delta$ or $k \neq \Omega$ & $k \neq lsg^*$, if $r_{k, i} = +\infty$ the label of the successor group, $lsg = t_{k, i} + 1$, is obtained and precedence levels of the tasks in

the successor group are replaced by $pr_{lsg,i}^* = +\infty$ for $i \in [0, n_{lsg} - 1]$. If any task i in group lsg has the release time $r_{lsg,i} = +\infty$, the priority value is replaced by $pr_{k,i}^* = 0$.

$max_{0 \leq k < K, 0 \leq i < n_k} pr_{k,i}^*$: The maximum of updated priority degrees.

$r_{k,i}^*$: Updated release time value of task i in group k where $k = 0, \dots, K - 1$ and $i = 0, \dots, n_k - 1$. For each $i = 0, \dots, n_{lsg^*} - 1$, $r_{lsg^*,i}^* = m_{lsg^*,i}$. If $k = \Omega$ & $i \neq \delta$ or $k \neq \Omega$ & $k \neq lsg^*$ and $pr_{k,i}^* = max_{0 \leq k < K, 0 \leq i < n_k} pr_{k,i}^*$, $r_{k,i}^* = max(ct, r_{k,i})$. If $pr_{k,i}^* \neq max_{0 \leq k < K, 0 \leq i < n_k} pr_{k,i}^*$, then $r_{k,i}^* = +\infty$.

$r_{\beta,\alpha}^*$: The minimum of earliest start times. That is $r_{\beta,\alpha}^* = min_{0 \leq k < K, 0 \leq i < n_k} r_{k,i}^*$. If there are several $r_{k,i}^*$ which are equal to $min_{0 \leq k < K, 0 \leq i < n_k} r_{k,i}^*$, β and α represent the task which has the largest $ns_{k,i}$.

$r_{\zeta,\gamma}^*$: The minimum of earliest start times obtained after excluding $t_{\beta,\alpha}$. It is equal to the minimum of $min_{0 \leq k < K, 0 \leq i < n_k} r_{k \neq \beta, i}^*$ and $min_{0 \leq i < n_k} r_{\beta, i \neq \alpha}$. If several tasks have the earliest start times equal to $r_{\zeta,\gamma}^*$, the task with the largest $ns_{k,i}$ among them is represented by ζ and γ . Note that ζ may coincide with β as long as $i \neq \alpha$ and γ may coincide with α as long as $k \neq \beta$.

$r_{\theta,\eta}^*$: The minimum of earliest start times found by excluding $t_{\beta,\alpha}$ and $t_{\zeta,\gamma}$. θ and η are the group and task labels of the task with $r_{\theta,\eta}^*$. The size of $ns_{k,i}$ is used again to determine θ and η if there is an equality. $r_{\theta,\eta}^*$ is computed by taking the minimum of following values:

$$\begin{aligned} & min_{0 \leq k < K, 0 \leq i < n_k} r_{k \neq \beta \& k \neq \zeta, i}^*; \\ & min[min_{0 \leq i < n_\beta} r_{\beta, i \neq \alpha}^*, min_{0 \leq i < n_\zeta} r_{\zeta, i \neq \gamma}^*] \text{ if } \beta \neq \zeta; \\ & min_{0 \leq i < n_\zeta} r_{\zeta, i \neq \gamma \& i \neq \alpha}^* \text{ if } \beta = \zeta. \end{aligned}$$

$r_{\Upsilon,\kappa}^*$: The minimum of earliest start times found by excluding $t_{\beta,\alpha}$, $t_{\zeta,\gamma}$ and $t_{\theta,\eta}$. Υ and κ denote the task with $r_{\Upsilon,\kappa}^*$. Also, $ns_{k,i}$ values are compared to choose Υ and κ . $r_{\Upsilon,\kappa}^*$ is computed by taking the minimum of following values:

$$\begin{aligned} & min_{0 \leq k < K, 0 \leq i < n_k} r_{k \neq \beta \& k \neq \zeta \& k \neq \theta, i}^*; \\ & min[min_{0 \leq i < n_\beta} r_{\beta, i \neq \alpha}^*, min_{0 \leq i < n_\zeta} r_{\zeta, i \neq \gamma}^*, min_{0 \leq i < n_\theta} r_{\theta, i \neq \eta}^*] \text{ if } \beta \neq \zeta, \zeta \neq \theta, \theta \neq \beta; \\ & min[min_{0 \leq i < n_\zeta} r_{\zeta, i \neq \gamma \& i \neq \alpha}^*, min_{0 \leq i < n_\theta} r_{\theta, i \neq \eta}^*] \text{ if } \beta = \zeta \neq \theta; \\ & min[min_{0 \leq i < n_\zeta} r_{\zeta, i \neq \gamma \& i \neq \eta}^*, min_{0 \leq i < n_\beta} r_{\beta, i \neq \alpha}^*] \text{ if } \beta \neq \zeta = \theta; \end{aligned}$$

$\min[\min_{0 \leq i < n_\beta} r_{\beta, i \neq \alpha \& i \neq \eta}^*, \min_{0 \leq i < n_\zeta} r_{\beta, i \neq \gamma}^*]$ if $\beta = \theta \neq \zeta$;

$\min_{0 \leq i < n_\beta} r_{\beta, i \neq \alpha \& i \neq \gamma \& i \neq \eta}^*$ if $\beta = \theta = \zeta$.

hc_j^β : The j^{th} value in the array of the required information for hidden chain β . The array is created by assigning $r_{\beta, \alpha}^*$ as the first entry, obtaining the label of successor group $lsg = t_{\beta, \alpha} + 1$, assigning $\min_{0 \leq i < n_{lsg}} r_{lsg, i}$, as the second entry, obtaining the label of successor group again and repeating the process till $pr_{lsg, i}$ becomes equal to 0 for any $i \in [0, n_{lsg} - 1]$. lsg is dynamically updated during this process. When $pr_{lsg, i} = 0$, the loop is broken and size of the array is set. Size of the array is shown by sz^β .

hc_j^ζ : The j^{th} value in the array of the required information for hidden chain ζ . The array is created by assigning $r_{\zeta, \gamma}^*$ as the first entry, obtaining the label of successor group $lsg = t_{\zeta, \gamma} + 1$ and continuing by the same process as for hc_j^β . Size of the array is shown by sz^ζ .

ft_z^β : The z^{th} value in the first array of free tasks around the hidden chain β . While generating the hc_j^β values, for $j = 1, \dots, sz^\beta - 1$ and $z = j - 1$, ft_z^β takes the minimum original release time in the associated group lsg after excluding the task with $\min_{0 \leq i < n_{lsg}} r_{lsg, i}$. Size of the array is equal to $sz^\beta - 1$. The tasks with the release times, ft_z^β , $z = 0, \dots, sz^\beta - 2$, do not form a chain.

fts_z^β : The z^{th} value in the second array of free tasks around the hidden chain β . While generating the hc_j^β values, for $j = 1, \dots, sz^\beta - 1$ and $z = j - 1$, fts_z^β takes the minimum original release time in the associated group lsg after excluding the tasks with $\min_{0 \leq i < n_{lsg}} r_{lsg, i}$ and with ft_z^β . Size of the array is equal to $sz^\beta - 1$. The tasks with the release times, fts_z^β , $z = 0, \dots, sz^\beta - 2$, do not form a chain.

lk_z^β : The group label of task z with the release time value ft_z^β where $z = 0, \dots, sz^\beta - 2$.

li_z^β : The task index, i , of the task z with the release time, ft_z^β , and group label, lk_z^β . That is, if ft_z^β corresponds to $r_{k, i}$, lk_z^β shows the group label k and li_z^β shows the task index i of group k .

ft_z^ζ : The z^{th} value in the first array of free tasks around the hidden chain ζ .

While generating the hc_j^ζ values, for $j = 1, \dots, sz^\zeta - 1$ and $z = j - 1$, ft_z^ζ takes the minimum original release time in the associated group lsg after excluding the task with $\min_{0 \leq i < n_{lsg}} r_{lsg,i}$. Size of the array is equal to $sz^\zeta - 1$. The tasks with the release times, ft_z^ζ , $z = 0, \dots, sz^\zeta - 2$, do not form a chain.

fts_z^ζ : The z^{th} value in the second array of free tasks around the hidden chain ζ . While generating the hc_j^ζ values, for $j = 1, \dots, sz^\zeta - 1$ and $z = j - 1$, fts_z^ζ takes the minimum original release time in the associated group lsg after excluding the tasks with $\min_{0 \leq i < n_{lsg}} r_{lsg,i}$ and with ft_z^ζ . Size of the array is equal to $sz^\zeta - 1$. The tasks with the release times, fts_z^ζ , $z = 0, \dots, sz^\zeta - 2$, do not form a chain.

lk_z^ζ : The group label of task z with the release time value ft_z^ζ where $z = 0, \dots, sz^\zeta - 2$.

li_z^ζ : The task index, i , of the task z with the release time, ft_z^ζ , and group label, lk_z^ζ . That is, if ft_z^ζ corresponds to $r_{k,i}$, lk_z^ζ shows the group label k and li_z^ζ shows the task index i of group k .

stc_j^ζ : For $j = 0, \dots, sz^\zeta - 2$ and $stc_0^\zeta = hc_0^\zeta$, $stc_{j+1}^\zeta = \max(stc_j^\zeta + 2p, hc_{j+1}^\zeta)$.

stc_j^β : For $j = 0, \dots, sz^\beta - 2$ and $stc_0^\beta = hc_0^\beta$, $stc_{j+1}^\beta = \max(stc_j^\beta + 2p, hc_{j+1}^\beta)$.

$chc^\beta, \theta_1, \eta_1, r_{\theta_1, \eta_1}^*$: Let the initial values of θ_1 , η_1 and r_{θ_1, η_1}^* be equal to θ , η and $r_{\theta, \eta}^*$. chc^β is the counter of the following inequality till it becomes invalid: $r_{\theta_1, \eta_1}^* > stc_{j+1}^\beta = \max(stc_j^\beta + 2p, hc_{j+1}^\beta)$ where $j = 0, \dots, sz^\beta - 2$ and r_{θ_1, η_1}^* is replaced by $r_{\theta_1, \eta_1}^* = \min[r_{\theta, \eta}^*, \min_{0 \leq z < j} ft_z^\beta, \min_{0 \leq z < j} ft_z^\zeta]$ together with the index couples θ_1 and η_1 which may correspond to θ and η , lk_z^β and li_z^β or lk_z^ζ and li_z^ζ iteratively. Note that for $j = 0$, $r_{\theta_1, \eta_1}^* = r_{\theta, \eta}^*$ since $\min_{0 \leq z < 0} ft_z^\beta$ and $\min_{0 \leq z < 0} ft_z^\zeta$ do not exist. The final updated value for r_{θ_1, η_1}^* that is found by the counting process is returned if r_{θ_1, η_1}^* is called.

$chc^\zeta, \theta_2, \eta_2, r_{\theta_2, \eta_2}^*$: Let the initial values of θ_2 , η_2 and r_{θ_2, η_2}^* be equal to θ , η and $r_{\theta, \eta}^*$. chc^ζ is the counter of the following inequality till it becomes invalid: $r_{\theta_2, \eta_2}^* > stc_{j+1}^\zeta = \max(stc_j^\zeta + 2p, hc_{j+1}^\zeta)$ where $j = 0, \dots, sz^\zeta - 2$ and r_{θ_2, η_2}^* is replaced by $r_{\theta_2, \eta_2}^* = \min[r_{\theta, \eta}^*, \min_{0 \leq z < j} ft_z^\beta, \min_{0 \leq z < j} ft_z^\zeta]$ together with the index couples θ_2 and η_2 which may correspond to θ and η , lk_z^β and li_z^β or lk_z^ζ and li_z^ζ iteratively. Note that

for $j = 0$, $r_{\theta_2, \eta_2}^* = r_{\theta, \eta}^*$ since $\min_{0 \leq z < 0} ft_z^\beta$ and $\min_{0 \leq z < 0} ft_z^\zeta$ do not exist. The final updated value for r_{θ_2, η_2}^* that is found by the counting process is returned if r_{θ_2, η_2}^* is called.

$chc_1^{\beta\zeta}$: The counter of the situations satisfying the inequality $\min[\max(stc_{chc^\beta}^\beta + 2p, r_{\theta_1, \eta_1}^*), stc_{j+1}^\beta] < stc_{j+1}^\zeta < stc_j^\zeta + 2p + l$ where $j = 0, \dots, chc^\beta$. Note that $chc_1^{\beta\zeta} = chc^\beta + 1$ when the given inequality is true for all $j = 0, \dots, chc^\beta$.

$chc_2^{\beta\zeta}$: The counter of the situations satisfying the inequality $\max[\min[\max(stc_{chc^\beta}^\beta + 2p, r_{\theta_1, \eta_1}^*), stc_{j+1}^\beta], stc_{j+1}^\zeta] < stc_j^\zeta + 2p + l$ where $j = 0, \dots, chc^\beta$. Note that $chc_2^{\beta\zeta} = chc^\beta + 1$ when the given inequality is true for all $j = 0, \dots, chc^\beta$.

$chc_1^{\zeta\beta}$: The counter of the situations satisfying the inequality $\min[\max(stc_{chc^\zeta}^\zeta + 2p, r_{\theta_2, \eta_2}^*), stc_{j+1}^\zeta] < stc_{j+1}^\beta < stc_j^\beta + 2p + l$ where $j = 0, \dots, chc^\zeta$. Note that $chc_1^{\zeta\beta} = chc^\zeta + 1$ when the given inequality is true for all $j = 0, \dots, chc^\zeta$.

$chc_2^{\zeta\beta}$: The counter of the situations satisfying the inequality $\max[\min[\max(stc_{chc^\zeta}^\zeta + 2p, r_{\theta_2, \eta_2}^*), stc_{j+1}^\zeta], stc_{j+1}^\beta] < stc_j^\beta + 2p + l$ where $j = 0, \dots, chc^\zeta$. Note that $chc_2^{\zeta\beta} = chc^\zeta + 1$ when the given inequality is true for all $j = 0, \dots, chc^\zeta$.

$control^\zeta$: The first difference $(stc_{j+1}^\zeta + 2p) - (stc_{j+1}^\beta + 2p)$ that meets the condition $(stc_{j+1}^\zeta + 2p) - (stc_{j+1}^\beta + 2p) \leq -l$ or $(stc_{j+1}^\zeta + 2p) - (stc_{j+1}^\beta + 2p) \geq l$ for $j = 0, \dots, chc^\beta - 1$.

$control^\beta$: The first difference $(stc_{j+1}^\beta + 2p) - (stc_{j+1}^\zeta + 2p)$ that meets the condition $(stc_{j+1}^\beta + 2p) - (stc_{j+1}^\zeta + 2p) \leq -l$ or $(stc_{j+1}^\beta + 2p) - (stc_{j+1}^\zeta + 2p) \geq l$ for $j = 0, \dots, chc^\zeta - 1$.

ϕ_1 : The label of successor group for t_{θ_1, η_1} that is obtained by $\phi_1 = t_{\theta_1, \eta_1} + 1$.

$\min_{0 \leq i < n_{\phi_1}} r_{\phi_1, i}$: The minimum original release time in group ϕ_1 . If $\min_{0 \leq i < n_{\phi_1}} r_{\phi_1, i} < r_{\theta_1, \eta_1}^*$, it is replaced by $\min_{0 \leq i < n_{\phi_1}} r_{\phi_1, i} = r_{\theta_1, \eta_1}^*$.

ϕ_2 : The label of successor group for t_{θ_2, η_2} that is obtained by $\phi_2 = t_{\theta_2, \eta_2} + 1$.

$\min_{0 \leq i < n_{\phi_2}} r_{\phi_2, i}$: The minimum original release time in group ϕ_2 . If $\min_{0 \leq i < n_{\phi_2}} r_{\phi_2, i} < r_{\theta_2, \eta_2}^*$, it is replaced by $\min_{0 \leq i < n_{\phi_2}} r_{\phi_2, i} = r_{\theta_2, \eta_2}^*$.

ω_1 : The task with the earliest start time est_{ω_1} which is the minimum of following values:

$$\begin{aligned}
 &hc_{chc^\beta+1}^\beta, \\
 &r_{\Upsilon,\kappa}^*, \\
 &r_{\theta,\eta}^* \text{ if } \theta \neq \theta_1, \eta \neq \eta_1, \\
 &\min_{0 \leq z < chc^\beta} ft_z^\beta \text{ for } lk_z^\beta \neq \theta_1, li_z^\beta \neq \eta_1, \\
 &\min_{0 \leq z < chc^\beta} ft_z^\zeta \text{ for } lk_z^\zeta \neq \theta_1, li_z^\zeta \neq \eta_1, \\
 &\min_{0 \leq z < chc^\beta} fts_z^\beta, \\
 &\min_{0 \leq z < chc^\beta} fts_z^\zeta, \\
 &\min_{0 \leq i < n_{\phi_1}} r_{\phi_1,i}, \\
 &\max[\max(stc_{chc^\beta}^\beta + 2p, r_{\theta_1,\eta_1}^*) + 2p + l; hc_{chc^\beta+2}^\zeta] \\
 &ft_{chc^\beta}^\zeta
 \end{aligned}$$

ω_2 : The task with the earliest start time est_{ω_2} which is the minimum of following values:

$$\begin{aligned}
 &hc_{chc^\zeta+1}^\zeta, \\
 &r_{\Upsilon,\kappa}^*, \\
 &r_{\theta,\eta}^* \text{ if } \theta \neq \theta_2, \eta \neq \eta_2, \\
 &\min_{0 \leq z < chc^\zeta} ft_z^\beta \text{ for } lk_z^\beta \neq \theta_2, li_z^\beta \neq \eta_2, \\
 &\min_{0 \leq z < chc^\zeta} ft_z^\zeta \text{ for } lk_z^\zeta \neq \theta_2, li_z^\zeta \neq \eta_2, \\
 &\min_{0 \leq z < chc^\zeta} fts_z^\beta, \\
 &\min_{0 \leq z < chc^\zeta} fts_z^\zeta, \\
 &\min_{0 \leq i < n_{\phi_2}} r_{\phi_2,i}, \\
 &\max[\max(stc_{chc^\zeta}^\zeta + 2p, r_{\theta_2,\eta_2}^*) + 2p + l; hc_{chc^\zeta+2}^\beta] \\
 &ft_{chc^\zeta}^\beta
 \end{aligned}$$

At any iteration, using the updated values of the variables, several conditions are checked. These conditions are necessary to define some propositions that involve the optimal selection strategies.

Optimality conditions : The conditions checked at each iteration are as follows:

CASE I.

$$(i) \ r_{\zeta, \gamma}^* - r_{\beta, \alpha}^* < l.$$

CASE I - Condition I.

$$(i) \ chc^\zeta \neq 0,$$

$$(ii) \ chc_1^{\zeta\beta} = chc^\zeta + 1,$$

$$(iii) \ max_{0 \leq j \leq chc^\zeta - 1} |(stc_{j+1}^\beta + 2p) - (stc_{j+1}^\zeta + 2p)| < l,$$

$$(iv) \ est_{\omega_2} < stc_{chc^\zeta + 1}^\beta + 2p.$$

CASE I - Condition II.

$$(i) \ chc^\zeta \neq 0,$$

$$(ii) \ chc_1^{\zeta\beta} \neq chc^\zeta + 1,$$

$$(iii) \ max_{0 \leq j \leq chc^\zeta - 1} |(stc_{j+1}^\beta + 2p) - (stc_{j+1}^\zeta + 2p)| < l,$$

$$(iv) \ chc_2^{\zeta\beta} = chc^\zeta + 1,$$

$$(v) \ stc_{chc^\zeta}^\beta < stc_{chc^\zeta}^\zeta,$$

$$(vi) \ max(stc_{chc^\zeta}^\zeta + 2p, r_{\theta_2, \eta_2}^*) \leq r_{\theta_2, \eta_2}^* < stc_{chc^\zeta + 1}^\beta,$$

$$(vii) \ est_{\omega_2} < stc_{chc^\zeta + 1}^\beta + 2p.$$

CASE I - Condition III.

$$(i) \ chc^\zeta = 0,$$

$$(ii) \ max(stc_0^\zeta + 2p, r_{\theta_2, \eta_2}^*) \leq r_{\theta_2, \eta_2}^* < stc_1^\beta,$$

$$(iii) \ est_{\omega_2} < stc_1^\beta + 2p.$$

CASE I - Condition IV.

$$(i) \ chc^\beta \neq 0,$$

$$(ii) \ chc_1^{\beta\zeta} = chc^\beta + 1,$$

$$(iii) \ \max_{0 \leq j \leq chc^\beta - 1} |(stc_{j+1}^\zeta + 2p) - (stc_{j+1}^\beta + 2p)| < l,$$

$$(iv) \ est_{\omega_1} < stc_{chc^\beta + 1}^\zeta + 2p.$$

CASE I - Condition V.

$$(i) \ chc^\beta \neq 0,$$

$$(ii) \ chc_1^{\beta\zeta} \neq chc^\beta + 1,$$

$$(iii) \ \max_{0 \leq j \leq chc^\beta - 1} |(stc_{j+1}^\zeta + 2p) - (stc_{j+1}^\beta + 2p)| < l,$$

$$(iv) \ chc_2^{\beta\zeta} = chc^\beta + 1,$$

$$(v) \ stc_{chc^\beta}^\zeta < stc_{chc^\beta}^\beta,$$

$$(vi) \ \max(stc_{chc^\beta}^\beta + 2p, r_{\theta_1, \eta_1}^*) \leq r_{\theta_1, \eta_1}^* < stc_{chc^\beta + 1}^\zeta,$$

$$(vii) \ est_{\omega_1} < stc_{chc^\beta + 1}^\zeta + 2p.$$

CASE I - Condition VI.

$$(i) \ chc^\beta = 0,$$

$$(ii) \ \max(stc_0^\beta + 2p, r_{\theta_1, \eta_1}^*) < stc_1^\zeta,$$

$$(iii) \ est_{\omega_1} < stc_1^\zeta + 2p.$$

CASE II.

(i) $r_{\zeta,\gamma}^* - r_{\beta,\alpha}^* \geq l$.

Proposition 1. If all the restrictions defined by CASE 1 and CASE 1 - Condition 1 are satisfied, optimal solution is obtained by assigning the task $t_{\zeta,\gamma}$ to the current position of the schedule at time $r_{\zeta,\gamma}^*$.

Proof. Suppose there is a group of tasks which meets the constraints in CASE 1, CASE 1 - Condition 1. For a contradiction, assume that optimal schedule is found by assigning the task $t_{\beta,\alpha}$ at time $r_{\beta,\alpha}^*$ into the current position. Then, with respect to given restrictions, if the best partial sequence σ^1 starting at time $r_{\beta,\alpha}^*$ is constructed, $r_{\zeta,\gamma}^* + (2 * chc^\zeta + 5)p < C_{max}(\sigma^1) < r_{\beta,\alpha}^* + (2 * chc^\zeta + 5)p + (chc^\zeta + 1)l$. Lower and upper bounds on the makespan exist because of the given time intervals for earliest start times. The lower bound is a cut on the solution for the best order when the earliest start times take the smallest possible values and the upper bound is the cut on the solution for the best order when the earliest start times take the largest possible values. By swapping the tasks $t_{\beta,\alpha}$ and $t_{\zeta,\gamma}$ in sequence σ^1 and subject to given constraints scheduling the remaining tasks in the best order, a new partial sequence σ^2 starting at time $r_{\zeta,\gamma}^*$ and having the makespan within the interval $r_{\zeta,\gamma}^* + (2 * chc^\zeta + 5)p \leq C_{max}(\sigma^2) < r_{\beta,\alpha}^* + (2 * chc^\zeta + 5)p + (chc^\zeta + 1)l$ is found. Since there exists a sequence σ^2 with $C_{max}(\sigma^2) < C_{max}(\sigma^1)$ when $C_{max}(\sigma^2) = r_{\zeta,\gamma}^* + (2 * chc^\zeta + 5)p$ and $C_{max}(\sigma^1) > r_{\zeta,\gamma}^* + (2 * chc^\zeta + 5)p$ assigning the task $t_{\beta,\alpha}$ into the current position is not optimal. Proof is valid for any value chc^ζ takes.

