**4. TITLE AND SUBTITLE**
One-Machine Generalized Precedence Constrained Scheduling

**6. AUTHOR(S)**
Erick D. Wi Kum, Captain

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
AFIT Student Attending: Georgia Institute of Technology

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
AFIT/CI
Wright-Patterson AFB OH 45433-6583

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for Public Release IAW 190-1
Distribution Unlimited
MICHAEL M. BRICKER, SMSgt, USAF
Chief Administration

**15. NUMBER OF PAGES**
120

**17. SECURITY CLASSIFICATION OF REPORT**
NSN 7540-0-00 5500

**19. SECURITY CLASSIFICATION OF ABSTRACT**
Unclassified

**12b. DISTRIBUTION CODE**
93-12596
One-machine Generalized Precedence Constrained Scheduling

Erick D. Wikum

Directed by Dr. Donna C. Llewellyn and Dr. George L. Nemhauser

We investigate one-machine scheduling problems subject to generalized precedence constraints. A precedence constraint specifies that the first of a pair of jobs must be completed before the second can begin. Under our generalized notion, not only must the first job be completed before the second can begin, but also, the difference between the start time of the second job and the completion time of the first job must fall in a given pair-dependent interval. The left endpoint of this interval, if greater than zero, specifies a minimum delay and the right endpoint, if finite, specifies a maximum delay between the two jobs.

To our knowledge, this dissertation contains the first explicit identification of generalized precedence constraints as we have defined them. As such, it represents the first systematic treatment of generalized precedence constrained scheduling.

Our major emphases include drawing the line between easy and hard problems with respect to precedence constraint type, precedence relation, and optimality criterion and identifying suitable algorithms and finding effective heuristics for problems that are easy and hard, respectively. We consider minimizing makespan, total completion time, or total weighted completion time subject to minimum delay precedence constraints, maximum delay precedence constraints, or a combination of the two for various precedence relations. We show that most of these problems are NP-hard for all but the simplest of precedence relations. We then present a miscellany of results including polynomially solvable special cases, heuristics, and bounds for two minimum makespan problems subject to minimum delay precedence constraints.
Bibliography


One-machine Generalized Precedence Constrained Scheduling

by

Erick D. Wikum

Captain, USAF

1992

119 pages

Doctor of Philosophy in Industrial and Systems Engineering

Georgia Institute of Technology
ONE-MACHINE GENERALIZED PRECEDENCE CONSTRAINED SCHEDULING

A THESIS
Presented to
The Academic Faculty

by

Erick D. Wikum

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Industrial and Systems Engineering

Georgia Institute of Technology
October 20, 1992
To the glory of my

Father in heaven
ACKNOWLEDGMENTS

I am indebted to my advisors, Dr. Donna Llewellyn and Dr. George Nemhauser, for the large part they had in bringing this dissertation to fruition. Dr. Llewellyn’s ability to understand immediately what had taken me a week or more to accomplish and Dr. Nemhauser’s ability to explain difficult concepts in a sentence or two never failed to amaze me. Dr. Llewellyn was always just a phone call away, with not only technical assistance, but also words of encouragement. Dr. Nemhauser helped me to develop and refine my ability to think critically, to ask the appropriate questions, and to answer those questions.

I am most appreciative for the help and support given to me by my fellow students at Georgia Tech, who could always sympathize with me since they were experiencing the same difficulties as I was. I especially want to thank Tina Barr, Arlin Johnson, Mike Coie, and Ismaei de Farias.

For the past eight years, I received a weekly phone call from my parents. My words cannot express the debt of gratitude I owe them for their love, friendship, advice, and emotional support.

When my doctoral program grew burdensome, God intervened. He sent me a helper, my wife, Elizabeth. I could never have completed my degree without the love, caring, laughter, and companionship she gave to me. Both she and her parents were especially helpful in the final stages of my program.

I am indebted most of all to my heavenly Father. He carried me when I was down, forgave me when I did wrong, provided for my every need, and loved me so much that he gave to me his only Son, my Lord and Savior, Jesus Christ!
# CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>iv</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>xi</td>
</tr>
</tbody>
</table>

## 1 INTRODUCTION

1.1 Motivation ............................................. 5
1.2 Related Literature ..................................... 6
1.3 Major Emphases ......................................... 8

## 2 1 / min and max delays, $k$ $n_1, \ldots, n_k$-chains / $C_{\text{max}}$

2.1 Min Delays Problems .................................... 10
  2.1.1 1 / min delays, $k$ 1-chains / $C_{\text{max}}$ .......... 13
  2.1.2 1 / min delays, $k$ 2,1, ... ,1-chains / $C_{\text{max}}$ ...... 15
  2.1.3 1 / min delays, $k$ 2-chains / $C_{\text{max}}$ ............. 18
  2.1.4 1 / min delays, $k$ $n_1,1,\ldots,1$-chains / $C_{\text{max}}$ .... 23

2.2 Max Delays Problems ...................................... 26
  2.2.1 1 / max delays, $k$ 1-chains / $C_{\text{max}}$ ............... 26
  2.2.2 1 / max delays, $k$ 2,1, ... ,1-chains / $C_{\text{max}}$ ....... 28
  2.2.3 1 / max delays, $k$ $n_1,1,\ldots,1$-chains / $C_{\text{max}}$ .... 31
### 2.3 Min and Max Delays Problems

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>1 / min or max delays, k 1-chains / $C_{\text{max}}$</td>
<td>35</td>
</tr>
<tr>
<td>2.3.2</td>
<td>1 / min and max delays, k 1-chains / $C_{\text{max}}$</td>
<td>36</td>
</tr>
</tbody>
</table>