Example 1.

Let c_j^β , $j = 0, \dots, sz^\beta - 1$, and c_j^ζ , $j = 0, \dots, sz^\zeta - 1$, denote the tasks in the hidden chains β and ζ and suppose that $chc^\zeta = 4$. This means that $stc_4^\zeta < r_{\theta_2,\eta_2}^* = \min[r_{\theta,\eta}^*, \min_{0 \leq z < 4} ft_z^\beta, \min_{0 \leq z < 4} ft_z^\zeta] \leq stc_5^\zeta$. In CASE 1 - Condition 1, $chc_1^{\zeta\beta} = chc^\zeta + 1 = 5$. Therefore, $stc_1^\zeta < stc_1^\beta < stc_0^\beta + 2p + l$, $stc_2^\zeta < stc_2^\beta < stc_1^\beta + 2p + l$, $stc_3^\zeta < stc_3^\beta < stc_2^\beta + 2p + l$, $stc_4^\zeta < stc_4^\beta < stc_3^\beta + 2p + l$ and $\max(stc_4^\zeta + 2p, r_{\theta_2,\eta_2}^*) < stc_5^\beta < stc_4^\beta + 2p + l$. Additionally, $est_{\omega_2} < stc_5^\beta + 2p$ where est_{ω_2} is the minimum of the following

values: hc_5^ζ , $r_{\Upsilon,\kappa}^*$, $r_{\theta,\eta}^*$ if $\theta \neq \theta_2$ and $\eta \neq \eta_2$, $\min_{0 \leq z < 4} ft_z^\beta$ for $lk_z^\beta \neq \theta_2$ and $li_z^\beta \neq \eta_2$, $\min_{0 \leq z < 4} ft_z^\zeta$ for $lk_z^\zeta \neq \theta_2$ and $li_z^\zeta \neq \eta_2$, $\min_{0 \leq z < 4} fts_z^\beta$, $\min_{0 \leq z < 4} fts_z^\zeta$, $\min_{0 \leq i < n_{\phi_2}} r_{\phi_2,i}$, $\max[\max(stc_4^\zeta + 2p, r_{\theta_2,\eta_2}^*) + 2p + l; hc_6^\beta]$, ft_4^β . Regarding these restrictions, the tightest partial sequence σ^1 that starts by $t_{\beta,\alpha}$, the tightest partial sequence σ^2 that starts by $t_{\zeta,\gamma}$, the longest partial sequence σ^1 that starts by $t_{\beta,\alpha}$ and the longest partial sequence σ^2 that starts by $t_{\zeta,\gamma}$ are shown in Figure 4.4.a, b, c and d, respectively. For the alternative partial sequences, the makespan remains in the following intervals: $r_{\zeta,\gamma}^* + 13p < C_{\max}(\sigma^1) < r_{\beta,\alpha}^* + 13p + 5l$ and $r_{\zeta,\gamma}^* + 13p \leq C_{\max}(\sigma^2) < r_{\beta,\alpha}^* + 13p + 5l$. Therefore, optimal schedule continues with task $t_{\zeta,\gamma}$ starting at time $r_{\zeta,\gamma}^*$ after the completion of previously assigned task.

Proposition 2. If the constraints in CASE 1 and CASE 1 - Condition 2 are met, optimal solution is obtained by assigning the task $t_{\zeta,\gamma}$ to the current position of the schedule at time $r_{\zeta,\gamma}^*$.

Proof. Suppose there is a group of tasks meeting the restrictions in CASE 1, CASE 1 - Condition 2. For a contradiction, assume that optimal schedule is found by assigning the task $t_{\beta,\alpha}$ at time $r_{\beta,\alpha}^*$ into the current position. Given the restrictions, the partial sequence σ^1 containing the task $t_{\beta,\alpha}$ in its first position has the makespan within the interval $r_{\zeta,\gamma}^* + (2 * chc^\zeta + 5)p < C_{\max}(\sigma^1) < r_{\beta,\alpha}^* + (2 * chc^\zeta + 5)p + (chc^\zeta + 1)l$. Initiating the partial schedule by the task $t_{\zeta,\gamma}$ instead of the task $t_{\beta,\alpha}$ and assigning the remaining tasks in the best possible order, a new sequence σ^2 with $r_{\zeta,\gamma}^* + (2 * chc^\zeta + 5)p < C_{\max}(\sigma^2) < r_{\beta,\alpha}^* + (2 * chc^\zeta + 5)p + (chc^\zeta + 1)l$ is obtained. Both solutions are the same, thus Proposition 2 remains valid by also allowing an alternative scheduling option with the opposite order of tasks. However, the lower and upper bounds which are set according to CASE 1 and CASE 1 - Condition 2 are critical time points as the optimal order of tasks is different in the event that some of the constraints in CASE 1 - Condition 2 are violated. Review Corollary 1, 2, 3. Proof is valid for any value of chc^ζ .

Example 2. Figure 4.5 shows examples to alternative scheduling options under the restrictions of CASE 1, CASE 1 - Condition 2. Let c_j^β , $j = 0, \dots, sz^\beta - 1$, and

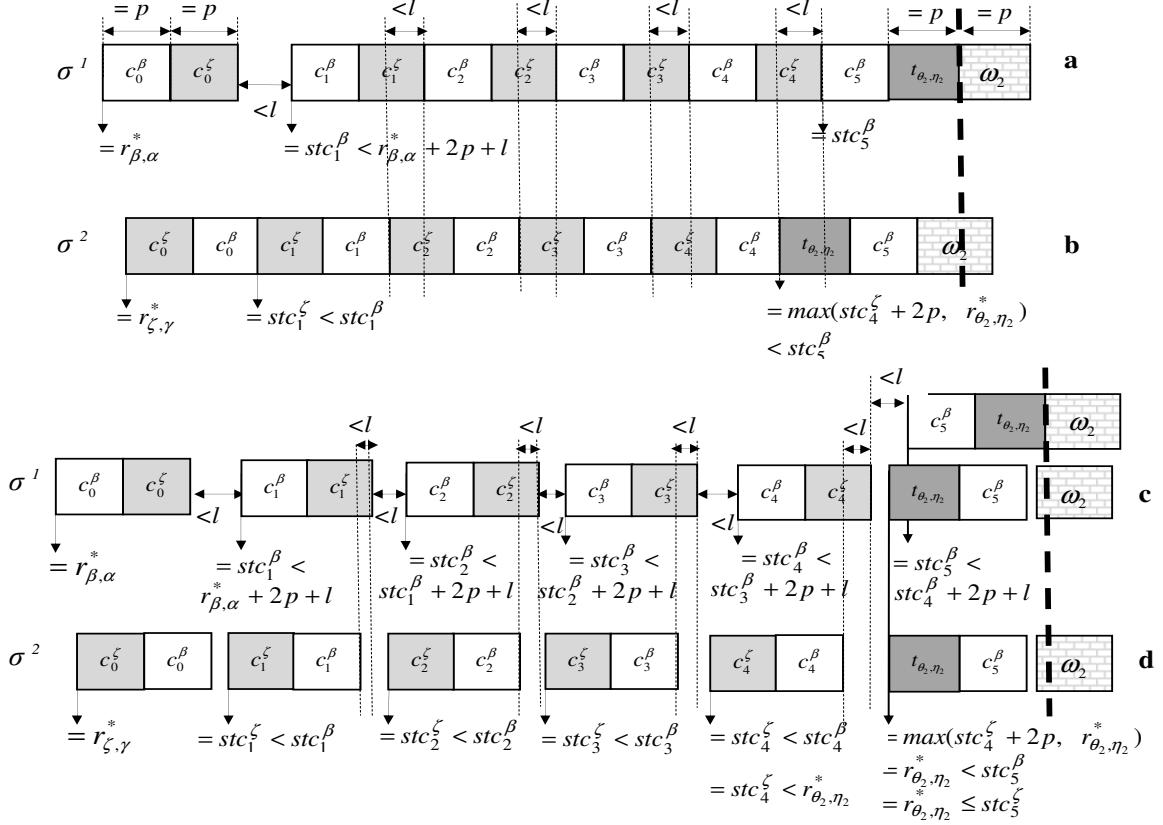


Figure 4.4: Alternative sequences with respect to CASE 1, CASE 1 - Condition 1 for $1|outtree(l < p); p_i = p; r_i|C_{max}$

c_j^ζ , $j = 0, \dots, sz^\zeta - 1$, denote the tasks in the hidden chains β and ζ and suppose that $chc^\zeta = 4$ as in Example 1. As $chc_2^{\zeta\beta} = chc^\zeta + 1$, $chc_2^{\zeta\beta} = 5$. Due to this equality, let the following structure be observed: $stc_1^\zeta < stc_1^\beta < stc_0^\beta + 2p + l$, $stc_2^\zeta < stc_2^\beta < stc_1^\beta + 2p + l$, $stc_3^\beta < stc_3^\zeta < stc_2^\beta + 2p + l$, $stc_4^\beta < stc_4^\zeta < stc_3^\beta + 2p + l$ and $\max(stc_4^\zeta + 2p, r_{\theta_2, \eta_2}^*) < stc_5^\beta < stc_4^\beta + 2p + l$. This structure is allowed to be in a different forms as long as those forms meet the required restrictions but the following are compulsory: $stc_4^\beta < stc_4^\zeta$, $\max(stc_4^\zeta + 2p, r_{\theta_2, \eta_2}^*) \leq r_{\theta_2, \eta_2}^* < stc_5^\beta$, $est_{\omega_2} < stc_5^\beta + 2p$. Example 1 contains the description of est_{ω_2} . Construct the partial sequences σ^1 and σ^2 , starting by $t_{\beta, \alpha}$ and $t_{\zeta, \gamma}$, respectively. Accordingly, $r_{\zeta, \gamma}^* + 13p < C_{max}(\sigma^1) < r_{\beta, \alpha}^* + 13p + 5l$, $r_{\zeta, \gamma}^* + 13p < C_{max}(\sigma^2) < r_{\beta, \alpha}^* + 13p + 5l$. Both scheduling alternatives generate the same solution. Figure 4.6 shows an example of the case where the constraint (vii) in CASE 1 - Condition 2 is violated by the

inequality $est_{\omega_2} > stc_5^\beta + 2p$. In this case the partial sequence σ^1 becomes the better alternative and Proposition 2 is not valid anymore. Note that both in Figure 4.5 and Figure 4.6, a shows the partial sequence σ^1 that starts by $t_{\beta,\alpha}$ and sets the lower bound, b shows the partial sequence σ^2 that starts by $t_{\zeta,\gamma}$ and sets the lower bound, c shows the partial sequence σ^1 that starts by $t_{\beta,\alpha}$ and sets the upper bound, d shows the partial sequence σ^2 that starts by $t_{\zeta,\gamma}$ and sets the upper bound

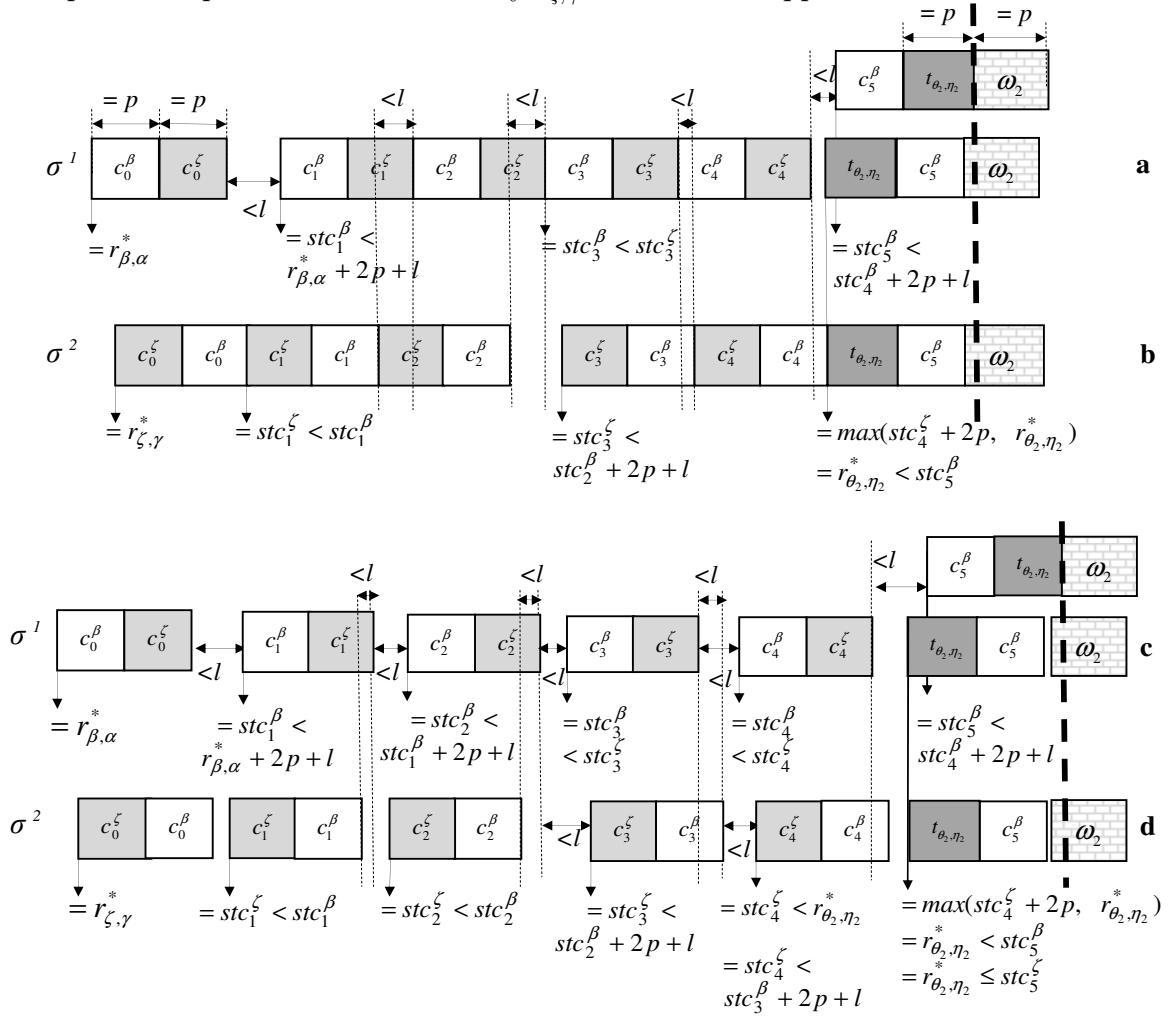


Figure 4.5: Alternative sequences with respect to CASE 1, CASE 1 - Condition 2 for $1|outtree(l < p); p_i = p; r_i|C_{max}$

Proposition 3. If the constraints in CASE 1 and CASE 1 - Condition 3 are met, optimal solution is found by assigning the task $t_{\zeta,\gamma}$ into the current position of the

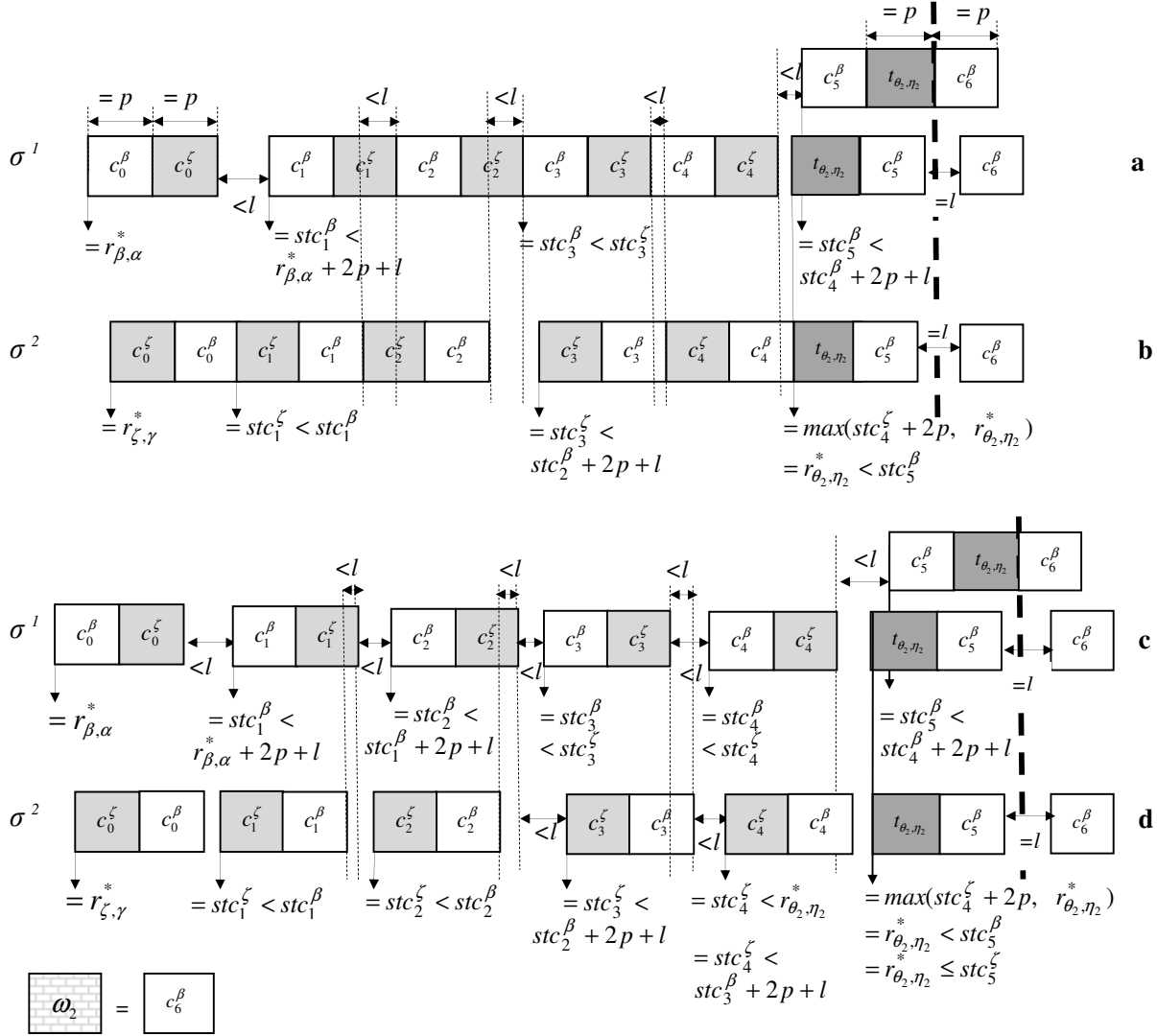


Figure 4.6: Alternative sequences when (vii) in CASE 1 - Condition 2 for $1|outtree(l < p); p_i = p; r_i|C_{max}$ is violated

schedule at time $r_{\zeta,\gamma}^*$.

Proof. Suppose that a group of tasks satisfies the constraints in CASE 1, CASE 1 - Condition 3. Let stc_1^β be bounded by $stc_0^\beta + 2p + l$ as in the other case the proof is trivial. Assume that there exists a partial sequence σ^1 that starts with with task $t_{\beta,\alpha}$ at time $r_{\beta,\alpha}^*$ after the completion of previously assigned task. Given the restrictions, $r_{\zeta,\gamma}^* + 5p \leq C_{max}(\sigma^1) < r_{\beta,\alpha}^* + 5p + l$. Swap the positions of $t_{\beta,\alpha}$ and $t_{\zeta,\gamma}$ and assign the remaining tasks accordingly to create a new sequence σ^2 . Since $r_{\zeta,\gamma}^* + 5p \leq C_{max}(\sigma^2) < r_{\beta,\alpha}^* + 5p + l$, Proposition 3 remains valid. Although both

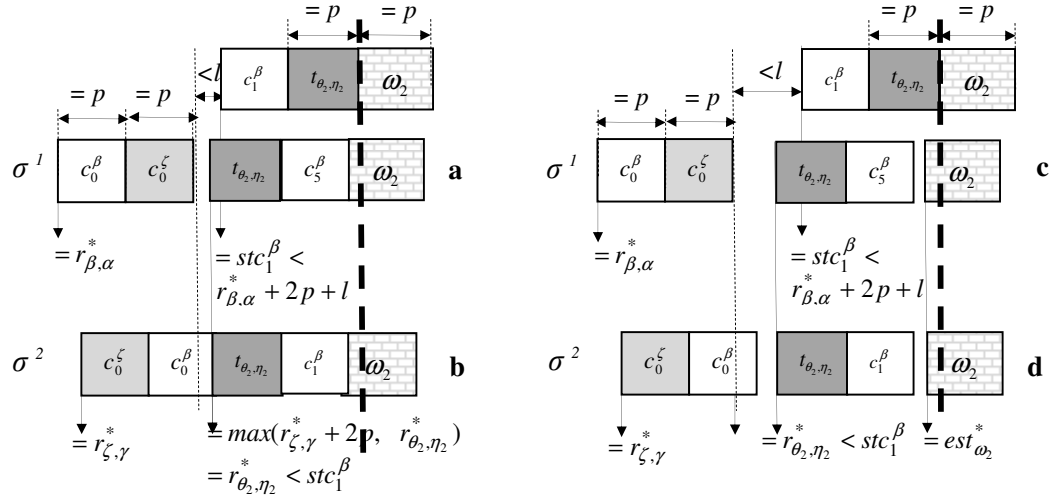


Figure 4.7: Alternative sequences with respect to CASE 1, CASE 1 - Condition 3 for $1|outtree(l < p); p_i = p; r_i|C_{max}$

scheduling options generate the same results, any minor change in the conditions of Proposition 3 may change the optimal selection strategy. Corollary 1, 2, 3 give the appropriate strategies for any violation in the constraints.

Example 3. Let $c_j^\beta, j = 0, \dots, sz^\beta - 1$, and $c_j^\zeta, j = 0, \dots, sz^\zeta - 1$, denote the tasks in the hidden chains β and ζ . Note that from $chc^\zeta = 0$, $r_{\theta_2, \eta_2}^* = r_{\theta, \eta}^* \leq stc_1^\zeta$. Regarding the constraints (ii) and (iii) in CASE 1 - Condition 3, $\max(stc_0^\zeta + 2p, r_{\theta_2, \eta_2}^*) \leq r_{\theta_2, \eta_2}^* < stc_1^\beta$ and $est_{\omega_2} < stc_1^\beta + 2p$ where est_{ω_2} is equal to the minimum of the following values: $hc_1^\zeta, r_{\Upsilon, \kappa}^*, \min_{0 \leq i < n_{\phi_2}} r_{\phi_2, i}, \max[\max(stc_0^\zeta + 2p, r_{\theta_2, \eta_2}^*) + 2p + l; hc_2^\beta], ft_0^\beta$. Also suppose that $stc_1^\beta < stc_0^\beta + 2p + l$. Given the constraints, alternative partial sequences which determine the bounds are given in Figure 4.7. Fig.4.7.a shows the partial sequence, σ^1 , that starts by the task $t_{\beta, \alpha}$ and sets the lower bound. Fig.4.7.b shows the partial sequence, σ^2 , that starts by the task $t_{\zeta, \gamma}$ and sets the lower bound. Fig.4.7.c shows the partial sequence, σ^1 , that starts by the task $t_{\beta, \alpha}$ and sets the upper bound. Fig.4.7.d shows the partial sequence, σ^2 , that starts by $t_{\zeta, \gamma}$ and sets the upper bound.