### 3 Min Delays Problems

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Min Delays Problems</td>
<td>40</td>
</tr>
<tr>
<td>3.1.1</td>
<td>1 / min delays, k 1-chains / $\sum C_j$</td>
<td>40</td>
</tr>
<tr>
<td>3.1.2</td>
<td>1 / min delays, k 1-chains / $\sum w_j C_j$</td>
<td>53</td>
</tr>
<tr>
<td>3.2</td>
<td>Max Delays Problems</td>
<td>57</td>
</tr>
<tr>
<td>3.2.1</td>
<td>1 / max delays, k 1-chains / $\sum C_j$</td>
<td>57</td>
</tr>
</tbody>
</table>

### 4 Min and Max Delays, 2 $n_1, n_2$-chains / $C_{\text{max}}$

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>1 / min delays, 2 $n_1, n_2$-chains / $C_{\text{max}}$</td>
<td>68</td>
</tr>
<tr>
<td>4.2</td>
<td>1 / max delays, 2 $n_1, n_2$-chains / $C_{\text{max}}$</td>
<td>76</td>
</tr>
</tbody>
</table>

### 5 SPECIAL CASES, HEURISTICS, AND BOUNDS

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>1 / min delays, k 2, 1,..., 1-chains / $C_{\text{max}}$</td>
<td>81</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Relative Ordering of Jobs $J_2,\ldots,J_k$</td>
<td>81</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Heuristic with Worst Case Performance Ratio 2</td>
<td>82</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Pseudo-polynomial Algorithm for Special Case with $l_2 = \ldots = l_k = 0$</td>
<td>88</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Heuristic with Worst Case Performance Ratio $\frac{5}{4} + \Delta$ for NP-hard Special Case</td>
<td>92</td>
</tr>
<tr>
<td>5.2</td>
<td>1 / min delays, 2 $n_1, n_2$-chains / $C_{\text{max}}$</td>
<td>98</td>
</tr>
<tr>
<td>5.2.1</td>
<td>No Schedule Has Makespan Greater Than Twice Optimal</td>
<td>98</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Polynomially Solvable Special Cases</td>
<td>99</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Disjunctive Graph Representation</td>
<td>101</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Lower Bounds</td>
<td>105</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Bicriterion Heuristic</td>
<td>108</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>39</td>
</tr>
<tr>
<td>2.2</td>
<td>39</td>
</tr>
<tr>
<td>2.3</td>
<td>39</td>
</tr>
<tr>
<td>3.1</td>
<td>51</td>
</tr>
<tr>
<td>3.2</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>108</td>
</tr>
</tbody>
</table>

1.1 Three-field $\alpha / \beta / \gamma$ classification scheme.  
1.2 Precedence constraint terminology.  
2.1 Complexity classification of min delays, minimum makespan problems.  
2.2 Complexity classification of max delays, minimum makespan problems.  
2.3 Complexity classification of min and max delays, $k$ 1-chains, minimum makespan problems.  
3.1 Possible values for $\delta_{ij}(\sigma)$, $\delta_{ij}(\sigma)$, $\delta_{ji}(\pi)$, and $\delta_{ii}(\pi)$.  
3.2 Complexity classification of $1 / \min$ and max delays, $k$ 1-chains / $\sum C_j$ or $\sum w_jC_j$ problems.  
5.1 Release dates, processing requirements, and tails.
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The $k n_1, \ldots, n_k$-chains precedence graph.</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Example Gantt chart.</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>Instance of $1 / \text{min delays}, k$ 1-chains / $C_{\text{max}}$.</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Instance of $1 / \text{min delays}, k 2,1,\ldots, 1$-chains / $C_{\text{max}} \leq y$.</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Instance of $1 / \text{min delays}, k 2$-chains / $C_{\text{max}} \leq y$.</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Instance of $1 / \text{min delays}, k n_1,1,\ldots, 1$-chains / $C_{\text{max}} \leq y$.</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Instance of $1 / \text{max delays}, k$ 1-chains / $C_{\text{max}}$.</td>
<td>27</td>
</tr>
<tr>
<td>2.7</td>
<td>Instance of $1 / \text{max delays}, k 2,1,\ldots, 1$-chains / $C_{\text{max}} \leq p(J)$.</td>
<td>29</td>
</tr>
<tr>
<td>2.8</td>
<td>Feasible schedule for constructed instance of $1 / \text{max delays}, k 2,1,\ldots, 1$-chains / $C_{\text{max}} \leq p(J)$.</td>
<td>30</td>
</tr>
<tr>
<td>2.9</td>
<td>Instance of $1 / \text{max delays}, k n_1,1,\ldots, 1$-chains / $C_{\text{max}} \leq p(J)$.</td>
<td>32</td>
</tr>
<tr>
<td>2.10</td>
<td>Feasible schedule for constructed instance of $1 / \text{max delays}, k n_1,1,\ldots, 1$-chains / $C_{\text{max}} \leq p(J)$.</td>
<td>34</td>
</tr>
<tr>
<td>2.11</td>
<td>Optimal schedule for the $1 / \text{min or max delays}, k$ 1-chains / $C_{\text{max}}$ problem.</td>
<td>36</td>
</tr>
<tr>
<td>3.1</td>
<td>Obtaining schedule $\sigma^*$ from schedule $\sigma$.</td>
<td>43</td>
</tr>
<tr>
<td>3.2</td>
<td>Instance of $1 / \text{min delays}, k$ 1-chains / $\sum C_j$.</td>
<td>48</td>
</tr>
<tr>
<td>3.3</td>
<td>Feasible schedule for constructed instance of $1 / \text{min delays}, k$ 1-chains / $\sum w_j C_j \leq Y$.</td>
<td>55</td>
</tr>
<tr>
<td>3.4</td>
<td>Schedule with sum of processing times of jobs from ${J_1, \ldots, J_{k-1}}$ completed after job $J_k$ less than $\frac{1}{2} \sum_{j \in A} s(j)$.</td>
<td>56</td>
</tr>
</tbody>
</table>
3.5 Schedule with sum of processing times of jobs from \( \{J_1, \ldots, J_{k-1}\} \) completed after job \( J_k \) greater than \( \frac{1}{2} \sum_{j \in A} s(j) \).