LEMMA 1. If CASE 1 - Condition 1, CASE 1 - Condition 2 or CASE 1 - Condition 3 is satisfied in addition to CASE 1, optimal solution is found by assigning the task $t_{\zeta,\gamma}$ to the current position of the schedule at time $r_{\zeta,\gamma}^*$.

Proof. Proof follows from Proposition 1, Proposition 2 and Proposition 3.

Corollary 1. When the constraint (iv) in CASE 1 - Condition 1 or (vii) in CASE 1 - Condition 2 or (iii) in CASE 1 - Condition 3 is violated by the inequality $est_{\omega_2} > stc_{chc^\zeta+1}^\beta + 2p$, optimal schedule continues with $t_{\beta,\alpha}$ which starts at time $r_{\beta,\alpha}^*$. However, if the violation stems from the equality $est_{\omega_2} = stc_{chc^\zeta+1}^\beta + 2p$, comparing the $ns_{k,i}$ for $t_{\zeta,\gamma}$ and $t_{\beta,\alpha}$ and selecting the one with the larger $ns_{k,i}$ for the current position generates the optimal solution. Figure 4.6 illustrates the violation in (vii) of CASE 1 - Condition 2.

Corollary 2. If the constraint (iv) in CASE 1 - Condition 2 is broken by $chc_2^{\zeta\beta} \neq chc^\zeta + 1$, optimal solution is obtained by assigning $t_{\zeta,\gamma}$ into the current position of the schedule at time $r_{\zeta,\gamma}^*$ when the restrictions in (v), (vi) and (vii) are still valid. When both the constraint (iv) and the constraint (vii) are violated, review Corollary 1. Also check Corollary 3.a for any violation in (v) or (vi) in addition to the violation in (iv). Corollary 3.a contains the results for the case $chc_2^{\zeta\beta} = chc^\zeta + 1$ which are also valid for the case $chc_2^{\zeta\beta} \neq chc^\zeta + 1$.

Corollary 3.

a. If the restriction (vi) in CASE 1 - Condition 2 is not satisfied while (i), (ii), (iii), (iv) and (v) are still valid, optimal solution is obtained by assigning $t_{\beta,\alpha}$ at time $r_{\beta,\alpha}^*$. If the constraints in (v) and (vi) in CASE 1 - Condition 2 are replaced by $stc_{chc^\zeta}^\beta \geq stc_{chc^\zeta}^\zeta$ and $max(stc_{chc^\zeta}^\zeta + 2p, r_{\theta_2, \eta_2}^*) < stc_{chc^\zeta+1}^\beta$, respectively, assigning the task $t_{\zeta,\gamma}$ at time $r_{\zeta,\gamma}^*$ is optimal. In this case, while $stc_{chc^\zeta}^\beta \geq stc_{chc^\zeta}^\zeta$, if $max(stc_{chc^\zeta}^\zeta + 2p, r_{\theta_2, \eta_2}^*) > stc_{chc^\zeta+1}^\beta$ selecting the task $t_{\beta,\alpha}$ is optimal and if $max(stc_{chc^\zeta}^\zeta + 2p, r_{\theta_2, \eta_2}^*) = stc_{chc^\zeta+1}^\beta$, sizes of $ns_{\beta,\alpha}$ and $ns_{\zeta,\gamma}$ are compared.

b. If there is a violation of the constraint (ii) of CASE 1 - Condition 3, optimal solution is obtained by assigning $t_{\beta,\alpha}$ at time $r_{\beta,\alpha}^*$ into the current position.

c. If $chc^\zeta \neq 0$ and $max_{0 \leq j \leq chc^\zeta-1} |(stc_{j+1}^\beta + 2p) - (stc_{j+1}^\zeta + 2p)| \geq l$ which violates the

constraint (iii) in CASE 1 - Condition 1 and in CASE 1 - Condition 2, if $control^\beta \geq l$ assigning $t_{\zeta,\gamma}$ into the current position of the schedule at time $r_{\zeta,\gamma}^*$ is optimal and if $control^\beta \leq -l$ assigning $t_{\beta,\alpha}$ into the schedule at time $r_{\beta,\alpha}^*$ is optimal.

Proposition 4. If the constraints in CASE 1 and CASE 1 - Condition 4 are completely satisfied, optimal solution is obtained by assigning the task $t_{\beta,\alpha}$ into the current position of the schedule at time $r_{\beta,\alpha}^*$.

Proof. Let the constraints in CASE 1 - Condition 4 be satisfied for a group of tasks and assume that optimal solution is obtained by assigning the task $t_{\zeta,\gamma}$ into the current position of the schedule at time $r_{\zeta,\gamma}^*$. Then, there exists a partial sequence σ^1 with $r_{\zeta,\gamma}^* + (2*chc^\beta + 5)p \leq C_{max}(\sigma^1) < r_{\zeta,\gamma}^* + (2*chc^\beta + 5)p + (chc^\beta + 1)l$. When the tasks $t_{\beta,\alpha}$ and $t_{\zeta,\gamma}$ in sequence σ^1 are swapped and the remaining tasks are relocated, a new sequence σ^2 with $r_{\beta,\alpha}^* + (2*chc^\beta + 5)p \leq C_{max}(\sigma^2) < r_{\zeta,\gamma}^* + (2*chc^\beta + 5)p + (chc^\beta + 1)l$ is found. This conflicts with the assumption that selecting the task $t_{\zeta,\gamma}$ for current position gives the optimal solution. Proof is valid for any value of chc^β .

Example 4. Let c_j^β , $j = 0, \dots, sz^\beta - 1$, and c_j^ζ , $j = 0, \dots, sz^\zeta - 1$, denote the tasks in the hidden chains β and ζ and assume that $chc^\beta = 4$. Accordingly, $stc_4^\beta < r_{\theta_1, \eta_1}^* = \min[r_{\theta, \eta}^*, \min_{0 \leq z < 4} ft_z^\beta, \min_{0 \leq z < 4} ft_z^\zeta] \leq stc_5^\beta$ and $chc_1^{\beta\zeta} = chc^\beta + 1 = 5$. Thus, $stc_1^\beta < stc_1^\zeta < stc_0^\beta + 2p + l$, $stc_2^\beta < stc_2^\zeta < stc_1^\beta + 2p + l$, $stc_3^\beta < stc_3^\zeta < stc_2^\beta + 2p + l$, $stc_4^\beta < stc_4^\zeta < stc_3^\beta + 2p + l$, and $\max(stc_4^\beta + 2p, r_{\theta_1, \eta_1}^*) < stc_5^\zeta < stc_4^\zeta + 2p + l$. Moreover, $est_{\omega_1} < stc_5^\zeta + 2p$ where est_{ω_1} is equal to the minimum of the following values: hc_5^β , $r_{\Upsilon, \kappa}^*$, $r_{\theta, \eta}^*$ if $\theta \neq \theta_1$ and $\eta \neq \eta_1$, $\min_{0 \leq z < 4} ft_z^\beta$ for $lk_z^\beta \neq \theta_1$ and $li_z^\beta \neq \eta_1$, $\min_{0 \leq z < 4} ft_z^\zeta$ for $lk_z^\zeta \neq \theta_1$ and $li_z^\zeta \neq \eta_1$, $\min_{0 \leq z < 4} fts_z^\beta$, $\min_{0 \leq z < 4} fts_z^\zeta$, $\min_{0 \leq i < n_{\phi_1}} r_{\phi_1, i}$, $\max[\max(stc_4^\beta + 2p, r_{\theta_1, \eta_1}^*) + 2p + l; hc_6^\zeta]$, ft_4^ζ . For the alternative partial sequences, the makespan remains in the following intervals: $r_{\zeta,\gamma}^* + 13p \leq C_{max}(\sigma^1) < r_{\zeta,\gamma}^* + 13p + 5l$ and $r_{\beta,\alpha}^* + 13p \leq C_{max}(\sigma^2) < r_{\zeta,\gamma}^* + 13p + 5l$. Therefore, optimal solution is found by selecting the task $t_{\beta,\alpha}$ for the current position of the schedule. Figure 4.8.a shows the partial sequence which sets the lower bound for σ^1 that starts by $t_{\zeta,\gamma}$, Figure 4.8.b shows the partial sequence which sets the lower bound for σ^2 that starts by $t_{\beta,\alpha}$, Figure 4.8.c shows the partial sequence which sets the upper bound for σ^1 that starts

by $t_{\zeta,\gamma}$, Figure 4.8.d shows the partial sequence which sets the upper bound for σ^2 that starts by $t_{\beta,\alpha}$.

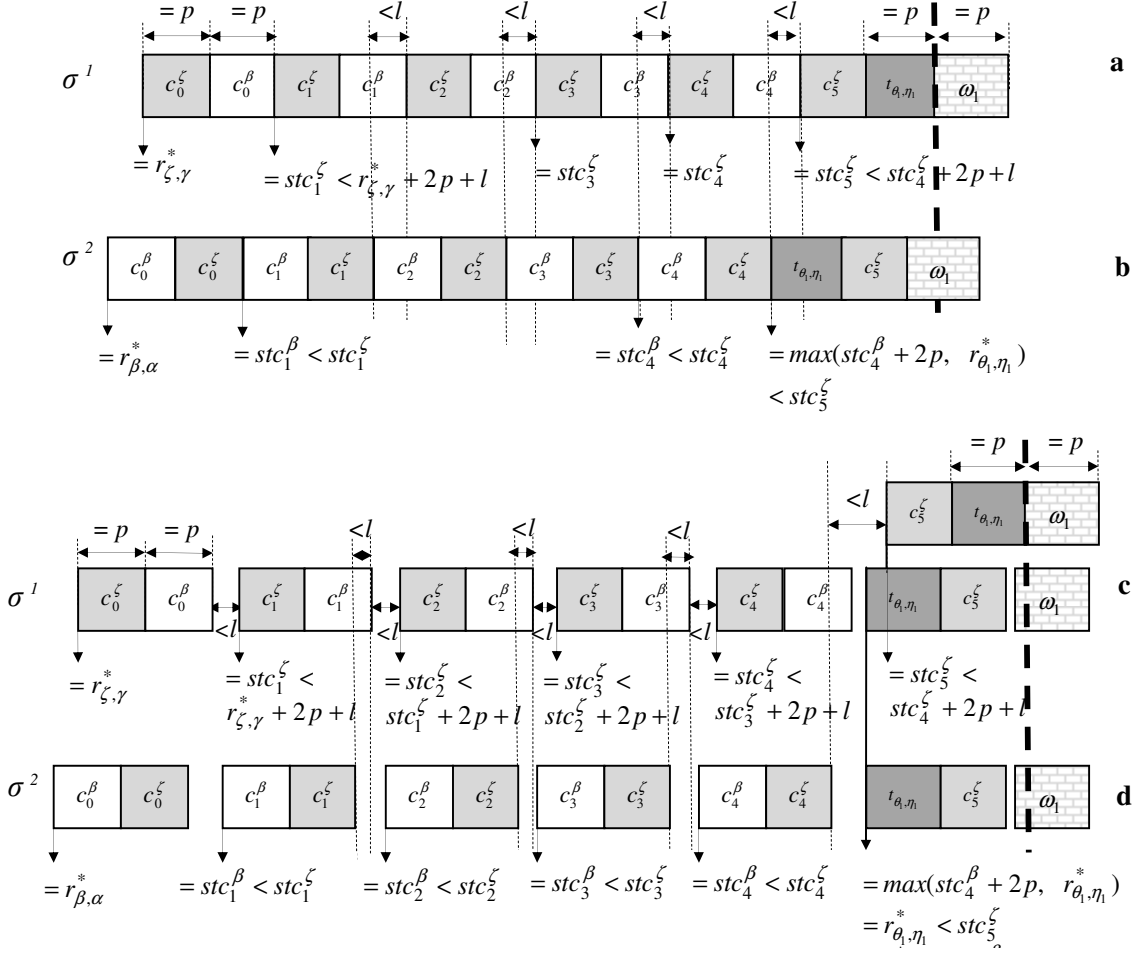


Figure 4.8: Alternative sequences with respect to CASE 1, CASE 1 - Condition 4 for $1|outtree(l < p); p_i = p; r_i|C_{max}$

Proposition 5. If CASE 1 and CASE 1 - Condition 5 hold, optimal solution is found by assigning the task $t_{\beta,\alpha}$ into the current position of the schedule at time $r_{\beta,\alpha}^*$.

Proof. Assume to reach a contradiction that assigning the task $t_{\zeta,\gamma}$ into the current position of the schedule at time $r_{\zeta,\gamma}^*$ generates the optimal solution when a group of tasks meets the restrictions in CASE 1, CASE 1 - Condition 5. In this case, the best partial sequence σ^1 that is created in accordance with precedence, timelag and

release time restrictions has the makespan within the interval $r_{\beta,\alpha}^* + (2 * chc^\beta + 5)p < C_{max}(\sigma^1) < r_{\zeta,\gamma}^* + (2 * chc^\beta + 5)p + (chc^\beta + 1)l$. By swapping the tasks $t_{\zeta,\gamma}$ and $t_{\beta,\alpha}$ and rearranging the remaining tasks in the best order with respect to given constraints, a new partial sequence σ^2 with $r_{\beta,\alpha}^* + (2 * chc^\beta + 5)p < C_{max}(\sigma^2) < r_{\zeta,\gamma}^* + (2 * chc^\beta + 5)p + (chc^\beta + 1)l$ is constructed. Neither a better nor a worse solution is found by the alternative scheduling option and because of this fact Proposition 5 preserves its validity. The bounds obtained by Proposition 5 are critical since by slightly shifting any one of the given release time values optimal strategy may change. Corollary 4, 5, 6 provides details.

Example 5. See Figure 4.9 for an illustration of the alternative sequences for CASE 1 and CASE 1 - Condition 5. The tasks in the hidden chains β and ζ are again shown by c_j^β , $j = 0, \dots, sz^\beta - 1$, and c_j^ζ , $j = 0, \dots, sz^\zeta - 1$. Suppose that $chc^\beta = 4$ as in Example 4. From CASE 1 - Condition 5, $chc_2^{\beta\zeta} = chc^\beta + 1 = 5$. Due to this equality, let the following structure be observed: $stc_1^\beta < stc_1^\zeta < stc_0^\zeta + 2p + l$, $stc_2^\beta < stc_2^\zeta < stc_1^\zeta + 2p + l$, $stc_3^\zeta < stc_3^\beta < stc_2^\zeta + 2p + l$, $stc_4^\zeta < stc_4^\beta < stc_3^\zeta + 2p + l$ and $max(stc_4^\beta + 2p, r_{\theta_1,\eta_1}^*) < stc_5^\zeta < stc_4^\zeta + 2p + l$. Note that the following are compulsory: $stc_4^\zeta < stc_4^\beta$, $max(stc_4^\beta + 2p, r_{\theta_1,\eta_1}^*) \leq r_{\theta_1,\eta_1}^* < stc_5^\zeta$, $est_{\omega_1} < stc_5^\zeta + 2p$. est_{ω_1} is defined as in Example 4. Construct the partial sequences σ^1 and σ^2 , starting by $t_{\zeta,\gamma}$ and $t_{\beta,\alpha}$, respectively. Accordingly, $r_{\beta,\alpha}^* + 13p < C_{max}(\sigma^1) < r_{\zeta,\gamma}^* + 13p + 5l$ and $r_{\beta,\alpha}^* + 13p < C_{max}(\sigma^2) < r_{\zeta,\gamma}^* + 13p + 5l$. Both are the same. Figure 4.9.a shows the tightest partial sequence σ^1 that starts by $t_{\zeta,\gamma}$, Figure 4.9.b shows the tightest partial sequence σ^2 that starts by $t_{\beta,\alpha}$, Figure 4.9.c shows the longest partial sequence σ^1 that starts by $t_{\zeta,\gamma}$, Figure 4.9.d shows the longest partial sequence σ^2 that starts by $t_{\beta,\alpha}$.

Proposition 6. If CASE 1 and CASE 1 - Condition 6 are met, optimal solution is found by assigning the task $t_{\beta,\alpha}$ into the current position of the schedule at time $r_{\beta,\alpha}^*$.

Proof. Suppose that $stc_1^\zeta < stc_0^\zeta + 2p + l$. Because in the other case the proof is trivial. Assume that assigning the task $t_{\zeta,\gamma}$ into the current position of the schedule is the optimal choice under the restrictions of CASE 1 and CASE 1 - Condition 6.

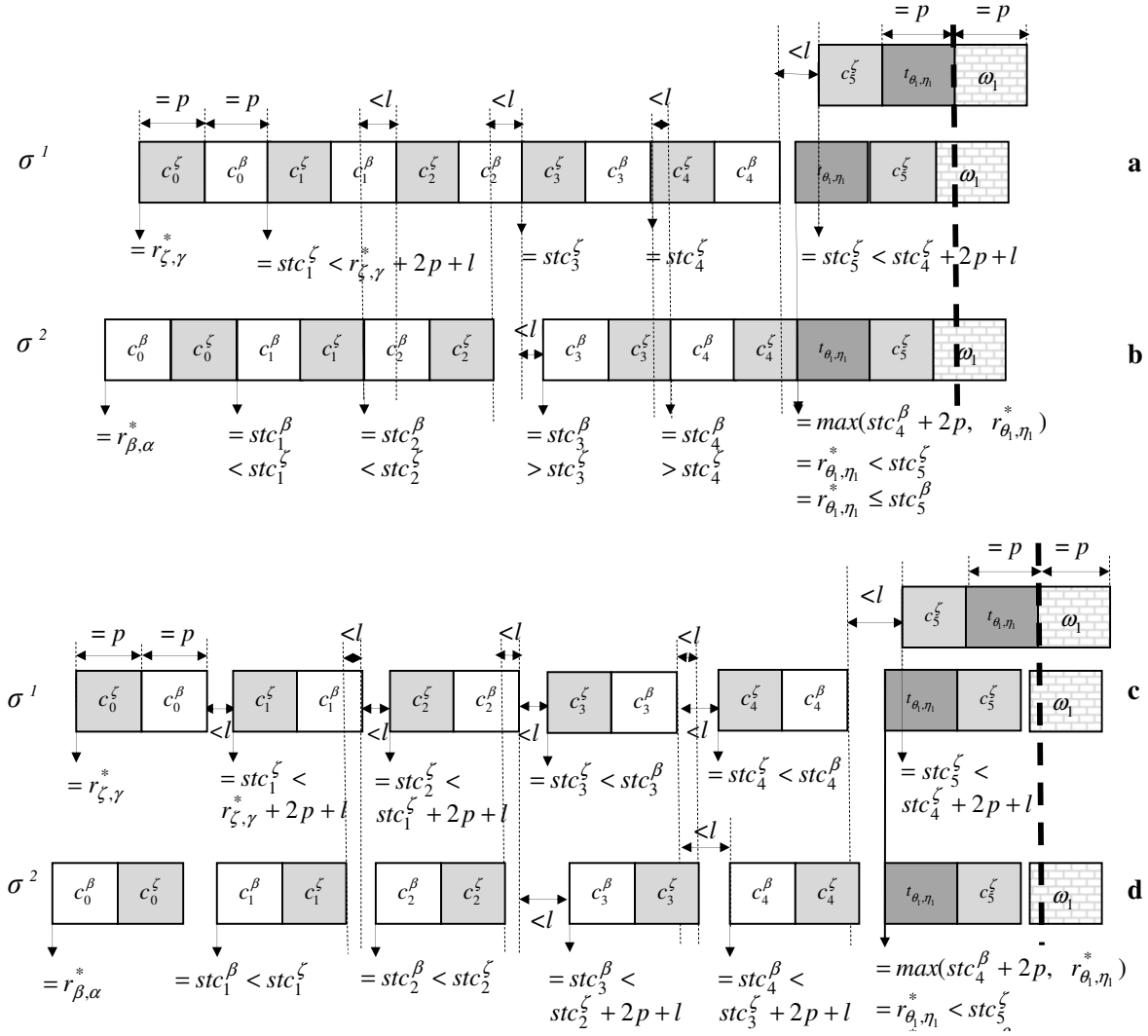


Figure 4.9: Alternative sequences with respect to CASE 1, CASE 1 - Condition 5 for $1|outtree(l < p); p_i = p; r_i|C_{max}$

Then, the partial sequence σ^1 starting at time $r_{\zeta,\gamma}^*$ by the task $t_{\zeta,\gamma}$ has the makespan bounded by the interval $r_{\zeta,\gamma}^* + 5p \leq C_{max}(\sigma^1) < r_{\zeta,\gamma}^* + 5p + l$. Swap the positions of $t_{\beta,\alpha}$ and $t_{\zeta,\gamma}$ to create the partial sequence σ^2 starting at time $r_{\beta,\alpha}^*$. This results in a shorter makespan within the interval $r_{\beta,\alpha}^* + 5p \leq C_{max}(\sigma^2) < r_{\zeta,\gamma}^* + 5p + l$. Since this is a contradiction, the proof is completed.

Example 6. Let $c_j^\beta, j = 0, \dots, sz^\beta - 1$, and $c_j^\zeta, j = 0, \dots, sz^\zeta - 1$ denote the tasks in the hidden chains β and ζ . From $chc^\beta = 0, r_{\theta_1, \eta_1}^* = r_{\theta, \eta}^* \leq stc_1^\beta$. With respect to (ii) and (iii) in Condition 6, $\max(stc_0^\beta + 2p, r_{\theta_1, \eta_1}^*) < stc_1^\zeta$ and $est_{\omega_1} < stc_1^\zeta + 2p$ where est_{ω_1} is equal to the minimum of the following values: $hc_1^\beta, r_{\Upsilon, \kappa}^*, \min_{0 \leq i < n_{\phi_1}} r_{\phi_1, i}, \max[\max(stc_0^\beta + 2p, r_{\theta_1, \eta_1}^*) + 2p + l; hc_2^\zeta], ft_0^\zeta$. Also assume that $stc_1^\zeta < stc_0^\zeta + 2p + l$. Construct the related partial sequences. Figure 4.10.a shows the tightest partial sequence σ^1 that starts by $t_{\zeta,\gamma}$, Figure 4.10.b shows the tightest partial sequence σ^2 that starts by $t_{\beta,\alpha}$, Figure 4.10.c shows the longest partial sequence σ^1 that starts by $t_{\zeta,\gamma}$, Figure 4.10.d shows the longest partial sequence σ^2 that starts by $t_{\beta,\alpha}$.

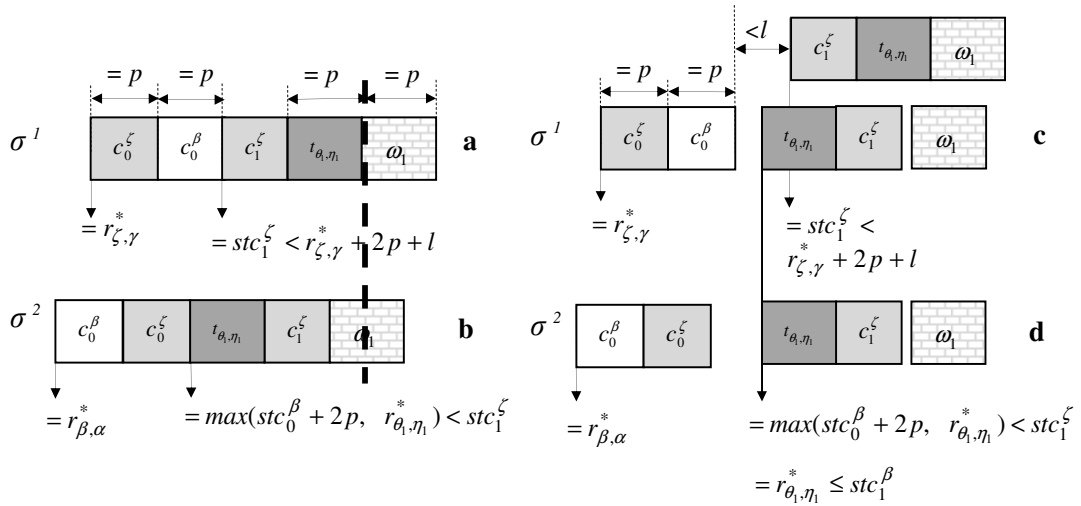


Figure 4.10: Alternative sequences with respect to CASE 1, CASE 1 - Condition 6 for $1 | outtree(l < p); p_i = p; r_i | C_{max}$

LEMMA 2. When CASE 1 holds, if CASE 1 - Condition 4, CASE 1 - Condition 5 or CASE 1 - Condition 6 is satisfied, optimal solution is found by assigning the task

$t_{\beta,\alpha}$ into the current position of the schedule at time $r_{\beta,\alpha}^*$.

Proof. Proof follows from Proposition 4, Proposition 5 and Proposition 6.

Corollary 4. If the constraint (iv) in CASE 1 - Condition 4 or (vii) in CASE 1 - Condition 5 or (iii) in CASE 1 - Condition 6 is violated by the inequality $est_{\omega_1} > stc_{chc^{\beta}+1}^{\zeta} + 2p$, optimal schedule continues with $t_{\zeta,\gamma}$ which starts at time $r_{\zeta,\gamma}^*$. On the other hand, if the equality $est_{\omega_1} = stc_{chc^{\beta}+1}^{\zeta} + 2p$ breaks the rule, comparing the $ns_{\zeta,\gamma}$ and $ns_{\beta,\alpha}$ and selecting the task with the larger $ns_{k,i}$ for the current position is the optimal choice.

Corollary 5. If the constraint (iv) in CASE 1 - Condition 5 is destroyed by $chc_2^{\beta\zeta} \neq chc^{\beta} + 1$, optimal solution is obtained by assigning $t_{\beta,\alpha}$ into the current position of the schedule at time $r_{\beta,\alpha}^*$ when the restrictions in (v), (vi) and (vii) are still valid. When both the constraint (iv) and the constraint (vii) are violated, the result in Corollary 5 is valid. Also check Corollary 6.a for any violation in (v) or (vi) in addition to the violation in (iv). Corollary 6.a contains the results for the case $chc_2^{\beta\zeta} = chc^{\beta} + 1$ which are also valid for the case $chc_2^{\beta\zeta} \neq chc^{\beta} + 1$.

Corollary 6.

a. If the restriction (vi) in CASE 1 - Condition 5 is not satisfied while (i), (ii), (iii), (iv) and (v) are still valid, optimal solution is obtained by assigning $t_{\zeta,\gamma}$ at time $r_{\zeta,\gamma}^*$. If the constraints in (v) and (vi) in CASE 1 - Condition 5 are replaced by $stc_{chc^{\beta}}^{\zeta} \geq stc_{chc^{\beta}}^{\beta}$ and $max(stc_{chc^{\beta}}^{\beta} + 2p, r_{\theta_1,\eta_1}^*) < stc_{chc^{\beta}+1}^{\zeta}$, respectively, initiating the task $t_{\beta,\alpha}$ at time $r_{\beta,\alpha}^*$ is optimal. In this case, while $stc_{chc^{\beta}}^{\zeta} \geq stc_{chc^{\beta}}^{\beta}$, if $max(stc_{chc^{\beta}}^{\beta} + 2p, r_{\theta_1,\eta_1}^*) > stc_{chc^{\beta}+1}^{\zeta}$ selecting the task $t_{\zeta,\gamma}$ is optimal and if $max(stc_{chc^{\beta}}^{\beta} + 2p, r_{\theta_1,\eta_1}^*) = stc_{chc^{\beta}+1}^{\zeta}$, sizes of $ns_{\beta,\alpha}$ and $ns_{\zeta,\gamma}$ are compared.

b. If there is a violation of the constraint (ii) of CASE 1 - Condition 6, optimal solution is obtained by assigning $t_{\zeta,\gamma}$ at time $r_{\zeta,\gamma}^*$ into the current position.

c. If $chc^{\beta} \neq 0$ and $max_{0 \leq j \leq chc^{\beta}-1} |(stc_{j+1}^{\zeta} + 2p) - (stc_{j+1}^{\beta} + 2p)| \geq l$ which violates the constraint (iii) in CASE 1 - Condition 4 and in CASE 1 - Condition 5, if $control^{\zeta} \geq l$ assigning $t_{\beta,\alpha}$ into the current position of the schedule at time $r_{\beta,\alpha}^*$ is optimal and if $control^{\zeta} \leq -l$ assigning $t_{\zeta,\gamma}$ into the schedule at time $r_{\zeta,\gamma}^*$ is optimal.

LEMMA 3. If CASE 2 is satisfied, optimal schedule has the task $t_{\beta,\alpha}$ that starts at time $r_{\beta,\alpha}^*$ in the current position.

Proof. Proof is trivial. Regarding CASE 2, construct the alternative sequences σ^1 and σ^2 starting at time $r_{\zeta,\gamma}^*$ and $r_{\beta,\alpha}^*$ for any random values of release times of the remaining tasks. In σ^1 , $t_{\beta,\alpha}$ follows $t_{\zeta,\gamma}$ and in σ^2 , $t_{\zeta,\gamma}$ follows $t_{\beta,\alpha}$ as usual. Therefore, because of the timelag constraint any task from the successor group of $t_{\beta,\alpha}$ cannot be assigned before $r_{\zeta,\gamma}^* + 2p + l$ after $t_{\beta,\alpha}$ in σ^1 and any task from the successor group of $t_{\zeta,\gamma}$ cannot be assigned before $r_{\beta,\alpha}^* + 2p + l$ after $t_{\zeta,\gamma}$ in σ^2 . Such excessive time delays may also occur in the further positions of the alternative sequences. However, thanks to the CASE 2, any excessive timelag can be tolerated in σ^2 . To give an example, assume that a timelag is able to be eliminated by assigning a task from the successor group of $t_{\zeta,\gamma}$ into the schedule after $t_{\beta,\alpha}$ in sequence σ^1 at time $r_{\zeta,\gamma}^* + 2p$. But the condition $r_{\zeta,\gamma}^* - r_{\beta,\alpha}^* \geq l$ in CASE 2 shows that $r_{\beta,\alpha}^* + 2p + l \leq r_{\zeta,\gamma}^* + 2p$ and any partial sequence built by selecting the task from the successor group of $t_{\zeta,\gamma}$ to start can also be built in σ^2 by assigning the selected task after $t_{\zeta,\gamma}$, with no additional time delay and even with a possible contribution to length of the schedule by initiating the selected task at an earlier time than $r_{\zeta,\gamma}^* + 2p$ depending on its release time and other constraints. This completes the proof.

4.3 Complexity of the proposed algorithm for $1|outtree(l < p); p_i = p; r_i|C_{max}$

General structure of the proposed algorithm is shown in Figure 11. In the proposed algorithm, at each iteration, candidate tasks for the current position are determined with respect to the precedence relations between the tasks; earliest start times for the candidate tasks are updated according to the timelag and release time constraints and the maximum completion time of the scheduled tasks; hidden chains containing the candidate tasks are found; the tasks which may be scheduled between the tasks from the hidden chains without violating the precedence, release time and timelag constraints are searched; alternative partial sequences starting by the candidate tasks and ending by a predetermined task at a further time are constructed and compared;

the task that is to be assigned into the current position of the schedule is chosen.

Details are given in the pseudo code:

Pseudo code

Input: $p, l, K, n_k, r_{k,i}, t_{k,i}, pr_{k,i}, ns_{k,i}$

While total number of unassigned tasks > 0

 Compute or update the following dynamic variables:

$ct, \Omega, \delta, lsg^*, r_{\Omega,\delta}, m_{lsg^*,i}, pr_{k,i}^*, max_{0 \leq k < K, 0 \leq i < n_k} pr_{k,i}^*, r_{k,i}^*, r_{\beta,\alpha}^*, r_{\zeta,\gamma}^*, r_{\theta,\eta}^*, r_{\Upsilon,\kappa}^*,$
 $hc_j^\beta, sz^\beta, ft_z^\beta, fts_z^\beta, lk_z^\beta, li_z^\beta, hc_j^\zeta, sz^\zeta, ft_z^\zeta, fts_z^\zeta, lk_z^\zeta, li_z^\zeta, stc_j^\zeta, stc_j^\beta, chc^\beta,$
 $\theta_1, \eta_1, r_{\theta_1,\eta_1}^*, chc^\zeta, \theta_2, \eta_2, r_{\theta_2,\eta_2}^*, chc_1^{\beta\zeta}, chc_2^{\beta\zeta}, \phi_1, min_{0 \leq i < n_{\phi_1}} r_{\phi_1,i}, \omega_1, chc_1^{\zeta\beta},$
 $chc_2^{\zeta\beta}, \phi_2, min_{0 \leq i < n_{\phi_2}} r_{\phi_2,i}, \phi_2, \omega_2, control^\zeta, control^\beta$

 If $r_{\zeta,\gamma}^* - r_{\beta,\alpha}^* < l$

 Check the optimality conditions:

 CASE 1 - Condition 1, CASE 1 - Condition 2, CASE 1 - Condition 3,
 CASE 1 - Condition 4, CASE 1 - Condition 5, CASE 1 - Condition 6,
 Corollary 1, 2, 3, 4, 5, 6

 Apply the corresponding strategy to choose a task for the current position.

 (Review LEMMA 1, 2, Corollary 1, 2, 3, 4, 5, 6 for related strategy)

 If $r_{\zeta,\gamma}^* - r_{\beta,\alpha}^* \geq l$

 Apply the strategy in LEMMA 3 to choose a task for the current position

 Determine the start time of the selected task.

 Repeat the procedure

Complexity of the proposed algorithm is analyzed. The results is given in the following LEMMA.

LEMMA 4. Complexity of the proposed algorithm is $O(n^3)$.

Proof. The size of the problem is the total number of tasks that will be scheduled on a single machine which is denoted by n . Note that $K = n + 1$ and $n_k < n$ where $k = 0, \dots, K - 1$. Let the number of dummy groups be denoted by n_d . Number of dummy groups is equal to the number of tasks belonging to the end leaves of the outtree for which there is no successor task and thus $n_d < n$. Also note that each

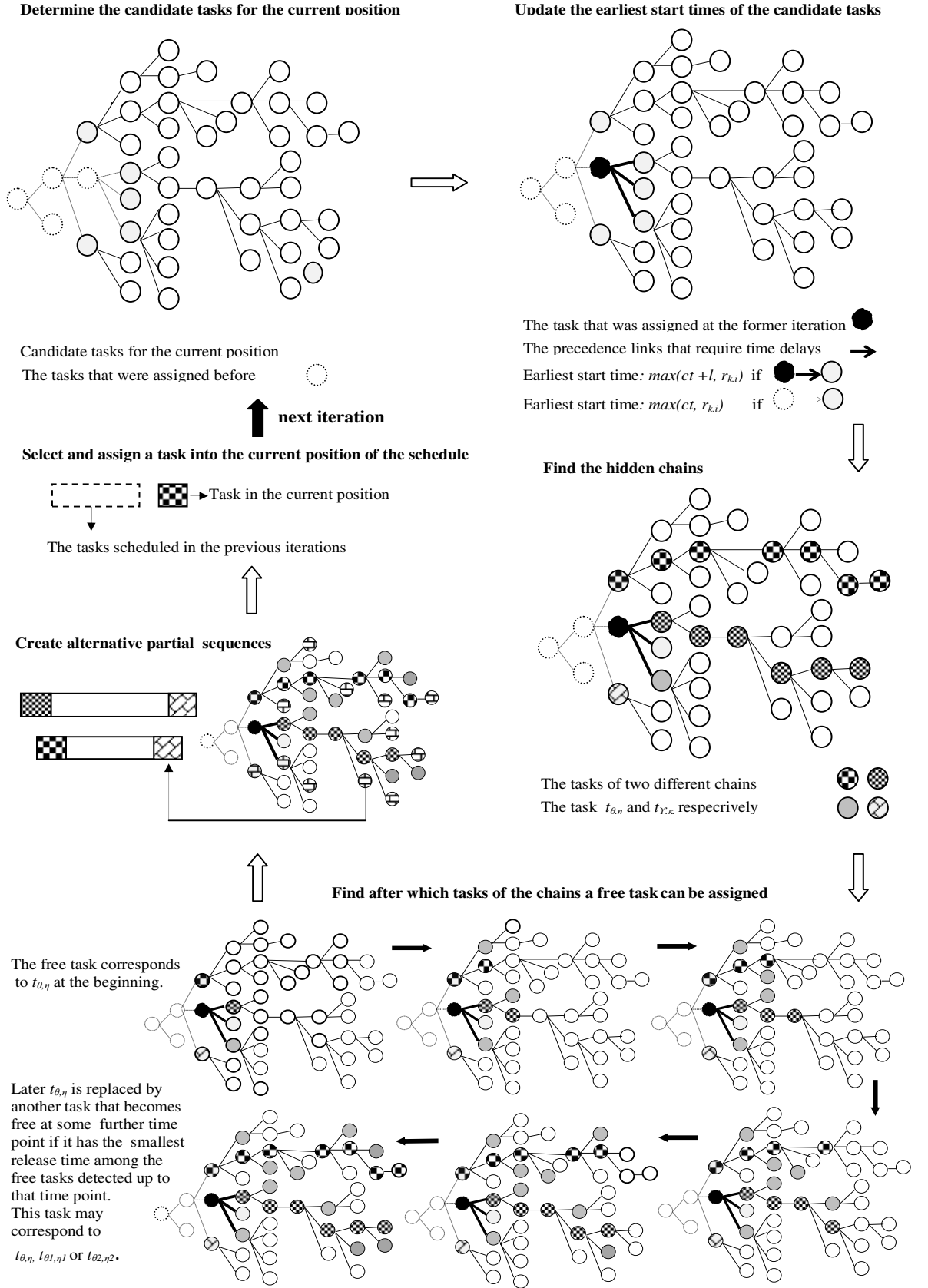


Figure 4.11: General structure of the proposed algorithm for $1|outtree(l < p); p_i = p; r_i|C_{max}$

dummy group is assumed to have one dummy task.

Updating the release time matrix $r_{k,i}^*$ for $k = 0, \dots, K - 1$ and $i = 0, \dots, n_k - 1$, obtaining the minimum values such as $\min_{0 \leq k < K, 0 \leq i < n_k} r_{k,i}^*$ are examples of procedures that take $O(\sum_{0 \leq k < K} n_k) = O(n + n_d) < O((n + 1)(n)) = O(n^2 + n) = O(n^2)$.

Obtaining the hidden chains takes at most $O(n_{prec}n_{lsg}) < O(n^2)$ where n_{prec} shows the total number of precedence levels which is smaller than n and n_{lsg} is the number of tasks in group lsg that corresponds to a different group at each precedence level.

Creating the arrays stc_j^ζ , $j = 0, \dots, sz^\zeta - 2$, and stc_j^β , $j = 0, \dots, sz^\beta - 2$ takes at most $O(sz^\zeta) < O(n_{prec}) < O(n)$ or $O(sz^\zeta) < O(n_{prec}) < O(n)$ time.

Counting procedures are $O(n_{prec}n_{prec}) < O(n^2)$. Because, for $j = 0, \dots, sz^\beta - 2$, r_{θ_1, η_1}^* is calculated which is equal to $\min[r_{\theta, \eta}^*, \min_{0 \leq z < j} ft_z^\beta, \min_{0 \leq z < j} ft_z^\zeta]$ in addition to the comparison operation that takes $O(1)$ time and for $j = 0, \dots, sz^\zeta - 2$ similar steps are repeated.

Computing the $\max_{0 \leq j \leq chc^\zeta - 1} |(stc_{j+1}^\beta + 2p) - (stc_{j+1}^\zeta + 2p)| < l$ or $\max_{0 \leq j \leq chc^\beta - 1} |(stc_{j+1}^\zeta + 2p) - (stc_{j+1}^\beta + 2p)| < l$ takes $O(\max(chc^\zeta, chc^\beta)) < O(n_{prec}) < O(n)$

Checking any condition is a comparison operation, therefore it is $O(1)$.

All procedures are repeated for n times. Therefore, on the whole, the algorithm is $O(n^3)$.

THEOREM. The scheduling problem $1|outtree(l < p); p_i = p; r_i|C_{max}$ is polynomially solvable.

Proof. Proof follows from LEMMA 1, LEMMA 2, LEMMA 3 and the related corollaries.

4.4 Integer programming model for $1|outtree(l < p); p_i = p; r_i|C_{max}$

A network model that solves the problem $1|outtree(l); p_i = p; r_i|C_{max}$ to optimality is proposed. Let $G(V, A, E)$ be the given graph where V is the set of vertices, A is the set of directed arcs (i, j) and E is the set of edges (i, j) . To convert the scheduling problem to the proposed network model, each task i is represented as a vertex of the graph and one more dummy task is defined as $n \in V$. The set of $(i, j) \in A$ includes

the directed links between the tasks in the outtree, the links emanating from the nodes at the end leaves of the outtree and going into the node n . All $(i, j) \in E$ are the undirected links between the tasks in the outtree for which there is no precedence relation. For integer programming model, the solution generated by the polynomial algorithm given in Section 4.2 is used as initial solution.

Notation

- Sets and Parameters :

O_i : Set of alternative destinations $j \in V$ where $(i, j) \in A$ or $(i, j) \in E$ for each $i \in V$ except for $i = n$.

I_j : Set of alternative sources $i \in V$ where $(i, j) \in A$ or $(i, j) \in E$ for each $j \in V$ except for $j = 0$.

r_i : Release time for task $i \in V$. For $i = n$, $r_n = 0$ and $i \neq n$, $r_i \geq 0$.

p_i : Processing time of task i where $i \in V$. If $i \neq n$, $p_i = p$. When $i = n$, $p_i = 0$.

l : Duration of timelag.

Z^0 : Objective function value of the initial solution that is obtained by applying the polynomial algorithm in Section 4.2.

LB_2^0 : Initial lower bound on the objective function value. This bound is equal to $\max[r_{0,0} + p + l; \min_{0 \leq i < n_1} r_{1,i}] + (n - 1)p$ where n is the original size of the scheduling problem that does not include the dummy node in the network model. For each instance, this lower bound is computed by an external program.

-Decision variables

x_{ij} : Binary variable. $x_{ij} = 1$ if task j is assigned immediately after task i . In the other case $x_{ij} = 0$.

x_{ij}^0 : $x_{ij}^0 = 1$ if task j is assigned immediately after task i in the initial solution.

S_i : Start time at node $i \in V$.

S_i^0 : Start time at node $i \in V$ in the initial solution.

C_i : Completion time at node $i \in V$.

C_i^0 : Completion time at node $i \in V$ in the initial solution.

-Objective function

C_n : Completion time at node n .

Model

$$\text{minimize } C_n \quad (1)$$

$$C_i = S_i + p_i \quad \text{for each } i \in V \quad (2)$$

$$S_n \geq C_i \quad \text{for each } i \in V \text{ where } i \neq n. \quad (3)$$

$$S_i \geq r_i \quad \text{for each } i \in V \quad (4)$$

$$S_j + M(1 - x_{ij}) \geq C_i \quad \text{for each } (i, j) \in E \text{ and } (i, j) \in A \quad (5)$$

$$S_j \geq C_i + l \quad \text{for each } (i, j) \in A, i \neq 0, j \neq n \quad (6)$$

$$\sum_{j \in O_i} x_{ij} = 1 \quad \text{for each } i \in V \text{ where } i \neq n \quad (7)$$

$$\sum_{i \in I_j} x_{ij} = 1 \quad \text{for each } j \in V \text{ where } j \neq 0 \quad (8)$$

$$x_{ij} + x_{ji} \leq 1 \quad \text{for each } (i, j) \in E \quad (9)$$

$$C_i \leq Z^0 \quad \text{for each } i \in V \quad (10)$$

$$S_i \leq Z^0 \quad \text{for each } i \in V \quad (11)$$

$$C_n \geq LB_2^0 \quad \text{for each } i \in V \quad (12)$$

In this model, (1) is the objective function. Aim is minimizing the makespan and in the proposed network makespan corresponds to completion time computed at node n .

The constraint set (2) computes the completion time of task i which is equal to sum of start time and processing time of task i .

In the constraint set (3), S_n , the minimum time the node n can start must be greater than or equal to C_i at any node.

The constraint set (4) requires that start time of a task must be larger than its release time.

The constraint set (5) is a disjunctive constraint. Accordingly, if task j is scheduled immediately after task i , start time of task j must be larger than the completion time of task i .

The constraint set (6) states that if the link between nodes i and j is a directed link emanating from i and entering into j , start time of task j must be always larger

than the C_i value of node i plus the timelag, l .

Due to the constraint set (7), in a feasible solution, only one of the incoming arcs is chosen for each i where $i \neq 0$.

Due to the constraint set (8), in a feasible solution, only one of the outgoing arcs is chosen for each i where $i \neq n$.

With respect to the constraint (9), for any edge (i, j) , at most one of the directed links which are from i to j and j to i is allowed to be in the optimal path from node 0 to node n .

Due to constraint set (10), objective function value of the initial solution is an upper bound for the decision variable, C_i , for each $i \in V$.

Due to constraint set (11), objective function value of the initial solution is an upper bound for the decision variable, S_i , for each $i \in V$.

Due to constraint (12), objective function value cannot be smaller than the lower bound, LB_2^0 .

Technical notes for the integer programming model

In the integer programming model, the solution generated by the polynomial algorithm given in Section 4.2 is used as the initial feasible solution. Thus, in GAMS/CPLEX option file, mipstart=1 instead of the default value mipstart=0. Details are given:

F^0 : Set of the links, (i, j) , which belong to the path generated by applying the polynomial algorithm in Section 4.2. This path represents the order of scheduled tasks.

ST_i^0 : Start time value of task i which is obtained by applying the polynomial algorithm in Section 4.2.

CT_i^0 : Completion time value of task i which is obtained by applying the polynomial algorithm in Section 4.2.

Z^0 : Objective function value of the initial solution that is obtained by applying the polynomial algorithm in Section 4.2.

$$x_{ij}^0 = 1 \quad \text{for each } (i, j) \in F^0 \quad (13)$$

$$S_i^0 = ST_i^0 \quad \text{for each } i \in V \quad (14)$$

$$C_i^0 = CT_i^0 \quad \text{for each } i \in V \quad (15)$$

$$C_n^0 = Z^0 \quad (16)$$

The constraint set (13) shows the binary variables which are equal to 1 in the initial solution.

The constraint set (14) shows the start time values of tasks with respect to the initial solution.

The constraint set (15) shows the completion time values of tasks with respect to the initial solution.

The constraint (16) shows the objective function value obtained in the initial solution.

Also the relative and absolute gaps are set to zero.

Example solution on an instance

The integer programming model is applied on an instance of the problem, $1|outtree(l < p); p_i = p; r_i|C_{max}$. Instance and the solution are given in the next example.

Instance: Let p be equal to 2 and l be equal to 1. Precedence graph and release time values of the scheduling problem are given in Figure 4.12.

Corresponding network instance is shown in Figure 4.13. Let node n be shown by *end*. In this graph, $\{(0,1), (0,2), (1,3), (1,4), (1,5), (2,6), (2,7), (5,8), (5,9), (6,10), (7,11), (7,12), (11,13), (11,14), (11,15), (3, end), (4, end), (8, end), (9, end), (10, end), (13, end), (14, end), (15, end)\} \in A$. $\{(1,2), (3,4), (3,5), (3,6), (3,7), (4,5), (4,6), (4,7), (5,6), (5,7), (1,6), (1,7), (2,3), (2,4), (2,5), (8,9), (8,10), (8,11), (8,12), (9,10), (9,11), (9,12), (10,11), (10,12), (11,12), (3,8), (3,9), (3,10), (3,11), (3,12), (4,8), (4,9), (4,10), (4,11), (4,12), (5,10), (5,11), (5,12), (6,8), (6,9), (6,11), (6,12), (7,8), (7,9), (7,10), (13,14), (13,15), (14,15), (8,13), (8,14), (8,15), (9,13), (9,14), (9,15), (10,13), (10,14), (10,15), (12,13), (12,14), (12,15)\} \in E$.