3.6 Feasible schedule for constructed instance of \( 1 / \text{max delays, } k \)-chains / \( \sum C_j \leq Y \).

4.1 Strict ordering requirements imposed by \( 2 n_1, n_2 \)-chains.

4.2 Bins into which the 2-jobs are placed.

4.3 Makespan contribution \( f_t, t \in \{1, \ldots, n_1\} \) and \( x_t = 0 \).

4.4 Makespan contribution \( f_t, t \in \{1, \ldots, n_1\}, x_t \in \{1, \ldots, n_2 - s_{t-1}\} \).

4.5 Makespan contribution \( f_{n_t+1}, s_{n_1} = n_2 \).

4.6 Makespan contribution \( f_{n_t+1}, s_{n_1} \in \{0, 1, \ldots, n_2 - 1\} \).

4.7 Schedule defined by \( x_1, \ldots, x_{t-1}, \text{ and } x_t \).

4.8 Schedule defined by \( x_1, \ldots, x_{n_1}, \text{ and } x_{n_1+1} \).

5.1 Instance of \( 1 / \text{min delays, } k \)-chains / \( C_{\text{max}} \).

5.2 First relaxation of \( 1 / \text{min delays, } k \)-chains / \( C_{\text{max}} \) instance.

5.3 Second relaxation of \( 1 / \text{min delays, } k \)-chains / \( C_{\text{max}} \) instance.

5.4 Weighted, directed graph corresponding to schedule \( \sigma^1 \).

5.5 Weighted, directed graph corresponding to schedule \( \sigma^2 \).

5.6 Obtaining schedule \( \bar{\sigma} \) from schedule \( \sigma^* \).

5.7 Four possible forms for any schedule with job \( J_{x_1} \) scheduled first.

5.8 Special case of \( 1 / \text{min delays, } k \)-chains / \( C_{\text{max}} \).

5.9 Bins into which the 2-jobs are placed.

5.10 Instance of \( 1 / \text{min delays, } 2 n_1, n_2 \)-chains / \( C_{\text{max}} \).

5.11 Graph \( H \) associated with instance of \( 1 / \text{min delays, } 2 n_1, n_2 \)-chains / \( C_{\text{max}} \).

5.12 Schedule generated using the preemptive version of the Schrage algorithm.

5.13 Bicriterion shortest path network.
SUMMARY

We investigate one-machine scheduling problems subject to generalized precedence constraints. A precedence constraint specifies that the first of a pair of jobs must be completed before the second can begin. Under our generalized notion, not only must the first job be completed before the second can begin, but also, the difference between the start time of the second job and the completion time of the first job must fall in a given pair-dependent interval. The left endpoint of this interval, if greater than zero, specifies a minimum delay and the right endpoint, if finite, specifies a maximum delay between the two jobs.

Generalized precedence constraints can arise in scheduling athletic competitions. An obvious requirement for scheduling of this kind is the inclusion of minimum delays between certain pairs of events (jobs) to allow athletes time to rest.

Literature directly related to generalized precedence constrained scheduling is seemingly scant, dealing mostly with special cases and related constraints. To our knowledge, this dissertation contains the first explicit identification of generalized precedence constraints as we have defined them and represents the first systematic treatment of generalized precedence constrained scheduling.

Our major emphases include drawing the line between easy and hard problems with respect to precedence constraint type, precedence relation, and optimality criterion and identifying suitable algorithms and finding effective heuristics for problems that are easy and hard, respectively. We consider minimizing makespan, total completion time, or total weighted completion time subject to minimum delay precedence constraints, maximum delay precedence constraints, or a combination of the
two for various precedence relations. We show that most of these problems are NP-hard for all but the simplest of precedence relations. We then present a miscellany of results including polynomially solvable special cases, heuristics, and bounds for two minimum makespan problems subject to minimum delay precedence constraints which are not known to be solvable in polynomial time.
In this dissertation, we investigate one-machine scheduling problems subject to 
generalized precedence constraints. Ordinarily, requiring that job \( J_j \) precedes job \( J_{j'} \) (denoted by \( J_j \rightarrow J_{j'} \)) means job \( J_j \) must be completed before job \( J_{j'} \) can begin. Under our generalized notion of precedence constraints, not only must job \( J_j \) be 
completed before job \( J_{j'} \) can begin, but also, the time between the end of job \( J_j \) and the beginning of job \( J_{j'} \) must be at least \( l_{jj'} \), but no more than \( u_{jj'} \), where \( 0 \leq l_{jj'} \leq u_{jj'} \) and \( l_{jj'} \) is finite. We adopt the convention \( u_{jj'} = \infty \) in the absence of an upper bound and we assume \( l_{jj'} \) is an integer and \( u_{jj'} \) is an integer whenever \( u_{jj'} < \infty \).

In general, we assume that a finite set \( J \) of jobs is to be scheduled on a single 
machine that can process no more than one job at a time and is continually available 
from time zero onwards. Each job \( J_j \in J \) has processing requirement \( p_j \), assumed 
without loss of generality to be a nonnegative integer. Preemption is not allowed, 
that is, each job \( J_j \) must remain on the machine without interruption for \( p_j \) time 
units. In order to be feasible, a schedule, \( \sigma \), which specifies a non-negative integer 
start time \( \sigma(j) \) (and completion time \( C_j(\sigma) = \sigma(j) + p_j \)) for each job \( J_j \in J \), must 
satisfy the generalized precedence constraints, given by

\[
l_{jj'} \leq \sigma(j') - C_j(\sigma) \leq u_{jj'} \quad \forall <J_j, J_{j'}> \in P,
\]

where \( P \subseteq J \times J \) is the set of all precedence constrained job pairs. Notice that if 
\( l_{jj'} = 0 \) and \( u_{jj'} = \infty \), then the generalized precedence constraint corresponding to
< \(J_j, J_{j'}\)\( \in P\) is an ordinary precedence constraint. Problems for which \(l_{j,j'} > 0\) (\(u_{j,j'} < \infty\)) for some \(< J_j, J_{j'} > \in P\) will be said to be subject to minimum (maximum) delay precedence constraints.

Minimum delay precedence constraints bear a close resemblance to the constraints of the oft studied sequential ordering problem (see [9] for example). The sequential ordering problem is to schedule jobs \(J_1, \ldots, J_n\) on a single machine to minimize makespan, where setup time \(c_{j,j'} \in \mathbb{Z}_0^+\) (\(Z_0^+ = \{0\} \cup \mathbb{Z}^+\) and \(Z^+ = \{1, 2, \ldots\}\)) must elapse between the end of job \(J_j\) and the beginning of job \(J_{j'}\) if and only if job \(J_j\) immediately precedes job \(J_{j'}\). The homologous minimum delay precedence constraint,

\[
\sigma(j') - C_j(\sigma) \geq l_{j,j'}, \quad < J_j, J_{j'} > \in P,
\]

must be satisfied whether or not job \(J_j\) immediately precedes job \(J_{j'}\).

Our objective is to find a feasible schedule \(\sigma\) which, among all feasible schedules, minimizes a specified function of the job completion times. The particular functions we are interested in include \(C_{\text{max}}(\sigma) = \max_{J \in J} C_j(\sigma), \sum_{J \in J} C_j(\sigma), \) and \(\sum_{J \in J} w_j C_j(\sigma), \) where \(w_j \in \mathbb{Z}_0^+\) for all \(J \in J\). These objective functions are commonly referred to as \textit{makespan}, \textit{total completion time}, and \textit{total weighted completion time}, respectively. A feasible schedule \(\sigma\) which, among all feasible schedules, minimizes the given objective function, is said to be \textit{optimal}.

We restrict ourselves to job sets of the form

\[
J = \bigcup_{i=1}^{k} \{J_{i,1}, \ldots, J_{i,n_i}\} \cup \{\ast\},
\]

where \(k \geq 2, n_i \in \mathbb{Z}^+\) for \(i = 1, \ldots, k\), and, without loss of generality, \(n_1 \geq \cdots \geq n_k\). We further restrict ourselves to precedence relations on \(J\) of the form

\[
P = \{< J_{i,j}, J_{i,j+1} >, \ i = 1, \ldots, k, \ j = 1, \ldots, n_i - 1\} \cup \{< J_{i,n_i}, \ast >, \ i = 1, \ldots, k\}.
\]

For ease of notation, we refer to such a precedence relation \(P\) on \(J\) as \(k\) \(n_1\)-\textit{chains}, or simply as \(k\) \(n_1\)-\textit{chains} if \(n_1 = \cdots = n_k\) (see Figure 1.1). Our usage of
“chains” differs from the customary definition of that term. We will frequently use diagrams like the one in Figure 1.1 and will often find it useful to label job nodes with processing requirements and precedence arcs with lower and upper bounds on delays. If the precedence relation is \( k n_1, \ldots, n_k \)-chains, then we will, without loss of clarity, refer to \( l_{i,j, i,j+1} (u_{i,j, i,j+1}) \) simply as \( l_{i,j} (u_{i,j}) \) for \( i = 1, \ldots, k \) and \( j = 1, \ldots, n_{i-1} \) and to \( l_{i,n_i} (u_{i,n_i}) \) simply as \( l_{i,n_i} (u_{i,n_i}) \) for \( i = 1, \ldots, k \).

The reader can easily verify that the \( k n_1, \ldots, n_k \)-chains precedence graph is \textbf{"node" transitive series parallel} (see Lawler [16]). Thus, any problem which is hard for \( k n_1, \ldots, n_k \)-chains is hard for the more general class of transitive series parallel precedence graphs.

We use the three-field \( \alpha / \beta / \gamma \) classification scheme of Lawler, Lenstra, Rin-
Table 1.1: Three-field $\alpha / \beta / \gamma$ classification scheme.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$ (Machine Environment)</td>
<td>$\alpha = 1$ for single machine</td>
</tr>
<tr>
<td>$\beta$ (Job Characteristics)</td>
<td>type of precedence constraints precedence relation</td>
</tr>
<tr>
<td>$\gamma$ (Optimality Criterion to Minimize)</td>
<td>$C_{max}$</td>
</tr>
<tr>
<td></td>
<td>$\sum C_j$</td>
</tr>
<tr>
<td></td>
<td>$\sum w_j C_j$</td>
</tr>
</tbody>
</table>

Table 1.2: Precedence constraint terminology.