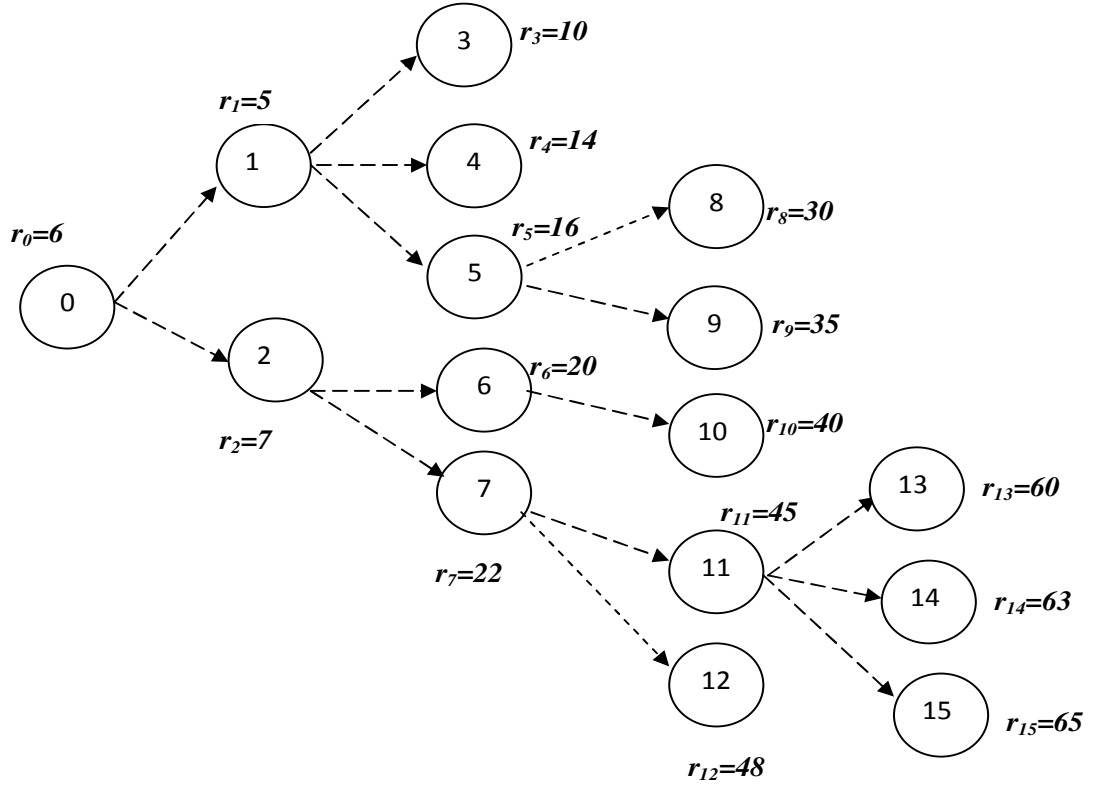


Figure 4.12: Scheduling problem instance -precedence graph and release times for $1|outtree(l < p); p_i = p; r_i|C_{max}$ -

Figure 4.14 shows the optimal solution for the given instance.

4.5 Computational results for $1|outtree(l < p); p_i = p; r_i|C_{max}$

Polynomial algorithm has been run in C and integer programming model of the problem has been tested by GAMS. Randomly generated instances have been solved by both methods. For tested instances, $n \in \{500, 1000, 1500\}$ where n is the total number of tasks to be scheduled. For each n two different precedence graph is generated

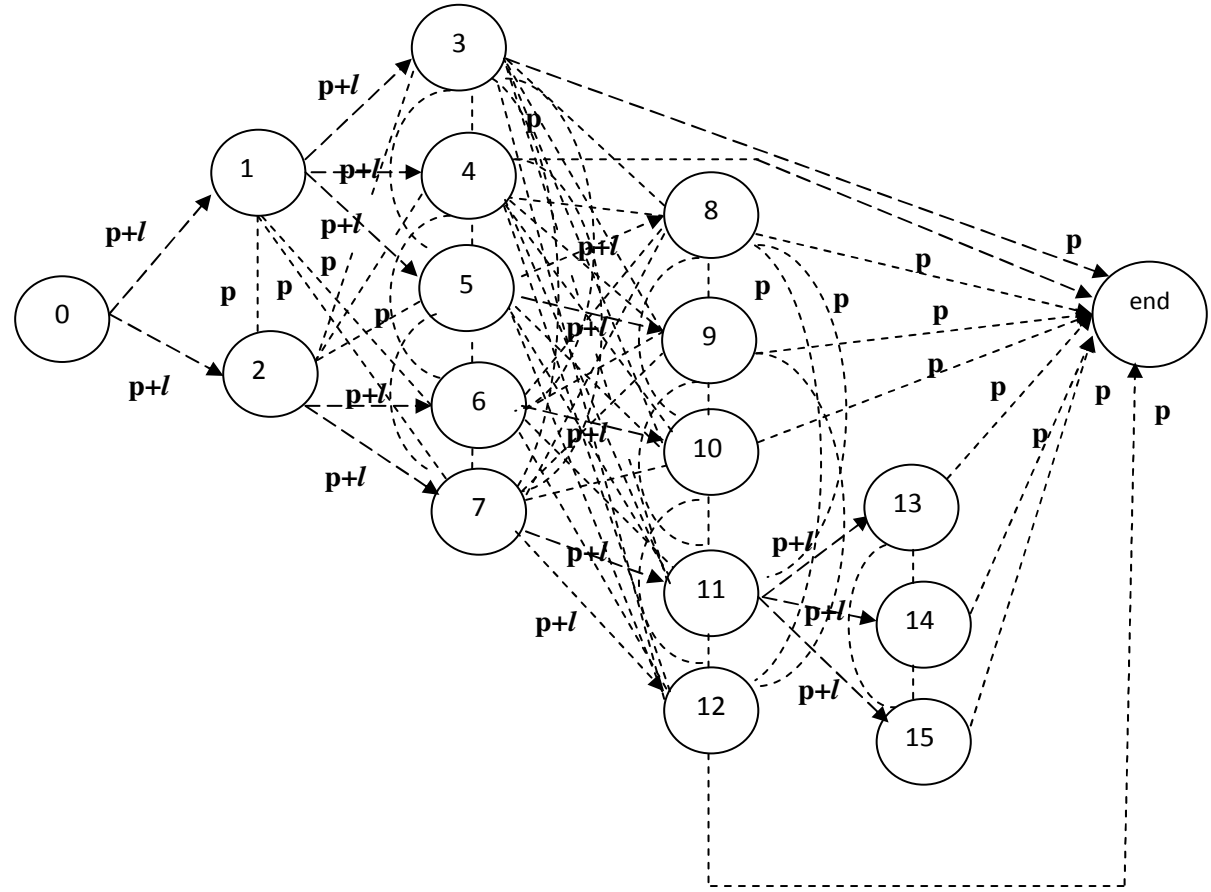


Figure 4.13: Instance for integer programming model for $1|outtree(l < p); p_i = p; r_i|C_{max}$

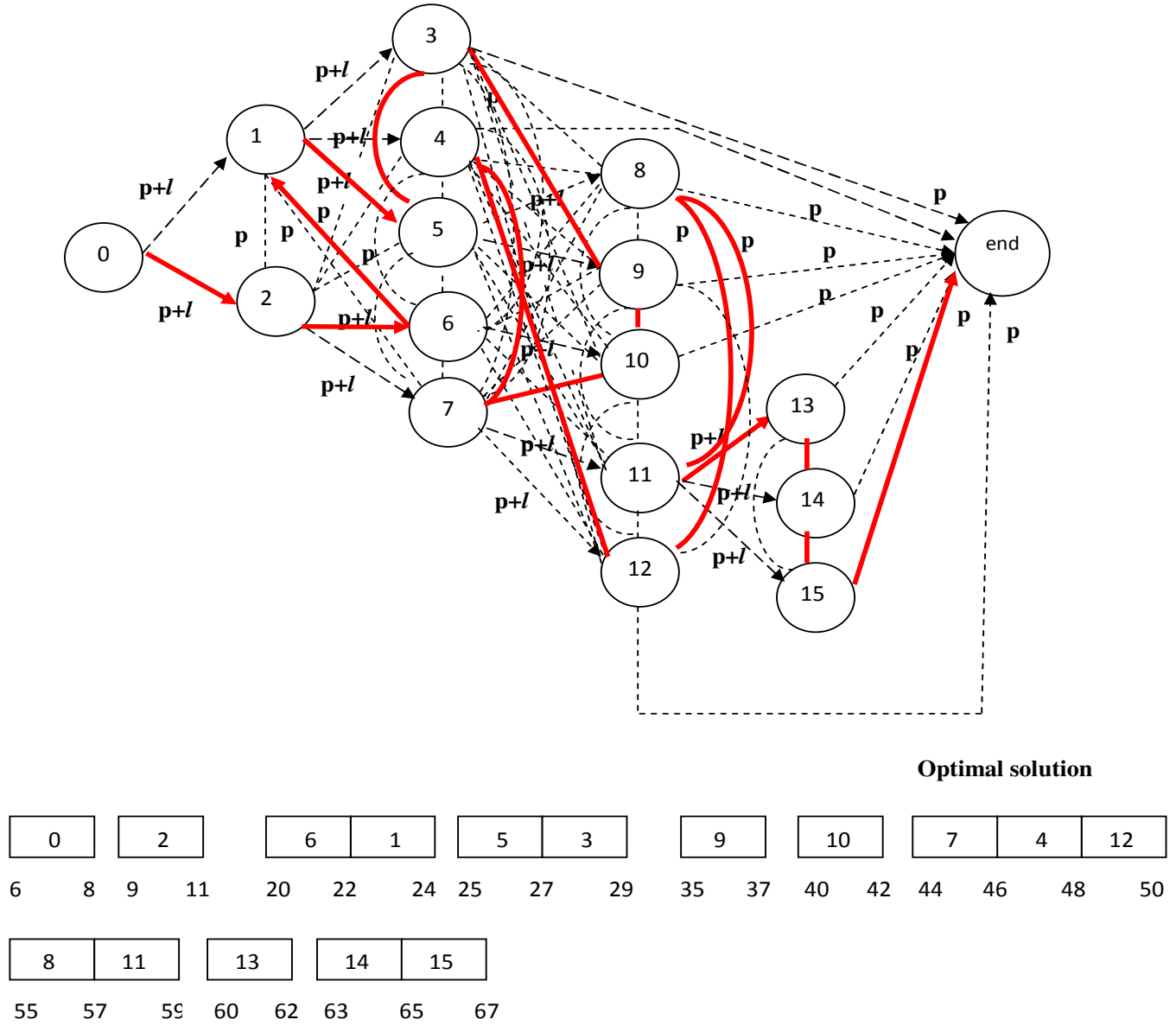


Figure 4.14: Solution for the instance for $1|outtree(l < p); p_i = p; r_i|C_{max}$

by changing the range of numbers from which *number of immediate successors* of each node is randomly chosen. This range of numbers is shown by I_s and taken as $[0,5]$ and $[0,10]$ to generate different precedence graphs for computational analysis. In the former case, each task has a random number of immediate successors between 0 and 5 and in the latter case, each task has a random number of immediate successors between 0 and 10. For $I_s = [0,5]$, number of precedence levels are larger than the case with $I_s = [0,10]$ if n is fixed. This means that for a certain value of problem size, when $I_s = [0,5]$ timelag restrictions are stronger. For release time and timelag generation, τ_1 and τ_2 are used as explained in the case for $1|chains(l < p); p_i = p; r_i|C_{max}$. To generate the release times for $1|outtree(l < p); p_i = p; r_i|C_{max}$, τ_1 is taken as 0.8 and 8. Moreover, $p = 20$. For data generation, an algorithm has been coded in C.

For each combination of different values of parameters, 10 instances are generated. Algorithms and models have been run on a 2.5 GHz PC with Intel Core i5 processor and 4GB of RAM. The polynomial algorithm has been coded in C. GAMS with the solver ILOG CPLEX 12.1.0 has been used to solve the integer programming model. Problem size, n , is the total number of tasks to be scheduled. Results are given in Table 4.1. In the table, the minimum, average and maximum solution time over ten instances is given for each combination of problem parameters and for the tests in C and GAMS. For GAMS, run time was limited by 300 seconds. Furthermore, the minimum, average and maximum gap at root and termination over ten instances in GAMS is presented for each parameter combination. Gaps are the relative gaps which are obtained by the formulation, $(best\ integer\ solution - best\ bound)/best\ integer\ solution$. For GAMS, initial solution is the output of the polynomial algorithm.

Computational results show that the $O(n^3)$ algorithm solves the instances in significantly small times. Let $n = 500$, $n = 1000$ and $n = 1500$ be shown as $n = x$, $n = 2x$ and $n = 3x$. Then, for the $O(n^3)$ algorithm, in the worst case, the solution times of the instances with $n = x$, $n = 2x$ and $n = 3x$ are expected to be proportional to 1, 8 and 27, respectively. However, in computational results, the solution times of $n = 2x$ and $n = 3x$ are proportional to the values which are less than 8 and 27.

Table 4.1: Computational results for $1|outtree(l < p); p_i = p; r_i|C_{max}$

n	I_s	τ_1	τ_2	C_{max}	Elapsed time (sec)				IP model				Initial gap for IP model				Final gap for IP model			
					min	avg	max		min	avg	max		min	avg	max		min	avg	max	
500	[0-5]	0.8	0.1	16487.20	0.04	0.04	0.05		17.67	130.84	300.00		0.00	0.06	0.21		0.00	0.06	0.21	
			0.6	16496.70	0.03	0.04	0.05		17.85	130.81	300.00		0.00	0.06	0.21		0.00	0.06	0.21	
		8	0.1	80625.50	0.05	0.06	0.09		19.48	210.19	300.00		0.00	0.01	0.05		0.00	0.01	0.05	
			0.6	80632.40	0.04	0.05	0.07		18.06	184.70	300.00		0.00	0.01	0.05		0.00	0.01	0.05	
	[0-10]	0.8	0.1	13256.70	0.04	0.05	0.07		16.99	271.70	300.00		0.00	0.07	0.14		0.00	0.06	0.14	
			0.6	13265.30	0.03	0.05	0.06		18.85	271.89	300.00		0.00	0.07	0.14		0.00	0.06	0.14	
		8	0.1	79925.70	0.05	0.05	0.06		17.05	130.52	300.00		0.00	0.00	0.01		0.00	0.00	0.01	
			0.6	79929.90	0.04	0.05	0.06		17.32	123.92	300.00		0.00	0.00	0.01		0.00	0.00	0.01	
1000	[0-5]	0.8	0.1	34561.80	0.15	0.16	0.18		82.34	129.94	300.00		0.00	0.05	0.29		0.00	0.05	0.29	
			0.6	34571.80	0.15	0.17	0.18		80.62	132.34	300.00		0.00	0.05	0.29		0.00	0.05	0.29	
		8	0.1	164405.20	0.15	0.16	0.17		89.63	278.96	300.00		0.00	0.03	0.09		0.00	0.03	0.09	
			0.6	164413.20	0.15	0.17	0.19		137.95	283.80	300.00		0.00	0.03	0.09		0.00	0.03	0.09	
	[0-10]	0.8	0.1	32193.10	0.13	0.15	0.20		80.67	85.04	111.18		0.00	0.00	0.00		0.00	0.00	0.00	
			0.6	32203.10	0.14	0.16	0.20		80.36	81.79	83.96		0.00	0.00	0.00		0.00	0.00	0.00	
		8	0.1	162206.20	0.14	0.15	0.19		125.30	248.30	300.00		0.00	0.01	0.04		0.00	0.01	0.04	
			0.6	162212.90	0.13	0.16	0.20		120.76	246.47	300.00		0.00	0.01	0.04		0.00	0.01	0.04	
1500	[0-5]	0.8	0.1	47752.30	0.38	0.41	0.48		-	-	-		-	-	-		-	-	-	
			0.6	47763.30	0.36	0.37	0.39		-	-	-		-	-	-		-	-	-	
		8	0.1	242632.20	0.34	0.36	0.43		-	-	-		-	-	-		-	-	-	
			0.6	242637.20	0.35	0.36	0.37		-	-	-		-	-	-		-	-	-	
	[0-10]	0.8	0.1	43307.30	0.31	0.33	0.39		-	-	-		-	-	-		-	-	-	
			0.6	43315.30	0.30	0.33	0.36		-	-	-		-	-	-		-	-	-	
		8	0.1	239913.80	0.31	0.33	0.42		-	-	-		-	-	-		-	-	-	
			0.6	239915.80	0.31	0.32	0.33		-	-	-		-	-	-		-	-	-	

For integer programming model, gaps are larger than the gaps for scheduling problem with chain like precedence relations in Chapter 3. In outtree, to eliminate a timelag by using different scheduling options is harder compared to the precedence graph, chains. In GAMS, dual solution is usually generated by considering the release times rather than the timelags which occur depending on the scheduling order. Also, the lower bound, LB_2^0 , is computed by relaxing nearly all timelag constraints. Therefore, in integer programming model for $1|outtree(l < p); p_i = p; r_i|C_{max}$, there is a large difference between a feasible solution and the dual bound which is the reason of large gaps.

Comparing the gaps only for the instances of $1|outtree(l < p); p_i = p; r_i|C_{max}$, it is seen that for the instances with $I_s = [0, 10]$ gaps were smaller than the gaps for the instances with $I_s = [0, 5]$. The reason of this difference is that, unavoidable timelags are more frequently seen in the optimal solution of an instance with $I_s = [0, 5]$, which increases the gap between the primal and dual solution.

Chapter 5

A BRANCH AND BOUND ALGORITHM FOR SCHEDULING CHAINS WITH TIMELAGS

In this chapter, an NP-hard scheduling problem is considered. The scheduling problem is denoted as $1|chains(l);r_i|\sum C_i$ by the three-field notation. In this problem, a number of tasks with arbitrary processing times, p_i , arbitrary release times, r_i , the precedence graph, $chains$, and the constant nonnegative timelag, l , has to be scheduled on a single machine to minimize the sum of completion times. The timelag, l , can take any value but it is the same for all tasks. Note that unlike the problems in Chapters 3 and 4, $p_i \neq p$ and there is no restriction on the size of l . Also the objective function is different. An easier case of this problem where the release time constraint is removed is known to be strongly NP-hard (Brucker and Knust, 1999). Thus, a branch and bound algorithm is suggested to solve the problem $1|chains(l);r_i|\sum C_i$.

5.1 Problem definition of $1|chains(l);r_i|\sum C_i$

The problem which is denoted by $1|chains(l);r_i|\sum C_i$ is to schedule n tasks with precedence constraints, $chains$, constant timelags, l , unequal processing times p_i and release times, r_i , on a single machine. Aim is to minimize the sum of completion times, $\sum C_i$. Each task has at most one immediate predecessor and one immediate successor in a chain like precedence graph which was also described in Chapter 3. When two consecutive tasks in a chain are to be scheduled successively or nonsuccessively, the earliest start time of the successor task has to be greater than or equal to the maximum of its release time and the time corresponding to sum of completion time of its predecessor and the timelag. Simultaneous processing and preemption are not allowed.

In this study, let K be the total number of chains and n_k be the number of tasks in chain k where $k = 0, \dots, K - 1$. $r_{k,i}$, $p_{k,i}$ and $c_{k,i}$ show the release time, processing time and completion time of task i , $i = 0, \dots, n_k - 1$, in chain k , $k = 0, \dots, K - 1$, respectively. With respect to this notation, the smaller the index i of a chain k , the larger priority the task represented by that index has.

5.2 Branch and bound algorithm for $1|chains(l); r_i| \sum C_i$

In the proposed algorithm, branch and bound tree is composed of partially scheduled tasks. At each node, partial solution is computed by tracking the path containing the ancestor nodes in the preceding levels till the root node of the tree. That is, for any node, the partial schedule for a group of tasks is known. To find the lower and upper bound at any node, makespan and total completion time of the partial schedule for that node are taken as inputs. For the unscheduled tasks, a relaxed schedule is found and combined with the partial schedule of the related node to set a lower bound. To determine the upper bound, a feasible schedule is obtained for the unscheduled tasks to complete the partial schedule. Feasible solution is the minimum of four solutions obtained by different heuristic algorithms. Each of these algorithms iteratively updates the precedence graph and the earliest start times of the candidate tasks to be scheduled with respect to release time and timelag constraints and applies a selection strategy by also checking the processing times of the candidate tasks. These heuristic algorithms differ from one another by their selection strategies. The branching scheme considers only the candidate tasks from all chains, which eliminates the feasibility check. Since only one task with the highest priority degree in a chain can be scheduled, number of successor nodes of any node in the tree is smaller than or equal to the total number of chains. To explore the solution space, depth first search, breadth first search, best lower bound first search and best upper bound first search are compared. Also, the classical pruning strategy and one problem specific dominance rule are employed.

5.2.1 Branching scheme

The node on the top of the branch and bound tree shows the case that none of the tasks are scheduled. Any descendant node represents the partial solution where one more task is scheduled in addition to the partial schedule that was built for the ancestor node. In the proposed branching scheme, only the nodes representing the assignment of feasible candidates are located as the immediate descendants of any node. By this way, feasibility check is eliminated. Feasibility is ensured by scheduling a task for which all the predecessors were processed before. Therefore, the children of a node show the partial solutions obtained by assigning the tasks having the highest priority degrees among the unscheduled ones. Due to the structure of the precedence graph, *chains*, exactly one task from each chain is allowed to be scheduled. Thus, number of immediate descendants of any node of the branch and bound tree is smaller than or equal to the total number of chains which is shown by K . Let any node show the assignment of task i , $i = 0, \dots, n_k - 1$, from chain k , $k = 0, \dots, K - 1$. Then, the children of that node show the assignment of the next feasible task, $i + 1$, from the same chain k , and the feasible tasks from different chains.

Formally, let A and B be any two nodes in the branch and bound tree and D_j^A and D_j^B where $j = 0, \dots, K - 1$ be the sets of immediate descendants of nodes A and B . Consider the case that $B \in D_j^A$, $j = 0, \dots, K - 1$. Let node B represent the task i , $i = 0, \dots, n_k - 1$, of chain k , $k = 0, \dots, K - 1$. Then the nodes included in D_j^B , $j = 0, \dots, K - 1$, represent the following tasks: If $j = k$, where k was represented by node B , node j shows the task $i + 1$ of chain k ; if $j \neq k$, where k was represented by node B , addressed tasks in $D_{j \neq k}^A$, $j = 0, \dots, K - 1$, are copied by the nodes in $D_{j \neq k}^B$, $j = 0, \dots, K - 1$.