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>min delays</td>
<td>minimum delays only</td>
</tr>
<tr>
<td></td>
<td>$(u_{jj'} = \infty \ \forall \ &lt; J_j, J_{j'} &gt; \in P)$</td>
</tr>
<tr>
<td>max delays</td>
<td>maximum delays only</td>
</tr>
<tr>
<td></td>
<td>$(l_{jj'} = 0 \ \forall \ &lt; J_j, J_{j'} &gt; \in P)$</td>
</tr>
<tr>
<td>min or max delays</td>
<td>either minimum or maximum delays but not both</td>
</tr>
<tr>
<td></td>
<td>$(l_{jj'} = 0$ or $u_{jj'} = \infty \ \forall \ &lt; J_j, J_{j'} &gt; \in P)$</td>
</tr>
<tr>
<td>min and max delays</td>
<td>both minimum and maximum delays allowed</td>
</tr>
<tr>
<td></td>
<td>(possible to have $l_{jj'} &gt; 0$ and $u_{jj'} &lt; \infty$)</td>
</tr>
</tbody>
</table>

nooy Kan, and Shmoys [18] in conjunction with special terminology to describe constraint types and precedence relations. This classification scheme is described briefly in Table 1.1. Our precedence constraint terminology is described in Table 1.2. Our precedence relation terminology was defined in the previous paragraph. As an example, $1 / \min$ delays, $k$ 1-chains / $\sum C_j$ is the problem of minimizing the total completion time of jobs $J_1, \ldots, J_k$, and $*$, where job $J_j$ must precede job $*$ by at least $l_{j,*}$ time units for $j = 1, \ldots, k$.

The primary topics of this dissertation are scheduling and computational complexity theory. Additional information on scheduling is contained in Lawler, Lenstra,

This chapter is organized as follows. In Section 1.1, we provide motivation for generalized precedence constrained scheduling (hereafter referred to as GPCS). Section 1.2 contains a review of existing literature related to GPCS. Finally, in Section 1.3, we describe the major emphases of our research and outline the remainder of this dissertation.

1.1 Motivation

The problem that gave rise to the present research is the scheduling of the Olympic Games. An obvious requirement for scheduling of this kind is the inclusion of minimum delays between certain pairs of events (jobs) to allow athletes time to rest. Andreu and Corominas [2] presented a binary integer program for scheduling the 1992 Barcelona Olympic Games in which they specified, for each precedence constrained pair of jobs, one minimum delay between the beginning of the first job and the beginning of the second job, and another minimum delay between the end of the first job and the end of the second job. Equivalently, they might have specified a single minimum delay between the end of the first job and the beginning of the second job for each precedence constrained pair.

In modeling the Olympic scheduling problem, Andreu and Corominas introduced 0-1 variables \( x_{jt} \), where

\[
x_{jt} = \begin{cases} 
1, & \text{if event } j \text{ begins at time } t \\
0, & \text{otherwise}.
\end{cases}
\]

Assuming that precedence constraints arise solely from the fact that a facility can accommodate only one event at a time, then requiring \( i \rightarrow j \) implies event \( j \) can start no earlier than time \( p_i + l_{ij} \). Hence, \( x_{j0} = x_{j1} = \ldots = x_{j,p_i+l_{ij}-1} = 0 \). Now,
if events 1, . . . , n must precede event j, then determining the earliest possible start time for event j is a one-machine minimum makespan problem subject to minimum delay precedence constraints. Solving such a problem for each event j allows us to fix some of the variables $x_{jt}$ at zero. Operations such as these which allow us to fix variables prior to solving a problem are known as preprocessing. The symmetric problem of determining the latest start time for each event j also allows us to fix some of the $x_{jt}$'s at zero.

Suppose we solve the Olympic scheduling problem by branch-and-bound as follows. For each problem in which the event order is not completely determined, we select a pair of events j and j' such that event j is allowed to precede event j' and vice-versa, and we consider two subproblems, one with event j preceding event j' and the other with event j' preceding event j. Fixing the order of events j and j' is the same as introducing an additional ordinary precedence constraint. Since each subproblem has exactly one more ordinary or generalized precedence constraint than its immediate predecessor, then it is possible to fix at least as many and likely more $x_{jt}$'s at zero for a subproblem than for its immediate predecessor. Thus, it may be worthwhile to preprocess by solving a sequence of GPCS problems at each node of the branch and bound tree.

To summarize, generalized precedence constraints (minimum delay precedence constraints in particular) can arise in the scheduling of athletic competitions. Moreover, GPCS problems arise naturally when solving such athletic scheduling problems modeled using 0-1 variables $x_{jt}$. Having provided motivation for GPCS, we now review the literature related to GPCS.

1.2 Related Literature

Literature directly related to GPCS is seemingly scant. The subjects of papers that do pertain to GPCS fall into two broad categories, namely, special cases and related constraints. We now review papers belonging to each of these categories.
The problem of minimizing makespan on a single machine where each job \( J_j \in J \) has release time \( r_j \) (i.e., the processing of job \( J_j \) cannot commence until time \( r_j \)), processing requirement \( p_j \), and tail \( q_j \) (i.e., job \( J_j \) must spend time \( q_j \) in the system after it has been processed) is a special case of \( 1 / \min \) delays, \( k \) 2-chains / \( C_{\max} \). This release time and tail problem has been widely studied by Carlier [5] and others in the context of job shop scheduling, where its solution provides lower bounds.

Another special case of GPCS is described by Shapiro [21]. Shapiro classifies the problem of scheduling pairs of jobs separated by known, fixed time intervals on a single machine to minimize makespan as a two-machine job shop problem in which each job consists of three operations. The first and third operations, corresponding to the pair of jobs, are processed on Machine 1, while the second operation, corresponding to the separation interval, is processed on Machine 2. Although Machine 1 can process at most one operation at a time, Machine 2 has unlimited capacity. No wait in processing is permitted, that is, once a job is begun, its operations \( O_1 \), \( O_2 \), and \( O_3 \) must be processed on the machines without delay between them. Shapiro’s problem is in fact a special case of \( 1 / \min \) and \( \max \) delays, \( k \) 2-chains / \( C_{\max} \).

As evidenced by Carlier [5] and Shapiro [21], special cases of GPCS problems are not new to the literature. Unfortunately, the treatment of such cases is rather limited in scope. To our knowledge, no comprehensive or systematic study of GPCS problems exists.

Let us now consider papers which detail related constraints. Generalized precedence constraints appear elsewhere in the literature. Chretienne [6] considered a problem related to parallel computer architectures wherein the number of processors is assumed to be infinite and minimum communication delays must occur between precedence constrained job pairs only if the two are processed by different processors. Chretienne’s “Generalized Precedence Constraints,” which apply to the multiple machine environment, are similar to but clearly not the same as our minimum delay precedence constraints.
Several authors describe models which include what might be called generalized disjunctive constraints. Disjunctive constraints specify for pairs of jobs $J_j$ and $J_j'$, that either job $J_j$ must precede job $J_j'$, or vice-versa. Analogous to generalized precedence constraints, generalized disjunctive constraints specify for pairs of jobs $J_j$ and $J_j'$, not only that either job $J_j$ must precede job $J_j'$, or vice-versa, but also that the time between the end of the first job to be completed and the beginning of the other job must be at least $l_{jj'} = l_{j'j}$, but no more than $u_{jj'} = u_{j'j}$, where $0 \leq l_{jj'} \leq u_{jj'}$ and $l_{jj'}$ is finite. Even more generally, we can assume the delays depend on the actual job ordering, that is, $l_{jj'} \neq l_{j'j}$ and $u_{jj'} \neq u_{j'j}$.

As mentioned earlier, fixing the order for a disjunctively constrained pair of jobs is the same as introducing an additional ordinary precedence constraint. By the same token, fixing the order for a generalized disjunctively constrained pair of jobs is the same as introducing an additional generalized precedence constraint. Thus, generalized disjunctive constraints are in essence a generalization of generalized precedence constraints.

Andreu and Corominas [21] specified not only minimum delay precedence constraints, but also minimum delay disjunctive constraints. In scheduling dressage competitions, Shirland [22] specified an optimal interval of 40 minutes to four hours between rides for the same competitor. Since Shirland imposes no precedence constraints on pairs of rides for the same competitor, then these requirements induce minimum and maximum delay disjunctive constraints.

1.3 Major Emphases

Our research includes two major emphases. The first involves drawing the line between easy and hard GPCS problems with respect to precedence constraint type, precedence relation, and optimality criterion. The second involves identifying suitable algorithms and finding effective heuristics for GPCS problems that are easy and hard, respectively.
The remainder of this dissertation is organized as follows. Chapters 2, 3, and 4 are devoted to the classification of GPCS problems with respect to computational complexity. The GPCS problems we address in Chapter 2 are minimum makespan problems for which the number of chains is a parameter, $k$. In Chapter 3, we discuss total completion time and total weighted completion time problems with precedence relation $k$ 1-chains. The problems we address in Chapter 4 are minimum makespan problems for which the number of chains is two. In Chapter 5, we present a miscellany of results including heuristics, polynomially solvable special cases, and bounds for two problems which are not known to be solvable in polynomial time, one from Chapter 2 and the other from Chapter 4. Finally, we summarize our research and make suggestions for further research in Chapter 6.
CHAPTER 2

1 / min and max delays, \( k \ n_1, \ldots, n_k \)-chains / \( C_{max} \)

In this chapter, we draw the line between easy and hard minimum makespan problems for which the number of chains is a parameter, \( k \), by considering increasingly complex \( 'k \) chains' precedence relations. In Section 2.1, minimum delay precedence constraints only are allowed, while in Section 2.2, maximum delay precedence constraints only are allowed. Minimum and maximum delay precedence constraints are permitted in Section 2.3.

2.1 Min Delays Problems

In general, solutions to 1 / min delays, \( k \ n_1, \ldots, n_k \)-chains / \( C_{max} \) problems include machine idle time. Consequently, sequences do not necessarily uniquely correspond to schedules. A sequence of the jobs in \( J \) that satisfies the underlying ordinary precedence constraints imposed by \( P \) is said to be feasible. We now show that corresponding to each feasible sequence is a unique schedule in which each job, and job \( * \) in particular, is scheduled as early as possible so as to respect the sequence and to satisfy the machine capacity and the minimum delay precedence constraints. The following discussion is adapted from Carlier [5].

Let \( n = |J| - 1 = \sum_{i=1}^{k} n_i \). The sequence \( J_{*1} \rightarrow \cdots \rightarrow J_{*n+1} \) of the jobs in \( J \) is feasible if and only if
1. \( J_{e_{n+1}} = * \) and

2. \( e_r = (i, j) \) and \( e_s = (i, j + 1) \) implies \( 1 \leq r < s \leq n \), where \( i \in \{1, \ldots, k\} \) and \( j \in \{1, \ldots, n_i - 1\} \).

We associate with each feasible sequence \( J_{e_1} \rightarrow \cdots \rightarrow J_{e_n} \rightarrow * \) a weighted, directed graph \( G = (X, A) \). The node set \( X = \{e_1, \ldots, e_n\} \cup \{*\} \) and the arc set \( A = A_1 \cup A_2 \cup A_3 \), where

\[
A_1 = \{(e_j, e_{j+1}) : j = 1, \ldots, n-1\},
\]

\[
A_2 = \{(e_n, *)\},
\]

and

\[
A_3 = \{(e_j, e_{j'}) : <J_{e_j}, J_{e_{j'}}> \in P\}.
\]

Each arc \((e_j, e_{j'}) \in A_3\) is assigned a weight of \( p_{e_j} + l_{e_{j'}} \), arc \((e_n, *)\) is assigned a weight of \( p_{e_n} \), and each arc \((e_j, e_{j'} + 1) \in A_1 \setminus A_3\) is assigned a weight of \( p_{e_j} \). The arcs in \( A_1 \) and \( A_2 \) represent the machine capacity constraints while the arcs in \( A_3 \) represent the minimum delay precedence constraints.