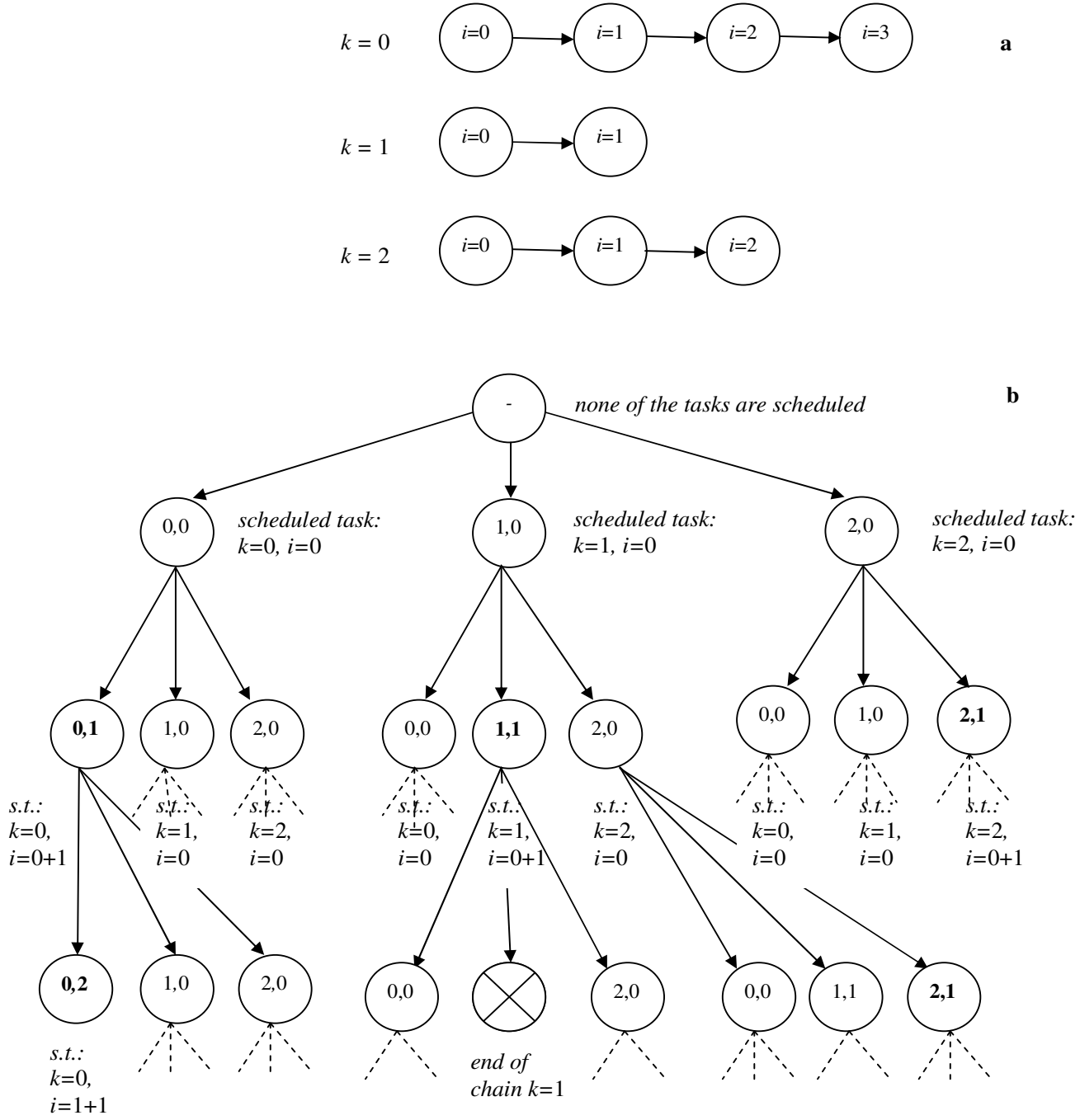
Figure 5.1 contains an instance of *chains* and the corresponding branching scheme. In Figure 5.1.a, there is an instance of the precedence graph, *chains*, where $K = 3$, $n_0 = 4$, $n_1 = 2$, and $n_2 = 3$. Figure 5.1.b demonstrates the branching scheme for the given instance. Since $K = 3$, each node in the branch and bound tree is branched to generate at most 3 nodes. In the root node of the branch and bound tree, none of

the tasks are scheduled. At the next level, tasks with the highest priority degrees of three chains, task 0 of chain 0, task 0 of chain 1 and task 0 of chain 2, are scheduled at different nodes. Consider one of these nodes which shows the partial schedule built by task 0 of chain 1. Succeeding this node, three nodes are created. These nodes respectively show the partial solutions by assigning task 0 of chain 0, task 1 of chain 1 and task 0 of chain 2 following the task 0 of chain 1. Other nodes have the same structure.

5.2.2 Partial solutions

For each node a partial solution is computed. To find a partial solution, first a partial schedule is constructed by combining the partial schedule of the preceding node and the task represented by the current node. Thus, a partial schedule for a node is composed of the tasks on the path created by the selected branches till reaching that node. Makespan of the partial schedule is found by considering the makespan of the partial schedule at the preceding node, release time and processing time of the task shown by the current node and timelag constraint. Partial solution at the current node is the sum of completion times of the tasks of the partial schedule. It is obtained by adding the makespan found for the current node to the partial solution - partial sum of completion times- of the preceding node.

To give a formal definition, let A and B be any two nodes in the branch and bound tree and node B be one of the immediate descendants of node A . Then, C_{max}^A and C_{max}^B are the partial makespans recorded at nodes A and B which are in fact the completion times of the tasks shown by nodes A and B . Also, $\sum C_i^A$ and $\sum C_i^B$ are the sum of completion times of the related partial schedules. Any node represents the task i , $i = 0, \dots, n_k - 1$, of chain k , $k = 0, \dots, K - 1$ as described before. If the nodes A and B represent any consecutive tasks $i = \alpha$ and $i = \alpha + 1$, of the same chain $k = \beta$, respectively, then $C_{max}^B = \max(C_{max}^A + l, r_{\beta, \alpha+1}) + p_{\beta, \alpha+1}$ where $C_{max}^A = c_{\beta, \alpha}$ and $c_{\beta, \alpha}$ is the completion time of task α of chain β represented by node A . Note that when the task α of chain β is scheduled, the release time of task $\alpha + 1$, $r_{\beta, \alpha+1}$, is

Figure 5.1: Instance of *chains* and the branching scheme for $1|chains(l); r_i| \sum C_i$

replaced by the time, $\max(c_{\beta,\alpha} + l, r_{\beta,\alpha+1})$, for the remaining iterations. If the nodes A and B represent any two tasks from different chains, such as the task $i = \alpha$ of chain $k = \beta$ and task $i = \zeta$ of chain $k = \theta$, respectively, then $C_{max}^B = \max(C_{max}^A, r_{\theta,\zeta}) + p_{\theta,\zeta}$ where $C_{max}^A = c_{\beta,\alpha}$. Finally, the partial solution for the node B is obtained by the equation $\sum C_i^B = \sum C_i^A + C_{max}^B$ where C_{max}^B is also shown by $c_{\theta,\zeta}$.

5.2.3 Lower bounds

To find a lower bound at any node, the partial schedule at that node is taken as input and the remaining tasks are assigned into the following positions of the schedule by relaxing some of the problem constraints. The lower bound is equal to sum of completion times for the whole schedule plus some additional values obtained by repair methods.

If the release time and timelag constraints are relaxed, we obtain the problem $1|chains; p_i| \sum C_i$. It is known that the scheduling problem $1|chains; p_i| \sum w_i C_i$ where $\sum w_i C_i$ is the weighted sum of completion times can easily be solved to optimality (Pinedo, 2008). If all weights are assumed to be equal to 1, the solution method for $1|chains; p_i| \sum w_i C_i$ can be adjusted to solve the problem $1|chains; p_i| \sum C_i$ to optimality as follows: Let weights for the jobs be denoted by $w_{k,i}$, $k = 0, \dots, K - 1$, $i = 0 \dots n_k$.

- (i) Obtain a ρ - factor for each chain k , $k = 0, \dots, K - 1$, where

$\rho_k = \max_{0 \leq h < n_k} (\sum_{0 \leq i \leq h} w_{k,i} / \sum_{0 \leq i \leq h} p_{k,i})$ and flag the task h which determines the ρ - factor.

- (ii) If the machine is free, select among the remaining chains with the highest ρ - factor, $\max_{0 \leq k < K} \rho_k$.
- (iii) Process the tasks from that chain one after another up to and including the job that determines its ρ - factor with no interruption from other chains.
- (iv) Repeat the steps until all tasks are scheduled.

Adjusted algorithm is applied for scheduling the remaining tasks after the partial schedule at any node of the branch and bound tree to set a lower bound. To improve this bound, two additional values, Δ_1 and Δ_2 , are obtained.

$\Delta_1 = [(\nu - 1) \cdot \nu \cdot l]/2$ where l is the duration of the timelag and ν is a variable found as follows : Let $k = *$ be the index of the longest chain for which $n_k = \max_{0 \leq k < K} n_k$. Then, $\nu = n_* - \sum_{0 \leq k < K} n_{k \neq *}$ if $n_* - \sum_{0 \leq k < K} n_{k \neq *} > 0$ and $\nu = 0$ if $n_* - \sum_{0 \leq k < K} n_{k \neq *} \leq 0$. Note that, in these formula, K and n_k are the number of chains and number of tasks in chain k when the tasks which were used to construct the partial schedule are removed from the precedence graph.

$\Delta_2 = [\min_{0 \leq k < K} [\max(C_{max}^P, r_{k,0}^*) - C_{max}^P]] \cdot (n - n^P)$ where C_{max}^P is the makespan of the partial schedule at any node and n^P is the number of tasks contained in that partial schedule while n is the total number of tasks in the problem instance. Also note that, $r_{k,0}^*$ denotes the release time of task 0 of chain k when the chains are updated after excluding the tasks of the partial schedule from the precedence graph.

Δ_1 and Δ_2 help to improve the relaxed solution by adding the unavoidable delays which are caused by the timelag and release time constraints to the second part of the complete schedule which was constructed by relaxing the timelag and release time constraints. Those delays may be larger, thus the improvement of the relaxed solution may be larger than $\Delta_1 + \Delta_2$ but cannot be smaller.

To sum up, the lower bound at any node of the branch and bound tree is obtained by the following equality : $lowerbound = \sum C_i^P + \sum C_i^R + \Delta_1 + \Delta_2$. In this equality, $\sum C_i^P$ is the sum of completion times of the partial schedule which has the makespan of C_{max}^P . Note that none of the constraints is relaxed to compute $\sum C_i^P$. On the other hand, $\sum C_i^R$ is the sum of completion times of the schedule which starts at time C_{max}^P and is built by scheduling the remaining tasks to optimality by relaxing the release time and timelag constraints. That is, $\sum C_i^R$ is the optimal solution of $1|chains; p_i| \sum C_i$ for a group of tasks which are not included at the partial schedule and which are all forbidden to start earlier than C_{max}^P .

5.2.4 Upper bounds

In this study, upper bound for any node at the branch and bound tree is the minimum of four feasible solutions. Each feasible solution is found by applying a different heuristic algorithm to schedule the remaining tasks after the partial schedule at the associated node.

To find the first feasible solution, algorithm takes the makespan of partial schedule, C_{max}^P , and related solution, $\sum C_i^P$, as inputs and works on the remaining tasks after removing the partially scheduled tasks from the precedence graph. Algorithm iteratively removes the scheduled task from the precedence graph and determines the candidate tasks for the current position of the schedule. When one task is removed from a chain k , number of tasks in that chain is decreased by one, thus release time and processing time of task $i + 1$ in chain k become the release time and processing time of task i in chain k , respectively. Therefore, at each iteration, $r_{k,0}^u$ and $p_{k,0}^u$, $k = 0, \dots, K - 1$, show the release time and processing time, respectively, of the task with the highest priority degree among the remaining tasks in chain k after updating the chain. Due to the iteratively gained information, algorithm updates the earliest start times of the candidate tasks by the following formula : Let $est_{k,0}^u$ be the earliest start time of task 0 in chain k where $k = 0, \dots, K - 1$. $est_{k,0}^u = \max(c^u + l, r_{k,0}^u)$ if the chain index k is the same as the chain index $k = \alpha$ of the task that was scheduled in the former iteration while $est_{k,0}^u = \max(c^u, r_{k,0}^u)$ if the chain index k is different than the one of the task that was scheduled in the former iteration. In these formula, c^u is the completion time of the formerly scheduled task which is equal to C_{max}^P at the beginning of the algorithm and iteratively updated with respect to chosen task. Note that except for the task scheduled at current iteration, the release times, $r_{k,0}^u$, are updated by the associated earliest start times, $est_{k,0}^u$, which will be used for the remaining iterations. Algorithm applies the following selection strategy to schedule a task at the current iteration: If $est_{\alpha,0}^u + p_{\alpha,0}^u > \min_{0 \leq k < K} r_{k \neq \alpha,0}^u$, the task with $\min_{0 \leq k < K} [est_{k \neq \alpha,0}^u + p_{k \neq \alpha,0}^u]$ is found. If that task is unique, it is chosen to be scheduled. If not, among the tasks with $est_{k,0}^u + p_{k,0}^u = \min_{0 \leq k < K} [est_{k,0}^u + p_{k,0}^u]$, the task with

the maximum $p_{k,0}^u$ is found. If that task is unique, it is scheduled. If not, the lengths of the chains to which the tasks with that maximum $p_{k,0}^u$ belong are compared and the task of the longest one is selected. If $est_{\alpha,0}^u + p_{\alpha,0}^u \leq \min_{0 \leq k < K} r_{k \neq \alpha,0}^u$, the task 0 of chain α is chosen to be scheduled. After the selection phase algorithm computes the completion time of the chosen task, c , updates the related data and sum of completion times and repeats the steps.

This first heuristic algorithm, first checks sum of the earliest start time and processing time for the available tasks in the chains except for the one from which a task one scheduled one iteration before, to make a selection. Because, in every step, it tries to eliminate the potential timelag at current position. However if the task with the minimum updated release time value among tasks of the chains which do not require a timelag to be scheduled is significantly large, -larger than the sum of the earliest start time and the processing time of the task that necessitates a timelag to be scheduled-, then the task of the chain which requires a timelag is allowed to be scheduled at the current position which is an extreme case. If this is not the case, the task with the smallest sum of earliest start time and processing time among the chains with no timelag restriction is scheduled. If there are several tasks for which the sum of earliest start time and processing time is equal to the smallest sum, then the one with the largest processing time is chosen. The aim is to eliminate the tasks with large processing times earlier if they will create the same completion time with their alternatives at the current iteration. By this way, completion times will be smaller at further positions and the objective function, sum of completion times, will be better. If also the processing times among the last compared group are equal then the lengths of chains are compared. Choosing the longer chains sometimes give better schedule lengths when we have timelag constraints. But, the release time constraints and objective function, sum of completion times, together with arbitrary processing times makes the effect of this strategy weaker. Therefore in this heuristic algorithm, lengths of chains is the least important criterion to make a selection.

To find the second feasible solution, as in the computation of the first feasible

solution, algorithm takes the information of partial schedule as input, iteratively determines the candidate tasks, earliest start times for the candidate tasks, completion time of the scheduled task and updates the precedence graph. However, a different selection strategy is applied to schedule a task at each iteration which is described as follows: If $est_{\alpha,0}^u + p_{\alpha,0}^u > \min_{0 \leq k < K} r_{k \neq \alpha,0}^u$, ρ - factor, ρ_k , $k = 0, \dots, K - 1$, is found for each chain $k \neq \alpha$ in the updated precedence graph. How to find a ρ - factor is explained in Subsection 5.2.3. The task having the highest priority degree in the chain for which $\rho_{k \neq \alpha} = \max_{0 \leq k < K} \rho_{k \neq \alpha}$ is obtained and flagged. If the release time of that task is smaller than or equal to $\min_{0 \leq k < K} [est_{k \neq \alpha,0}^u + p_{k \neq \alpha,0}^u]$, the task is chosen to be scheduled at the current iteration. Else, the selection strategy of the first feasible solution method is used to select a task for the current position of the schedule. If $est_{\alpha,0}^u + p_{\alpha,0}^u \leq \min_{0 \leq k < K} r_{k \neq \alpha,0}^u$, the task 0 of chain α is selected. Steps are repeated until all tasks are scheduled.

In this second heuristic algorithm, in addition to the strategy of first heuristic algorithm, also a method which evaluates the processing times of the unavailable tasks is integrated. Due to the precedence graph, these unavailable tasks following the available tasks will become available at further iterations depending on the choice for current iteration. Sometimes the processing times of these successor tasks may be in small sizes even if the predecessor task has a large processing time. By choosing this predecessor with large processing time at the current position, successors with small processing times become free at the earlier iterations which may lead to a smaller objective.

The algorithm to find the third feasible solution differs from the others by the selection strategy. To select a task for the current position of the schedule, the task with $\min_{0 \leq k < K} est_{k,0}^u$ is found. If that task is unique, it is scheduled. If not, among the tasks with $est_{k,0}^u = \min_{0 \leq k < K} est_{k,0}^u$, the one with the minimum $p_{k,0}^u$ is obtained. If only one task has this processing time value, it is scheduled. If not, among the tasks with the processing time equal to minimum $p_{k,0}^u$ and earliest start time $est_{k,0}^u = \min_{0 \leq k < K} est_{k,0}^u$, the one belonging to the longest chain is scheduled. All

steps are repeated until all tasks are scheduled.

Unlike the first and second heuristics, third heuristic algorithm considers to schedule the task that requires a timelag if it has the smallest earliest start time among the available tasks as an option. Also at first stage only the earliest start times are compared instead of the sum of earliest start times and processing times.

To find the fourth feasible solution, another selection strategy is applied to schedule a task. First, the task with $\min_{0 \leq k < K} [est_{k \neq \alpha, 0}^u + p_{k \neq \alpha, 0}^u]$ is found. When $est_{\alpha, 0}^u \geq \min_{0 \leq k < K} [est_{k \neq \alpha, 0}^u + p_{k \neq \alpha, 0}^u]$, if the task with $\min_{0 \leq k < K} [est_{k \neq \alpha, 0}^u + p_{k \neq \alpha, 0}^u]$ is unique, it is chosen to be scheduled. If not, among the tasks with $est_{k, 0} = \min_{0 \leq k < K} [est_{k \neq \alpha, 0}^u + p_{k \neq \alpha, 0}^u]$, the task from the longest chain is scheduled. When $est_{\alpha, 0}^u < \min_{0 \leq k < K} [est_{k \neq \alpha, 0}^u + p_{k \neq \alpha, 0}^u]$, task 0 of chain α is chosen. Steps are repeated until all tasks are scheduled.

In the fourth heuristic, choosing the tasks of longer chains or task of the chain from which a task was scheduled in the former iteration is more likely compared to first and second heuristic.

Depending on the instance, any heuristic may give the best upper bound as the problem parameters are highly complex. To summarize, the upper bound at any node is equal to the minimum of four feasible solutions which are computed by different heuristic algorithms.

5.2.5 Exploring the solution space

Best lower bound first, best upper bound first, breadth first and depth first search methods are tested to explore the solution space. Solutions are pruned either by local optimality or by detecting a solution node having a lower bound larger than the best known upper bound. Also, a dominance rule is proposed to prune the nodes. If two partial schedules contain the same number of tasks, have the same tasks scheduled in different orders but end with the same task, the partial schedule with larger sum of completion times and makespan is dominated. Therefore, the associated node is pruned.

5.3 Integer programming model for $1|chains(l); r_i| \sum C_i$

The problem is formulated as an integer programming model with disjunctive constraints. Two additional dummy tasks are introduced and all tasks are shown by the vertices of a graph $G(V, A, E)$ where V is the set of vertices, A is the set of directed links (i, j) and E is the set of undirected links (i, j) . Dummy tasks are denoted by the vertices 0 and $n + 1$. If task j is an immediate successor of i , then $(i, j) \in A$. If j is such a task that has the highest priority in any chain and $i = 0$, then $(i, j) \in A$. If i is such a task that has the lowest priority in any chain and $j = n + 1$, then $(i, j) \in A$. If i and j belong to different chains, then $(i, j) \in E$. Also note that $i \prec j$ means that i precedes j which is shown on the graph either by the link $(i, j) \in A$ or by an indirect relation such as $(i, h) \in A$, $(h, k) \in A$ and $(k, j) \in A$ where i, h, k, j are the partially ordered tasks in a chain.

Notation

- Sets and Parameters :

O_i : Set of alternative destinations, $j \in V$ where $(i, j) \in A$ or $(i, j) \in E$, for any $i \in V$.

I_j : Set of alternative sources, $i \in V$ where $(i, j) \in A$ or $(i, j) \in E$, for each $j \in V$.

r_i : Release time for each task $i \in V$. Note that $r_0 = 0$, $r_{n+1} = 0$ and other tasks have positive values.

p_i : Processing time for each task $i \in V$. Note that $p_0 = 0$, $p_{n+1} = 0$ and other tasks have positive values.

- Decision variables :

x_{ij} : Binary variable for the link between $i \in V$ and $j \in V$. $x_{ij} = 1$ if task j is assigned immediately after task i . $x_{ij} = 0$ in the other case.

S_i : Start time of task i which is nonnegative and integer.

C_i : Completion time of task i which is nonnegative and integer.

- Objective function :

$\sum C_{i \leq n}$: Sum of completion times for $0 \leq i \leq n$. Note that $i = 0$ and $i = n + 1$ are the dummy nodes where $C_0 = 0$ and $C_{n+1} > 0$. Thus C_{n+1} is ignored while

computing the objective function value.

Model

$$\text{minimize } \sum C_{i \leq n} \quad (1)$$

$$S_i \geq r_i \quad \text{for each } i \in V \quad (2)$$

$$C_i = S_i + p_i \quad \text{for each } i \in V \quad (3)$$

$$S_j > S_i + p_i + \sum_{i \prec k \prec j} p_k \quad \text{for each } i, j \in V \text{ with } i \prec j \quad (4)$$

$$S_j \geq C_i + l \quad \text{for each } (i, j) \in A \text{ where } i \neq \text{start} \text{ or } j \neq n+1 \quad (5)$$

$$S_j + M(1 - x_{ij}) \geq C_i \quad \text{for each } (i, j) \in E, (0, j) \in A, (j, n+1) \in A \quad (6)$$

$$\sum_{j \in O_i} x_{ij} = 1 \quad \text{for each } i \in V \text{ where } i \neq n+1 \quad (7)$$

$$\sum_{i \in I_j} x_{ij} = 1 \quad \text{for each } j \in V \text{ where } j \neq 0 \quad (8)$$

$$C_{n+1} \geq C_i \quad \text{for each } i \in V \quad (9)$$

$$\sum x_{ij} + x_{ji} \leq 1 \quad \text{for each } (i, j) \in E \quad (10)$$

Objective is to minimize sum of completion times which is shown in (1).

The constraint set (2) requires that start time of a task must be greater than or equal to its release time.

The equation set in (3) computes the completion time of a task which is equal to its start time plus its processing time.

With respect to constraint set (4), if task j is any successor of task i , start time of task j must be larger than start time of task i plus processing time of task i plus the sum of processing times of the tasks which are both the successors of task i and predecessors of task j .

The constraint set (5) shows the timelag constraint. Accordingly, if task j is the immediate successor of task i , start time of task j must be greater than or equal to the completion time of task i plus the timelag value.

The constraint set (6) is a disjunctive constraint. This constraint is defined to ensure that the start time of a task j is greater than or equal to completion time of task i if task j is chosen to be scheduled right after task i .

According to the constraint set (7), in the optimal path, number of links emanating from a node must be equal to one except for the node $n+1$.

According to the constraint set (8), in the optimal path, number of links entering into a node must be equal to one except for the node *start*.

The constraint set (9) generates a cut on the completion time at node $n+1$, which must be greater than or equal to the completion time at each node.

In the solution, direction of the flow must be in only one way on any given undirected link. This restriction is defined by the constraint set (10).