Consider the schedule, \( \sigma \), where \( \sigma(e_1) = 0 \), \( \sigma(e_j) \) is equal to the weight of the maximum weight path in \( G \) from node \( e_1 \) to node \( e_j \) for \( j = 2, \ldots, n \), and \( \sigma(*) \) is equal to the weight of the maximum weight path in \( G \) from node \( e_1 \) to node \( * \). By weight of a path, we mean the sum of the arc weights over all arcs in the path. Due to the special linear structure of \( G \), schedule \( \sigma \) can be computed in time \( O(|A|) = O(n) \). By construction, the weight of any path from node \( e_1 \) to node \( e_j \) (node \( * \)), and the weight of the maximum weight path from node \( e_1 \) to node \( e_j \) (node \( * \)) in particular, provides a lower bound for the start time of job \( J_{e_j} \) (job \( * \)) in any schedule that respects the sequence \( J_{e_1} \rightarrow \cdots \rightarrow J_{e_n} \rightarrow * \) and satisfies both the machine capacity and the minimum delay precedence constraints. Hence, in schedule \( \sigma \), each job, and job \( * \) in particular, is scheduled as early as possible so as to respect the sequence \( J_{e_1} \rightarrow \cdots \rightarrow J_{e_n} \rightarrow * \) and to satisfy the machine capacity and the minimum delay precedence constraints.
We refer to a schedule computed in this manner as the active schedule associated with the given sequence. Clearly, an optimal schedule is an active schedule. Thus, 1 / min delays, $k n_1, \ldots, n_k$-chains / $C_{\max}$ is the problem of finding, among all feasible sequences, a sequence that has associated active schedule with minimum makespan. Hereafter, we drop the modifier 'active' unless required for clarity.

A convenient means of visually portraying a schedule is provided by the Gantt chart (see Figure 2.1). The Gantt chart includes a rectangular box for each job. The width of the box for a given job is proportional to that job's processing requirement. Machine idle time is represented by a dashed box with width proportional to the amount of idle time. The horizontal line at the bottom of the chart is a time axis, with time zero on the left-hand side, from which job start and completion times can be read.

In a manner synonymous with the computation of the active schedule associated with the sequence $J_{e_1} \rightarrow \cdots \rightarrow J_{e_n} \rightarrow \ast$ from the weighted, directed graph, we can construct the Gantt chart for this active schedule. First, we draw the box for job $J_{e_1}$. Next, for $j = 2, \ldots, n$, we draw the box for job $J_{e_j}$, inserting between this box and the box for job $J_{e_{j-1}}$ the smallest amount of idle time necessary to satisfy the minimum delay precedence constraint corresponding to $< \ast, J_{e_j} > \in P$. Finally, we draw the box for job $\ast$, inserting between this box and the box for job $J_{e_n}$ the smallest amount of idle time necessary to satisfy the minimum delay precedence constraints corresponding to $< \ast, \ast > \in P$. 

![Gantt chart](image-url)
We first consider the min delays problem with the simplest $k$ chains precedence relation, that is, $k$ 1-chains. For ease of notation, assume $J = \{J_1, \ldots, J_k\} \cup \{\ast\}$ (see Figure 2.2). Clearly, any feasible sequence has the form $J_{e_1} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow \ast$. The schedule, $\sigma$, associated with the sequence $J_{e_1} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow \ast$ has

$$\sigma(e_1) = 0,$$

$$\sigma(e_j) = \sum_{i=1}^{j-1} p_{e_i} \text{ for } j = 2, \ldots, k, \text{ and}$$

$$\sigma(\ast) = \max_{j=1, \ldots, k} \{\sigma(e_j) + p_{e_j} + l_{e_j}\} = \max_{j=1, \ldots, k} \{C_{e_j}(\sigma) + l_{e_j}\}.$$ 

Hence,

$$C_{\text{max}}(\sigma) = C_{\ast}(\sigma) = \sigma(\ast) + p_{\ast} = \max_{j=1, \ldots, k} \{C_j(\sigma) + l_j\} + p_{\ast}.$$ 

For simplicity, we assume $p_{\ast} = 0$ since the makespan of a schedule with $p_{\ast} = c > 0$ differs from the makespan of the corresponding schedule with $p_{\ast} = 0$ by precisely $c$.

We now give two proofs that $1 / \text{min delays, } k \text{ 1-chains } / C_{\text{max}}$ is solved by sequencing jobs $J_1, \ldots, J_k$ in order of nonincreasing precedence delay. The first proof involves a straightforward pairwise interchange argument.

**Proposition 2.1** The $1 / \text{min delays, } k \text{ 1-chains } / C_{\text{max}}$ problem is solved by sequencing jobs $J_1, \ldots, J_k$ in order of nonincreasing precedence delay.
Proof 1: Assume $\sigma$ is an optimal schedule in which jobs $J_1, \ldots, J_k$ are not ordered by nonincreasing precedence delay. Then, there exist adjacent jobs $J_j$ and $J_j'$ such that $J_j \rightarrow J_j'$ but $l_j < l_{j'}$. Let $\sigma'$ be the schedule obtained from schedule $\sigma$ by interchanging jobs $J_j$ and $J_j'$. Let $\Delta = \max_{J_r \in J \setminus \{J_j, J_j', \ast\}} \{C_r(\sigma') + l_r\} = \max_{J_r \in J \setminus \{J_j, J_j', \ast\}} \{C_r(\sigma) + l_r\}$. Then

$$C_{\max}(\sigma') = \max\{\Delta, C_j(\sigma') + l_j, C_{j'}(\sigma') + l_{j'}\}$$

$$= \max\{\Delta, \sigma(j) + p_j + p_{j'} + l_j, \sigma(j') + p_j + p_{j'} + l_{j'}\}$$

$$\leq \max\{\Delta, \sigma(j) + p_j + p_{j'} + l_j, \sigma(j) + p_{j} + p_{j'} + l_{j'}, \sigma(j) + p_j + p_{j'} + l_{j'}\}$$

$$= \max\{\Delta, \sigma(j) + p_j + p_{j'} + l_{j'}\}$$

$$= \max\{\Delta, \sigma(j) + p_j + l_j, \sigma(j) + p_{j} + p_{j'} + l_{j'}\}$$

$$= \max\{\Delta, C_j(\sigma) + l_j, C_{j'}(\sigma) + l_{j'}\}$$

$$= C_{\max}(\sigma).$$

Repeating this argument, we see that schedule $\sigma$ can be transformed into a schedule in which jobs $J_1, \ldots, J_k$ are ordered by nonincreasing precedence delay without increasing the makespan. $\square$