5.4 Computational results for the branch and bound and integer programming model for $1|chains(l); r_i| \sum C_i$

Proposed branch and bound algorithm and integer programming model have been run on a 2.5 GHz PC with Intel Core i5 processor and 4GB of RAM. C programming language and GAMS software with ILOG CPLEX 12.1 have been used to implement the branch and bound algorithm and integer programming model. Instances have been randomly generated. For one instance, processing times, p_i , are randomly generated within the range $[1, 20]$. Then, release times, r_i , are produced within the range $[1, \tau^1 \sum p_i]$, and timelag, l , is obtained by the equation, $l = \lceil \tau^2 \sum p_i \rceil / n$ where n is the total number of tasks to be scheduled. For each combination of the problem sizes $n = 11, 20, 22, 32, 39$, the release time coefficients $\tau^1 = 0.2, 0.5, 0.8$ and the timelag coefficients $\tau^2 = 0.5, 1, 2$, ten instances are created. For smaller sizes with $n = 11, 20, 22$, best upper bound first, best lower bound first, breadth first and depth first search strategies have been tested for each instance. Accordingly, branch and bound algorithm with best lower bound first search gives the best results in terms of CPU time. Thus, for large sizes only lower bound first search has been used. In the first experimental results, in addition to classical pruning strategy a dominance rule was used. However, when the experiments were repeated for all instances without the proposed dominance rule, it was seen that the CPU times were much better while there is an increase in the number of visited nodes. Thus, it is inferred that even though the proposed dominance rule is notably efficient on the number of visited nodes, it negatively affects the CPU time. Thus, the proposed

dominance rule is removed from the algorithm. Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8 contain the experimental results for the branch and bound algorithm only with the classical pruning strategy. Table 5.1, Table 5.2 and Table 5.3 show the minimum, the average and the maximum CPU time and number of visited node values overall ten instances for each combination of problem parameters. Table 5.4, Table 5.5, Table 5.6, Table 5.7 and Table 5.8 compare the results of the branch and bound algorithm with best lower bound search strategy and integer programming model for $n = 11, 20, 22, 32, 39$, respectively. With respect to the computational results, for the problem sizes, $n = 11, 20, 22, 32$, where for each of them ten random instances were tested, performance of the branch and bound algorithm was very good. 322 of 360 instances were solved within 29 CPU seconds. For 27 of 360 instances, CPU time was between 30 and 779 seconds. For 7 of 360 instances, CPU time was between 989 and 3441 seconds. For only 4 of 360 instances, CPU time was larger than one hour, which changes between 3949 and 4987 seconds. When, $n = 39$, 64 of 90 instances were solved within one hour, 15 of 90 instances had the CPU time between one and three hours and for 11 of 90 instances CPU time was larger than three hours. Experimental results show that solution time for the branch and bound algorithm is dependent on the properties of the data. This is because the problem parameters are unstable. That is, the objective function, sum of completion times, is highly sensitive to arbitrary release times, arbitrary processing times, timelags, precedence graph, and the relation between these parameters. The objective function sum of completion times is usually better when the tasks with shorter processing times are scheduled at earlier positions of the schedule. However, in the addressed problem, a task with a small processing time may create an excessive timelag, or any task with a large processing time may have attractive successors to schedule. Also the sizes of the time spaces between the release times compared to the sizes of arbitrary processing times and timelags lead to many scheduling alternatives. Computational results for integer programming model were satisfactory only for $n = 11$. For large sizes, optimal solutions were not found in reasonable times.

Table 5.1: Preliminary tests for branch and bound algorithm for $|chains(l); p_i; r_i| \sum C_i$ for $n=11$ and $K=3$

Branch and bound														
CPU second					Number of visited nodes									
τ^1	τ^2	best ub first	best lb first	best lb first	breadth first	best ub first	best lb first	best lb first	breadth first	depth first	best ub first	best lb first	breadth first	depth first
		min	av	max	min	av	max	min	av	max	min	av	max	min
0.2	0.5	0.06	0.07	0.08	0.06	0.07	0.08	0.07	0.08	0.08	0.09	0.08	0.09	0.09
1	0.06	0.07	0.09	0.06	0.07	0.08	0.07	0.07	0.08	0.08	27	89.40	225	27
2	0.06	0.07	0.08	0.06	0.07	0.09	0.07	0.08	0.09	0.09	27	86.10	207	27
0.5	0.5	0.06	0.07	0.08	0.06	0.07	0.08	0.06	0.07	0.08	27	52.50	102	27
1	0.06	0.07	0.07	0.06	0.07	0.07	0.08	0.06	0.07	0.08	27	51	102	27
2	0.06	0.07	0.09	0.06	0.07	0.08	0.06	0.07	0.08	0.08	27	46.80	84	27
0.8	0.5	0.06	0.07	0.08	0.06	0.07	0.08	0.06	0.07	0.07	39	54.90	81	39
1	0.06	0.06	0.07	0.06	0.07	0.07	0.07	0.06	0.07	0.08	24	45.60	78	24
2	0.06	0.07	0.08	0.06	0.07	0.07	0.07	0.06	0.07	0.07	24	46.20	69	24

Table 5.2: Preliminary tests for branch and bound algorithm for $|chains(l); p_i; r_i| \sum C_i$ for $n=22$ and $K=4$

Branch and bound														
CPU second					Number of visited nodes									
τ^1	τ^2	best ub first	best lb first	best lb first	breadth first	best ub first	best lb first	best lb first	breadth first	depth first	best ub first	best lb first	breadth first	depth first
		min	av	max	min	av	max	min	av	max	min	av	max	min
0.2	0.5	0.14	0.90	2.64	0.64	1.31	3.12	0.16	0.99	3.27	0.71	2.31	10.83	676
1	0.14	0.84	2.50	0.64	1.24	2.53	0.16	0.90	2.25	0.73	2.03	7.57	688	5977.70
2	0.14	0.81	2.45	0.65	1.21	2.31	0.14	0.79	2.23	0.73	2.05	6.89	696	5816
0.5	0.5	0.12	0.23	0.54	0.60	0.67	0.79	0.13	0.22	0.31	0.66	0.85	1.36	376
1	0.11	0.26	0.84	0.60	0.71	1.30	0.14	0.26	0.90	0.64	0.88	1.43	407	1591.50
2	0.11	0.25	0.64	0.60	0.70	1.07	0.13	0.28	0.66	0.68	0.87	1.38	384	4724
0.8	0.5	0.10	0.58	1.95	0.60	1.04	2.36	0.13	0.66	1.79	0.63	1.35	2.62	316
1	0.10	0.49	1.19	0.59	0.95	1.61	0.12	0.55	1.11	0.62	1.17	1.84	316	3416
2	0.10	0.36	0.89	0.61	0.83	1.31	0.12	0.40	0.80	0.61	0.97	1.45	316	2396

Table 5.3: Preliminary tests for branch and bound algorithm for $1|chains(l); p_i; r_i| \sum C_i$ for $n=20$ and $K=6$

Branch and bound													
CPU second													
τ^1	τ^2	best ub first				best lb first				breadth first			
		min	av	max	min	av	max	min	av	max	min	av	max
0.2	0.5	0.12	3.79	14.01	0.11	1.88	10.15	0.12	22.96	217.78	0.13	6.38	26.17
	1	0.13	3.85	13.38	0.12	1.93	10.55	0.13	39.07	378.69	0.12	7.04	31.75
	2	0.13	3.98	13.58	0.11	1.89	10.11	0.12	102.69	1015.11	0.12	10.61	67.99
0.5	0.5	0.19	1.54	6.38	0.15	0.57	1.32	0.20	1.84	12.71	0.20	2.61	6.48
	1	0.20	1.60	6.55	0.16	0.53	1.22	0.20	1.99	14.70	0.20	2.70	5.90
	2	0.20	1.64	7.35	0.14	0.53	1.21	0.20	2.17	16.37	0.20	3.27	10.78
0.8	0.5	0.14	4.82	35.73	0.13	3.61	27.62	0.19	5.64	38.22	0.15	14.45	125.77
	1	0.14	3.69	25.47	0.13	2.63	18.70	0.16	4.65	46.47	0.15	9.11	74.38
	2	0.13	3.63	24.46	0.13	2.54	17.66	0.16	5.05	31.49	0.15	8.70	69.65
Branch and bound													
Number of visited nodes													
τ^1	τ^2	best ub first				best lb first				breadth first			
		min	av	max	min	av	max	min	av	max	min	av	max
0.2	0.5	462	25496.40	83958	462	16476	83910	462	65954.4	570438	462	37118.4	128520
	1	462	26405.40	84042	462	16629	83910	462	90659.40	818142	462	38905.80	144270
	2	450	26968.80	84240	450	16569.60	83910	450	144957.7	1361689	450	48341.40	243186
0.5	0.5	1020	11436	47052	888	4620.60	13800	1020	11992.2	82356	1026	18442.20	45120
	1	1020	11757	48834	774	4580.40	13800	990	12764.40	90468	1026	19232.40	41550
	2	1020	12064.80	53484	774	4579.80	13800	990	13599.60	98700	1026	22537.20	67590
0.8	0.5	498	25231.80	167268	498	24568.80	167268	672	31336.40	189630	678	46536.60	341952
	1	498	20688.60	129906	498	20106	129906	654	27638.40	157944	660	36770.40	257418
	2	498	20784.60	125844	498	19738.20	125832	654	29265.60	164310	660	36642	248868

Table 5.4: Comparison of computational results for branch and bound and integer programming model for n=11 and K=3

				CPU second				Number of visited nodes						# of optimal solutions per hour			
τ^1	τ^2	BB with best lb first search			IP			BB with best lb first search			IP			BB with best lb first s.		IP	
		min	av	max	min	av	max	min	av	max	min	av	max				
0.2	0.5	0.06	0.07	0.08	2.33	3.61	5.72	27	88.80	231	15314	25803.1	42576	10	10	10	
	1	0.06	0.07	0.08	1.77	3.75	8.34	27	89.40	225	10367	27441.9	64188	10	10	10	
	2	0.06	0.07	0.09	0.84	2.42	4.53	27	84.30	207	3296	16942.7	37692	10	10	10	
0.5	0.5	0.06	0.07	0.08	0.95	2.10	4.85	27	52.50	102	5358	14165.4	35060	10	10	10	
	1	0.06	0.07	0.07	1.03	3.47	15.20	27	50.70	102	5233	26082.9	129172	10	10	10	
	2	0.06	0.07	0.08	0.74	1.43	2.54	27	45.90	84	2634	8274.6	17621	10	10	10	
0.8	0.5	0.06	0.07	0.08	0.48	1.33	5	39	54.90	81	1027	7685.82	33714	10	10	10	
	1	0.06	0.07	0.07	0.54	1.16	4.16	24	44.40	69	1474	6782.3	33714	10	10	10	
	2	0.06	0.06	0.07	0.51	1.07	3	24	43.50	60	2105	6168	20749	10	10	10	

Table 5.5: Comparison of computational results for branch and bound and integer programming model for n=22 and K=4

CPU second					Number of visited nodes					# of optimal solutions per hour						
τ^1	τ^2	BB with best lb first search			IP			BB with best lb first search			IP			BB with best lb first s.		IP
		min	av	max	min	av	max	min	av	max	min	av	max			
0.2	0.5	0.64	1.31	3.12	> 3600	> 3600	> 3600	676	5762.80	18960	> 10321100	> 10321100	> 10321100	10	0	0
	1	0.64	1.24	2.53	> 3600	> 3600	> 3600	664	5287.20	15016	> 10313100	> 10313100	> 10313100	10	0	0
	2	0.65	1.21	2.31	> 3600	> 3600	> 3600	632	5017.20	13472	> 9846100	> 9846100	> 9846100	10	0	0
0.5	0.5	0.60	0.67	0.79	> 3600	> 3600	> 3600	376	911.60	1856	> 10719100	> 10719100	> 10719100	10	0	0
	1	0.60	0.71	1.30	> 3600	> 3600	> 3600	400	1212.80	5888	> 10581100	> 10581100	> 10581100	10	0	0
	2	0.60	0.70	1.07	> 3600	> 3600	> 3600	384	1136.80	4184	> 10526100	> 10526100	> 10526100	10	0	0
0.8	0.5	0.60	1.04	2.36	> 3600	> 3600	> 3600	316	3728.80	13612	> 10437421	> 10437421	> 10437421	10	0	0
	1	0.59	0.95	1.61	> 3600	> 3600	> 3600	316	3109.20	8324	> 15721908	> 15721908	> 15721908	10	0	0
	2	0.61	0.83	1.31	> 3600	> 3600	> 3600	316	2154.80	6088	> 16341280	> 16341280	> 16341280	10	0	0

Table 5.6: Comparison of computational results for branch and bound and integer programming model for n=20 and K=6

τ^1	τ^2	CPU second				Number of visited nodes						# of optimal solutions per hour	
		BB with best lb first search		IP		BB with best lb first search		IP		IP		BB with best lb first s.	
		min	av	max	min	max	min	av	max	min	max	min	max
0.2	0.5	0.11	1.88	10.15	> 3600	> 3600	462	16476	83910	> 11916100	> 11916100	10	0
	1	0.12	1.93	10.55	> 3600	> 3600	462	16629	83910	> 10411100	> 10411100	10	0
	2	0.11	1.89	10.11	> 3600	> 3600	450	16569.60	83910	> 10355100	> 10355100	10	0
0.5	0.5	0.15	0.57	1.32	> 3600	> 3600	888	4620.60	13800	> 9954100	> 9954100	10	0
	1	0.16	0.53	1.22	> 3600	> 3600	774	4580.40	13800	> 9954100	> 9954100	10	0
	2	0.14	0.53	1.21	> 3600	> 3600	774	4579.80	13800	> 9262100	> 9262100	10	0
0.8	0.5	0.13	3.61	27.62	> 3600	> 3600	498	24568.80	167268	> 13235105	> 13235105	10	0
	1	0.13	2.63	18.70	> 3600	> 3600	498	20106	129906	> 19382950	> 19382950	10	0
	2	0.13	2.54	17.66	> 3600	> 3600	498	19738.20	125832	> 17645040	> 17645040	10	0

Table 5.7: Comparison of computational results for branch and bound and integer programming model for n=32 and K=4

τ^1	τ^2	CPU second				Number of visited nodes						# of optimal solutions per hour	
		BB with best lb first search		IP		BB with best lb first search		IP		IP		BB with best lb first s.	
		min	av	max	min	max	min	av	max	min	max	min	max
0.2	0.5	0.64	566.39	4896.45	> 3600	> 3600	64	446918.4	2580284	> 11393444	> 11393444	9	0
	1	0.63	521.04	4473.44	> 3600	> 3600	64	434980.4	2455648	> 14368266	> 14368266	9	0
	2	0.63	466.16	3949.09	> 3600	> 3600	64	424450.8	2348432	> 10017914	> 10017914	9	0
0.5	0.5	0.62	413.42	4039.63	> 3600	> 3600	140	305320	2432768	> 14077669	> 14077669	9	0
	1	0.62	395.79	3440.28	> 3600	> 3600	140	338936.8	2265228	> 11676534	> 11676534	10	0
	2	0.63	418.64	3005.35	> 3600	> 3600	140	381408.8	2138208	> 14129306	> 14129306	10	0
0.8	0.5	2.62	638.94	2802.15	> 3600	> 3600	16368	528892.4	1929968	> 10274269	> 10274269	10	0
	1	1.69	443.13	3326.65	> 3600	> 3600	10496	370697.2	2218900	> 10576769	> 10576769	10	0
	2	0.93	106.31	778.88	> 3600	> 3600	3232	195605.2	991476	> 12714224	> 12714224	10	0

Table 5.8: Comparison of computational results for branch and bound and integer programming model for n=39 and K=5

		CPU second				Number of visited nodes						# of optimal solutions per hour				
τ^1	τ^2	BB with best lb first search			IP			BB with best lb first search			IP			BB with best lb first s.		IP
		min	av	max	min	av	max	min	av	max	min	av	max			
0.2	0.5	0.25	7915.72	35836.88	> 3600	> 3600	> 3600	1445	2538270.5	8136535	> 11481032	> 11481032	> 11481032	6	0	0
	1	0.26	7113.01	26176.97	> 3600	> 3600	> 3600	1445	2436231.6	7041450	> 10935972	> 10935972	> 10935972	6	0	0
	2	0.31	6522.49	25817.63	> 3600	> 3600	> 3600	1445	2364487	7040345	> 12585680	> 12585680	> 12585680	6	0	0
0.5	0.5	9.40	3987.29	19212.24	> 3600	> 3600	> 3600	45685	1713138.6	6312675	> 10669606	> 10669606	> 10669606	7	0	0
	1	9.15	4089.69	20086.65	> 3600	> 3600	> 3600	45620	1716932.5	6311655	> 12090182	> 12090182	> 12090182	7	0	0
	2	9.05	4115.87	20205.86	> 3600	> 3600	> 3600	44780	1715049.5	6311360	> 12496688	> 12496688	> 12496688	7	0	0
0.8	0.5	0.26	467.84	8399.46	> 3600	> 3600	> 3600	1520	1047960	3837960	> 12216162	> 12216162	> 12216162	9	0	0
	1	0.24	1580.95	7681.48	> 3600	> 3600	> 3600	1305	1032610	3664680	> 11411084	> 11411084	> 11411084	8	0	0
	2	0.24	2155.99	11284.41	> 3600	> 3600	> 3600	1305	1151022	4562670	> 9255867	> 9255867	> 9255867	8	0	0

5.5 A warm start integer programming model for $1|chains(l);r_i|\sum C_i$

A warm start integer programming model is proposed to solve the scheduling problem, $1|chains(l);r_i|\sum C_i$. In this model, feasible solution obtained by one of the four heuristic algorithms which are used to determine the upper bounds for the branch and bound algorithm is used as an initial solution for the integer programming model. Also, a new lower bound is computed in addition to the *lowerbound* defined in Section 5.3. The new lower bound assumes that all the processing times are equal to the smallest processing time in the original instance and considers the release time and timelag constraints. Maximum of this new lower bound and the *lowerbound* used in the branch and bound algorithm is set as the lower bound on the objective function in the integer programming model. When the release times are large and the processing times and timelags are relatively small to fill the gaps between the release times, the new lower bound provides a better dual bound. In the opposite case, *lowerbound* provides a better dual bound. The model is as follows:

Notation

- *Sets and Parameters :*

O_i : Set of alternative destinations, $j \in V$ where $(i, j) \in A$ or $(i, j) \in E$, for any $i \in V$.

I_j : Set of alternative sources, $i \in V$ where $(i, j) \in A$ or $(i, j) \in E$, for each $j \in V$.

r_i : Release time for each task $i \in V$. For $i = 0$ or $i = n + 1$, $r_i = 0$. For $i \neq 0$ and $i \neq n + 1$, $r_i \geq 0$.

p_i : Processing time of task i where $i \in V$. If $i \neq 0$ and $i \neq n + 1$, $p_i = p$. When $i = 0$ or $i = n + 1$, $p_i = 0$.

l : Duration of timelag.

Z^0 : Objective function value of the initial solution that is obtained by applying the third heuristic algorithm in Section 5.3.

LB_3^0 : Initial lower bound on the objective function value. This bound is equal to $\max[lowerbound, lowerbound^2]$ where the *lowerbound* is obtained as explained in Section 5.3 and $lowerbound^2$ is obtained by the following method:

(i) First the release times for the original problem are updated as follows:

for $k = 0 \dots K - 1$

for $i = 1 \dots n_k - 1$

if $r_{k,i} < r_{k,i-1} + p_{k,i-1} + l$

$r_{k,i} = r_{k,i-1} + p_{k,i-1} + l$

(ii) For $0 \leq k < K$ and $0 \leq i < n_k$, the minimum processing time, $p_{k,i}$ is obtained.

Let this minimum processing time value be shown by p^m .

(iii) The following procedure is applied:

completion time = 0;

*lowerbound*² = 0;

while $n > 0$

{

obtain the $\min_{0 \leq k < K} r_{k,0}$ and flag the corresponding chain as $k^c = k$;

if $r_{k^c,0} < \text{completion time}$

$r_{k^c,0} = \text{completion time}$;

completion time = $r_{k^c,0} + p^m$;

*lowerbound*² = *lowerbound*² + *completion time* ;

for $i = 1 \dots n_{k^c} - 1$

$r_{k^c,i} = r_{k^c,i+1}$;

$n_{k^c} = n_{k^c} - 1$;

$n = n - 1$;

}

- *Decision variables* :

x_{ij} : Binary variable for the link between $i \in V$ and $j \in V$. $x_{ij} = 1$ if task j is assigned immediately after task i . $x_{ij} = 0$ in the other case.

S_i : Start time at node $i \in V$.

C_i : Completion time at node $i \in V$.

- *Objective function* :

$\sum C_{i \leq n}$: Sum of completion times for $0 \leq i \leq n$. Note that $i = 0$ and $i = n + 1$

are the dummy nodes where $C_0 = 0$ and $C_{n+1} > 0$. Thus C_{n+1} is ignored while computing the objective function value.

Model

$$\text{minimize } \sum C_{i \leq n} \quad (1)$$

$$S_i \geq r_i \quad \text{for each } i \in V \quad (2)$$

$$C_i = S_i + p_i \quad \text{for each } i \in V \quad (3)$$

$$S_j > S_i + p_i + \sum_{i \prec k \prec j} p_k \quad \text{for each } i, j \in V \text{ with } i \prec j \quad (4)$$

$$S_j \geq C_i + l \quad \text{for each } (i, j) \in A \text{ where } i \neq 0 \text{ or } j \neq n+1 \quad (5)$$

$$S_j + M(1 - x_{ij}) \geq C_i \quad \text{for each } (i, j) \in E \text{ and } (i, j) \in A \quad (6)$$

$$\sum_{j \in O_i} x_{ij} = 1 \quad \text{for each } i \in V \text{ where } i \neq n+1 \quad (7)$$

$$\sum_{i \in I_j} x_{ij} = 1 \quad \text{for each } j \in V \text{ where } j \neq 0 \quad (8)$$

$$S_{n+1} \geq C_i \quad \text{for each } i \in V \text{ where } i \neq n+1 \quad (9)$$

$$\sum x_{ij} + x_{ji} \leq 1 \quad \text{for each } (i, j) \in E \quad (10)$$

$$C_i \leq \sum p_i + \tau_1 \sum p_i + nl \quad \text{for each } i \in V \quad (11)$$

$$S_i \leq \sum p_i + \tau_1 \sum p_i + nl \quad \text{for each } i \in V \quad (12)$$

$$\sum C_i \geq LB_3^0 \quad \text{for each } i \in V \quad (13)$$

The constraint sets (1),..., (10) are the same as in the integer programming model given in Section 5.3.

Due to constraint set (11), an integer upper bound is assigned for the decision variable, C_i , for each $i \in V$.

Due to constraint set (12), an integer upper bound is assigned for the decision variable, S_i , for each $i \in V$.

Due to constraint (13), objective function value cannot be smaller than the lower bound, LB_3^0 .

Technical notes for warm start integer programming model

mipstart=1 in the option file.

F^0 : Set of the links, (i, j) , which belong to the path generated by applying the third heuristic algorithm to find the upper bounds in Section 5.3. This path represents the

order of scheduled tasks.

ST_i^0 : Start time value of task i which is obtained by applying the third heuristic algorithm in Section 5.3.

CT_i^0 : Completion time value of task i which is obtained by applying the third heuristic algorithm in Section 5.3.

Z^0 : Objective function value of the initial solution that is obtained by applying the third heuristic algorithm in Section 5.3.

x_{ij}^0 : $x_{ij}^0 = 1$ if task j is assigned immediately after task i in the initial solution.

S_i^0 : Start time at node $i \in V$ in the initial solution.

C_i^0 : Completion time at node $i \in V$ in the initial solution.

$$x_{ij}^0 = 1 \quad \text{for each } (i, j) \in F^0 \quad (14)$$

$$S_i^0 = ST_i^0 \quad \text{for each } i \in V \quad (15)$$

$$C_i^0 = CT_i^0 \quad \text{for each } i \in V \quad (16)$$

$$\sum C_{i \leq n}^0 = Z^0 \quad (17)$$

The constraint set (14) shows the binary variables which are equal to 1 in the initial solution.

The constraint set (15) shows the start time values of tasks with respect to the initial solution.

The constraint set (16) shows the completion time values of tasks with respect to the initial solution.

The constraint (17) shows the objective function value obtained in the initial solution.

5.6 Computational results for warm start integer programming model

for $1|chains(l); r_i| \sum C_i$

All algorithms determining the upper and lower bounds and the initial feasible solution were implemented in C and the output files were called by GAMS to solve the integer programming model. The warm start integer programming model was tested for $n \in \{100, 500, 1000\}$. The parameters of the original scheduling problem are

denoted by K , n_k , $r_{k,i}$, $p_{k,i}$ where $0 \leq k < K$ and $0 \leq i < n_k$. Formulations and coefficients which are used to generate data are given as follows:

- (i) $K = \lfloor \kappa_1 n \rfloor$ where $\kappa_1 = 0.1$,
- (ii) $\max_{0 \leq k < K} n_k = \lceil n/K + \kappa_2 [n - K - n/K + 1] \rceil$ where $\kappa_2 \in \{0.2\}$,
- (iii) $p_{k,i} \in U[1, 20]$,
- (iv) $r_{k,i} \in U[0, \tau_1 \sum p_{k,i}]$ where $\tau_1 \in \{0.5, 5\}$,
- (v) $l = \tau_2 \sum p_{k,i}/n$ where $\tau_2 \in \{0.5, 2\}$.

Due to (i), number of chains, K , depends on the coefficient κ_1 and problem size, n . Due to (ii), number of tasks in the longest chain, $\max_{0 \leq k < K} n_k$, depends on K , n and κ_2 . When $\max_{0 \leq k < K} n_k$ is determined, number of tasks in the other chains are randomly generated. In this generation, the lengths of these chains have to be smaller than $\max_{0 \leq k < K} n_k$ and sum of the lengths of all chains cannot exceed the problem size, n . With respect to (iii) processing times are uniformly distributed between 1 and 20. Due to (iv), release times are uniformly distributed between 0 and $\tau_1 \sum p_{k,i}$ for $\tau_1 \in \{0.5, 5\}$. For, $\tau_1 = 0.5$, release times are small which means that more tasks are available at earlier times. For $\tau_2 = 5$, release times take large values which tend to generate cuts on the solution space. Due to (v), timelag is dependent on $\sum p_{k,i}/n$ and τ_2 . Data was generated in C.

For each combination of different values of parameters, 10 instances were generated. All tests were performed on a 2.5 GHz PC with Intel Core i5 processor and 4GB of RAM. For integer programming model, GAMS/CPLEX solver with Cplex Callable Library from ILOG CPLEX 12.1.0 was used.

Results are given in Table 5.9, Table 5.10 and Table 5.11. In the tables, Z^0 is the sum of completion times which is obtained by the heuristic algorithm in C. Z^0 is set as an upper bound for the solution of integer programming model in GAMS. Also the related feasible solution set including Z^0 , C_i^0 , S_i^0 and $x_{i,j}^0$ is used as the initial feasible solution in GAMS. LB_3^0 is the lower bound for the objective function which is computed in C. This lower bound is imported by GAMS to improve the dual bound of the integer programming model. The values of the column belonging to $\sum C_i$ show

the solutions obtained by GAMS in a time limitation of 300 seconds. Next columns present the gaps at root and termination in GAMS. The last column gives the solution times for the integer programming model.

Table 5.9, Table 5.10 and Table 5.11 show the computational results for $n = 100$, $n = 500$ and $n = 1000$, respectively. Regarding the results, for the instances with $\tau_1 \in \{5\}$, gaps were considerably smaller than the gaps for the instances with $\tau_1 \in \{0.5\}$. For the instances with $\tau_1 \in \{5\}$, the difference between the initial primal solution and the dual solution were small which is the reason of small gaps.

5.7 Implications

The research in Section 5.1 and Section 5.2 indicates that the warm start integer programming model in Section 5.2 outperforms both the integer programming model and the branch and bound algorithm in Section 5.1.

The branch and bound algorithm in Section 5.1 was able to solve the instances with the problem sizes $n = 11$, $n = 20$, $n = 22$ and $n = 32$ and $n = 39$ in a reasonable time with zero gap in the final solution. For the instances with $n = 39$, average solution time over ten instances for each different combination of parameters changes between 467.84 and 7915.72 seconds while the maximum solution time over ten instances for each different combination of parameters changes between 7681.48 and 35836.88 seconds. For $n > 39$, the solution time to obtain the optimal solution for any instance is larger than the worst solution time for $n = 39$. For the integer programming model in Section 5.1, the time elapsed to obtain the optimal solution could not be observed in most of the instances as it was considerably large. For only the instances with $n = 11$, optimal solutions could be obtained in less than one hour.

As the addressed problem is the generalization of two strongly NP-hard problems, an efficient dominance rule which does not have a negative effect on the solution time is hard to find. By introducing an efficient dominance rule to any branch and bound algorithm, number of visited nodes should become smaller without increasing the solution time of the algorithm since the main intention is to decrease the solution time

Table 5.9: Results for warm start integer programming model for $n = 100$

n	κ_1	κ_2	τ_1	τ_2	Z^0	LB_3^0	$\sum C_i$	<i>Initial gap</i>	<i>Final gap</i>	<i>CPU time (sec)</i>
100	0.1	0.2	0.5	0.5	47095	44080	47095	0,06	0.06	300.00
					54395	50431	54395	0,07	0.07	300.00
					51213	45098	51209	0,12	0.12	300.00
					52724	47955	52724	0,09	0.09	300.00
					47784	41783	47784	0,13	0.13	300.00
					53516	44942	53516	0,16	0.16	300.00
					51201	44474	51201	0,13	0.13	300.00
					52437	47655	52437	0,09	0.09	300.00
					45438	40370	45432	0,11	0.11	300.00
					44677	40803	44590	0,09	0.08	300.00
				2	48688	44080	48688	0,09	0.09	300.00
					54745	50431	54745	0,08	0.08	300.00
					53867	45098	53867	0,16	0.16	300.00
					55288	47955	55288	0,13	0.13	300.00
					48470	41783	48470	0,14	0.14	300.00
					57392	44942	57392	0,22	0.22	300.00
					54330	44474	54330	0,18	0.18	300.00
					54435	47655	54435	0,13	0.13	300.00
					46495	40370	46495	0,13	0.13	300.00
					46677	40803	46677	0,13	0.13	300.00
			5	0.5	310038	308908	310038	0,00	0.00	1.33
					282125	280927	282108	0,00	0.00	8.40
					222848	221935	222848	0,00	0.00	0.96
					226671	225702	226669	0,00	0.00	1.36
					231812	230791	231812	0,00	0.00	2.21
					255327	254144	255280	0,00	0.00	14.25
					245634	244559	245623	0,00	0.00	19.03
					251633	250611	251631	0,00	0.00	1.37
					243073	241979	243050	0,00	0.00	3.84
					175430	174532	175417	0,01	0.00	5.59
				2	310128	309008	310119	0,00	0.00	2.24
					282201	281027	282181	0,00	0.00	5.07
					222890	221977	222890	0,00	0.00	1.16
					226696	225727	226694	0,00	0.00	1.47
					231948	230914	231948	0,00	0.00	2.69
					255353	254233	255342	0,00	0.00	6.06
					245842	244778	245828	0,00	0.00	4.34
					251824	250761	251808	0,00	0.00	2.46
					243181	242088	243166	0,00	0.00	6.26
					175509	174617	175487	0,01	0.00	2.56

Table 5.10: Results for warm start integer programming model for $n = 500$

n	κ_1	κ_2	τ_1	τ_2	Z^0	LB_3^0	$\sum C_i$	<i>Final gap</i>	<i>Final gap</i>	<i>CPU time (sec)</i>
500	0.1	0.2	0.5	0.5	1288559	1139160	1288559	0.12	0.12	300.00
					1203407	1089887	1203407	0.10	0.09	300.00
					1307086	1180052	1307086	0.10	0.10	300.00
					1219885	1098106	1219885	0.11	0.10	300.00
					1234752	1083127	1234752	0.12	0.12	300.00
					1206130	1102620	1206130	0.09	0.09	300.00
					1211579	1086491	1211579	0.10	0.10	300.00
					1230840	1112435	1230840	0.11	0.10	300.00
					1238619	1120765	1238619	0.10	0.10	300.00
					1194804	1090472	1190034	0.10	0.08	300.00
				2	1319155	1139160	1319155	0.14	0.14	300.00
					1217433	1089887	1217433	0.10	0.10	300.00
					1340722	1180052	1340722	0.12	0.12	300.00
					1247651	1098106	1247651	0.12	0.12	300.00
					1259480	1083127	1259480	0.14	0.14	300.00
					1224497	1102620	1224497	0.10	0.10	300.00
					1246208	1086491	1246208	0.13	0.13	300.00
					1272445	1112435	1272445	0.13	0.13	300.00
					1247975	1120765	1247975	0.10	0.10	300.00
					1208740	1090472	1208740	0.10	0.10	300.00
			5	0.5	7016198	7010499	7016198	0.00	0.00	300.00
					6360870	6355506	6360868	0.00	0.00	300.00
					6527731	6522308	6527731	0.00	0.00	300.00
					6960361	6954424	6960361	0.00	0.00	300.00
					5705795	5700870	5705795	0.00	0.00	300.00
					6830732	6825113	6830732	0.00	0.00	300.00
					6378224	6372886	6378224	0.00	0.00	300.00
					6757714	6752297	6757714	0.00	0.00	300.00
					6331690	6326240	6331690	0.00	0.00	300.00
					6419329	6413869	6419329	0.00	0.00	300.00
				2	7016486	7010793	7016486	0.00	0.00	300.00
					6360981	6355625	6360981	0.00	0.00	300.00
					6527850	6522471	6527850	0.00	0.00	300.00
					6960524	6954618	6960524	0.00	0.00	300.00
					5705948	5701038	5705948	0.00	0.00	300.00
					6831046	6825471	6831046	0.00	0.00	300.00
					6378406	6373038	6378406	0.00	0.00	300.00
					6757991	6752592	6757991	0.00	0.00	300.00
					6331825	6326403	6331825	0.00	0.00	300.00
					6419455	6413995	6419455	0.00	0.00	300.00

Table 5.11: Results for warm start integer programming model for $n = 1000$

n	κ_1	κ_2	τ_1	τ_2	Z^0	LB_3^0	$\sum C_i$	<i>Initial gap</i>	<i>Final gap</i>	<i>CPU time (sec)</i>
1000	0.1	0.2	0.5	0.5	5046803	4608685	5046803	0.09	0.09	300.00
					4914422	4473898	4914422	0.09	0.09	300.00
					5253969	4748289	5253969	0.10	0.10	300.00
					4927891	4499654	4927891	0.09	0.09	300.00
					5065760	4563917	5060760	0.10	0.10	300.00
					4936614	4493329	4936614	0.10	0.10	300.00
					4868449	4397708	4868449	0.10	0.10	300.00
					4997552	4498440	4997552	0.10	0.10	300.00
					4958003	4458559	4958003	0.10	0.10	300.00
					5046704	4607023	5046704	0.09	0.09	300.00
				2	5119561	4608685	5119561	0.10	0.10	300.00
					5046881	4473898	5046881	0.11	0.11	300.00
					5322001	4748289	5322001	0.11	0.11	300.00
					5033724	4499654	5033724	0.11	0.11	300.00
					5179730	4563917	5179730	0.12	0.12	300.00
					5003762	4493329	5003762	0.10	0.10	300.00
					4910470	4397708	4910470	0.10	0.10	300.00
					5084902	4498440	5084902	0.12	0.12	300.00
					5025449	4458559	5025449	0.11	0.11	300.00
					5086411	4607023	5086411	0.09	0.09	300.00
			5	0.5	26523008	26511893	26523008	0.00	0.00	300.00
					26407217	26396328	26407217	0.00	0.00	300.00
					25567841	25556878	25567841	0.00	0.00	300.00
					27222170	27211004	27222170	0.00	0.00	300.00
					27138929	27127789	27138929	0.00	0.00	300.00
					27651105	27639542	27651105	0.00	0.00	300.00
					25210213	25199376	25210213	0.00	0.00	300.00
					26841231	26830248	26841231	0.00	0.00	300.00
					26346669	26335822	26346669	0.00	0.00	300.00
					26821021	26810295	26821021	0.00	0.00	300.00
				2	26523425	26512343	26523425	0.00	0.00	300.00
					26407593	26396705	26407593	0.00	0.00	300.00
					25568287	25557311	25567979	0.00	0.00	300.00
					27222425	27211252	27222425	0.00	0.00	300.00
					27139187	27128024	27139187	0.00	0.00	300.00
					27651319	27639795	27651319	0.00	0.00	300.00
					25210596	25199785	25210596	0.00	0.00	300.00
					26841560	26830557	26841560	0.00	0.00	300.00
					26346864	26336006	26346864	0.00	0.00	300.00
					26821437	26810712	26821437	0.00	0.00	300.00

of the algorithm by decreasing the number of visited nodes. However, the procedure applied to check whether a dominance rule is valid or not may also consume a considerable time. For the addressed problem, a dominance rule that needs to compare the nodes representing the partial schedules which contain the same number of tasks, have different order of tasks but end with the same tasks was suggested in Section 5.1. However, since the partial schedules with desired properties are searched after visiting each node, the procedure is time consuming. Although there is a decrease in the number of visited nodes, the related decrease in solution time does not cover the time spent to search the desired nodes.

The integer programming model in Section 5.2 spends a large amount of time to find an initial feasible solution. Thus a warm start integer programming model where the initial feasible solution is obtained by one of the heuristic methods used in branch and bound is proposed. By this way starting from a known feasible solution that is obtained by a heuristic saves much time in the solution of the integer programming model. Also, introducing the lower bounds as cuts on the objective function value helps to reduce the solution time. Warm start integer programming model was run at most 300 seconds for each instance. For $n = 100$, $n = 500$, $n = 1000$, half of the instances were solved with 0 gap or with a gap in the neighborhood of 0.

Chapter 6

CONCLUSION

In this thesis single machine scheduling problems with timelag constraints have been studied. Earlier studies have shown that most of the classical scheduling problems such as general shop problems, problems with multiprocessor tasks, problems with multi-purpose machines, and single machine problems with sequence-dependent changeover times can be polynomially reduced to single machine scheduling problems with timelag constraints. Moreover, scheduling problems with timelag constraints are used to model many other problems in real life. Thus, solving the single machine scheduling problems with timelag constraints to optimality is crucial for complexity theory or practical applications.

There has been a considerable amount of work with regard to complexity of single machine scheduling problems with timelag constraints in the literature. A number of problems has already been classified as NP-hard or polynomially solvable. However, a group of problems is still open. Open problems are also separated into groups as minimal open and maximal open problems. For minimal open problems, it is known that all easier problems are polynomially solvable. For maximal open problems, it is known that all harder problems are NP-hard. Providing the complexity of any minimal or maximal open problem contributes to determine the boundary between the classes of polynomially solvable and NP-hard problems for single machine scheduling problems with timelag constraints.

The studies in Chapter 3 and Chapter 4 of this thesis are motivated by the existing gap regarding the complexity classes of single machine scheduling problems with timelag constraints. An easier case of the problem considered in Chapter 3 is a minimal open problem. Thus the harder problem analyzed in Chapter 3 is also open.

To begin with the definition of the easier problem, three-field notation is given as $1|chains(1); p_i = p; r_i|C_{max}$ and the problem is to schedule a set of tasks with chain like precedence relations, arbitrary release times, identical processing times and unit timelags to minimize the makespan. As far as the harder problem is considered, three-field notation is given as $1|chains(l < p); p_i = p; r_i|C_{max}$ and in this case timelag is not necessarily unit although its size has to be smaller than the processing time. In Chapter 3, a new algorithm that minimizes the makespan for the scheduling problem, $1|chains(l < p); p_i = p; r_i|C_{max}$, has been proposed. Furthermore, optimality of the solution generated by the proposed algorithm has been proven and it has been shown that the proposed algorithm is $O(n^2)$ which means that the algorithm is polynomial. Therefore, as a result of the study in Chapter 3, it has been concluded that the problem, $1|chains(l < p); p_i = p; r_i|C_{max}$, is polynomially solvable which means also that the easier problem, $1|chains(1); p_i = p; r_i|C_{max}$, is polynomially solvable. Thus, the contribution of the study in Chapter 3 is to prove that two scheduling problems which had been classified as the problems with open complexity before are polynomially solvable. The problem analyzed in Chapter 4 is classified as maximal open. The problem is denoted by $1|outtree(l < p); p_i = p; r_i|C_{max}$ by the three-field notation. The difference of this problem from the problem studied in Chapter 3 is its precedence graph. The precedence graph is *outtree* instead of being *chains*. *Outtree* is a more complicated structure than chains and makes the problem harder. For the optimal solution of the problem an $O(n^3)$ algorithm has been suggested in Chapter 4. Even though $O(n^3)$ time complexity is worse than $O(n^2)$ time complexity, it is still a polynomial bound of time. The study in this thesis showed that the problem, $1|outtree(l < p); p_i = p; r_i|C_{max}$, which is the subcase of a maximal open problem is polynomially solvable.

Unlike Chapters 3 and 4, Chapter 5 deals with a single machine scheduling problem which is known to be NP-hard. The problem is shown by $1|chains(l); r_i|\sum C_i$ by three-field notation. The problem is to schedule a group of tasks with arbitrary processing times, arbitrary release times and identical timelags without violating the

chain like precedence relations to minimize the sum of completion times. In this problem, timelag does not have to be unit and it is also not bounded from above. In this thesis a branch and bound algorithm has been developed to solve this problem. In the literature, the studies providing branch and bound algorithms for the scheduling problems with the objective function sum of completion times are relatively less than the studies focusing on the branch and bound methods for scheduling problems with the objective function makespan. When the processing times are arbitrary in addition to precedence and timelag constraints and arbitrary release times, the objective function sum of completion times becomes harder to be approximated by heuristic methods. Thus obtaining tight upper bounds for any solution node in the branch and bound tree is not easy. Obtaining tight lower bounds by relaxation methods is also difficult. Proposed branch and bound algorithm computes the upper bound by four different heuristics and chooses the best bound at each iteration. Also, a constructive algorithm is used to find the lower bound at each node. All the algorithms generating the upper and lower bounds are polynomial. Computational results for the proposed branch and bound algorithm are significantly better than the results for the associated integer programming model. To solve the large instances, a warm start integer programming model is proposed which gives significantly better solutions when the release time generation range is wider.

BIBLIOGRAPHY

- Ambühl, C., Mastrolilli, M., Svensson, O. (2006). Approximating precedence constrained single machine scheduling by coloring. *APPROX and RANDOM*, 4110, 15 - 26.
- Baptiste, P., Sadykov, R. (2010). Time-indexed formulations for scheduling chains on a single machine: An application to airborne radars. *European Journal of Operational Research*, 203, 476 - 483.
- Brucker, P., Knust, S. (1999). Complexity results for single-machine problems with positive finish-start timelags. *Computing*, 63, 299 - 316.
- Brucker P., Hilbig T., Hurink J., (1999). A branch and bound algorithm for a single-machine scheduling problem with positive and negative timelags. *Discrete Applied Mathematics*, 94, 77 - 99.
- Brucker, P., Hurink, J., Knust, S. (2002). A polynomial algorithm for $P|p_j = 1, r_j, outtree| \sum C_j$. *Mathematical Methods of Operations Research*, 56, 407 - 412.
- Brucker, P., Knust, S., Oguz, C. (2006). Scheduling chains with identical jobs and constant delays on a single machine. *Mathematical Methods of Operations Research*, 63(1), 63 - 75.
- Bruno, J., Jones, III J.W., So, K. (1980). Deterministic scheduling with pipelined processors. *IEEE Transactions on Computers*, 29(4), 308 - 316.
- Condotta, A., Shakhlevich, N.V. (2012). Scheduling coupled-operation jobs with exact timelags. *Discrete Applied Mathematics*, 160(16-17), 2370 - 2388.
- Ecker, K., Tanas, M. (2012). Complexity of scheduling of coupled tasks with chains precedence constraints and any constant length of gap. *Journal of the Operational Research Society*, 63(4), 524 - 529.
- Finta, L., Liu, Z. (1996). Single machine scheduling subject to precedence delays.

Discrete Applied Mathematics, 70, 247 - 266.

Fondrevelle, J., Oulamara, A., Portmann, M.C., (2006). Permutation flowshop scheduling problems with maximal and minimal time lags. *Computers & Operations Research*, 33, 1540 - 1556.

Fondrevelle, J., Oulamara, A., Portmann, M.C., (2008). Permutation flowshop scheduling problems with time lags to minimize the weighted sum of machine completion times. *International Journal of Production Economics*, 112, 168 - 176.

Graham, R.E., Lawler, E.L., Lenstra, J.K., Rinnooy Kan A.H.G., (1979). Optimization and approximation in deterministic sequencing and scheduling. *Annals of Discrete Mathematics*, 4, 287 - 326.

Hu, T.C. (1961). Parallel sequencing and assembly line problems. *Operations Research*, 9(6), 841-848.

Huo, Y., Leung, J.Y.T. (2005). Minimizing total completion time for UET tasks with release time and outtree precedence constraints. *Mathematical Methods of Operations Research*, 62, 275 - 279.

Karp, R.M., (1972). Reducibility Among Combinatorial Problems, in Complexity of Computer Computations. New York : Plenum Press, 85 - 103.

KARP, R.M., (1975). On the Computational Complexity of Combinatorial Problems. *Networks*, 5, 45 - 68.

Lawler, E. L. (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19, 544 - 546.

Lawler, E.L. (1978). Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2, 75 - 90.

Lenstra, J.K., Rinnooy Kan, A.H.G. (1978). The complexity of scheduling under precedence constraints. *Operations Research*, 26, 22 - 35.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1, 343 - 362

Liu, Z. (2010). Single machine scheduling to minimize maximum lateness subject to release dates and precedence constraint. *Computers & Operations Research*, 37,

1537 - 1543.

Munier, A., Sourd, F. (2003). Scheduling chains on a single machine with nonnegative timelags. *Mathematical Methods of Operations Research*, 57(1), 111 - 123.

Pinedo, M.L. (2008). Scheduling: Theory, Algorithms and Systems. New York : Springer.

Potts, C.N., Whitehead, J.D. (2007). Heuristics for a coupled-operation scheduling problem. *Journal of the Operational Research Society*, 58, 1375 -1388.

Sheen, G.J., Liao, L.W., (2007). A branch and bound algorithm for the one-machine scheduling problem with minimum and maximum time lags. *European Journal of Operational Research*, 181, 102 -116.

Tanaev, V.S., Sotskov, Y.N., Strusevich, V.A. (1994). Scheduling theory: Multi-stage systems. Mathematics and its Applications, Dordrecht: Kluwer Academic Publishers Group.

Timkovsky, V.G. (1998): Scheduling unit-time operation jobs on identical parallel machines and in a flowshop: complexity and correlation. Technical Report. Department of Computer Science and Systems, McMaster University Hamilton.

Wang, J.B., Wang, J.J., Ji, P. (2011). Scheduling jobs with chain precedence constraints and deteriorating jobs. *Journal of the Operational Research Society*, 62, 1765 - 1770.

Wikum, E.D., Llewellyn, D.C., Nemhauser, G.L., (1994). One-machine generalized precedence constrained scheduling problems. *Operations Research Letters*, 16(2), 87 - 99.

Yu, W. (1996). The two machine flow shop problem with delays and the one machine total tardiness problem. PhD Thesis, Technische Universiteit Eindhoven.

Yu, W., Hoogeveen, H., Lenstra, J.K. (2004). Minimizing makespan in a two machine flow shop with delays and unit time operations is NP-hard. *Journal of Scheduling*, 7(5), 333 - 348.