The second proof of Proposition 2.1 relies on the following lemma, which gives a lower bound for the makespan of an optimal schedule for $1 / \min$ delays, $k$ 1-chains / $C_{\max}$.

**Lemma 2.2** Let $\sigma^*$ be an optimal schedule. Then

$$C_{\max}(\sigma^*) \geq h(S) = \sum_{J_j \in S} p_j + \min_{J_j \in S} l_j \ \forall \ S \subseteq J \setminus \{\ast\}.$$ 

**Proof:** Let $S \subseteq J \setminus \{\ast\}$ and let $J_m = \arg\max_{J_j \in S}\{\sigma^*(j)\}$. Then

$$C_m(\sigma^*) = \sigma^*(m) + p_m \geq \sum_{J_j \in S} p_j.$$
It follows that

\[ C_{\text{max}}(\sigma^*) = C_*(\sigma^*) \geq C_m(\sigma^*) + l_m \geq \sum_{j \in S} p_j + \min_{j \in S} l_j. \]  

We now present a second, more elegant proof that 1 / min delays, k 1-chains / \(C_{\text{max}}\) is solved by sequencing jobs \(J_1, \ldots, J_k\) in order of nonincreasing precedence delay. Lemma 2.2 and the following proof are after the manner of Carlier [5].

**Proof 2:** Let \(\sigma\) be the schedule associated with the sequence \(J_1 \rightarrow \cdots \rightarrow J_k \rightarrow \ast\), where \(l_1 \geq \cdots \geq l_k\). Let \(J_c\) be a critical job, that is, a job such that

\[ C_{\text{max}}(\sigma) = C_*(\sigma) = C_c(\sigma) + l_c. \]

Let \(S = \{1, \ldots, c\}\). By assumption, \(\min_{j \in S} l_j = l_c\). Since

\[ C_c(\sigma) = \sigma(c) + p_c = \sum_{j=1}^{c} p_j = \sum_{j \in S} p_j, \]

then

\[ C_{\text{max}}(\sigma) = \sum_{j \in S} p_j + \min_{j \in S} l_j = h(S). \]

The result follows from Lemma 2.2, since \(h(S)\) is a lower bound on the optimal makespan.  

The most time consuming step in solving the 1 / min delays, k 1-chains / \(C_{\text{max}}\) problem is sorting the jobs by precedence delay. Therefore, 1 / min delays, k 1-chains / \(C_{\text{max}}\) can be solved in time \(O(k \log k)\).

**2.1.2 1 / min delays, k 2, 1, \ldots, 1-chains / \(C_{\text{max}}\)**

We now show that whereas 1 / min delays, k 1-chains / \(C_{\text{max}}\) is solvable in polynomial time, the problem obtained from it by allowing one of the chains to include
two jobs is \textit{NP-hard}. An optimization problem is said to be NP-hard if the decision problem obtained from that problem by introducing an additional parameter, say \( y \), and asking the question, \textquotedblleft is there a solution with value at most (or at least) \( y \)?\textquotedblright\ is NP-complete. The decision problem version of \( 1 / \min \text{ delays, } k \geq 2, 1,...,1\text{-chains} / C_{\max} \), which we refer to appropriately as \( 1 / \min \text{ delays, } k \geq 2, 1,...,1\text{-chains} / C_{\max} \leq y \), is defined as follows.

**INSTANCE:** Job set \( J = \{J_{z_1}, J_{z_2}\} \cup \{J_2, \ldots, J_k\} \cup \{\ast\} \), processing requirement \( p_j \in \mathbb{Z}_+^+ \forall J_j \in J \), precedence relation \( P \) on \( J \) of the form \( P = \{< J_{z_1}, J_{z_2} >\} \cup \{< J_{z_2}, \ast >\} \cup \{< J_j, \ast >, j = 2, \ldots, k\} \), nonnegative integer minimum delays \( l_{z_1}, l_{z_2}, \) and \( l_j \) for \( j = 2, \ldots, k \), and a positive integer \( y \) (see Figure 2.3).

**QUESTION:** Is there a one-machine schedule for \( J \) (i.e., a function \( \sigma: J \rightarrow \mathbb{Z}_0^+ \) with \( \sigma(j) > \sigma(j') \) implying \( \sigma(j) \geq \sigma(j') + p_{j'} \)) that satisfies the minimum delay precedence constraints (i.e., \( \sigma(j') - C_j(\sigma) \geq l_j \forall < J_j, J_{j'} > \in P \), where \( C_j(\sigma) = \sigma(j) + p_j \forall J_j \in J \) and that meets the overall deadline (i.e., \( C_\ast(\sigma) \leq y \))?

The problem we use for the reduction is PARTITION, which is defined as follows.

**INSTANCE:** Index set \( A = \{1, \ldots, a\} \) and size \( s(j) \in \mathbb{Z}_+^+ \forall j \in A \).

**QUESTION:** Is there a subset \( A' \subseteq A \) such that \( \sum_{j \in A'} s(j) = \sum_{j \notin A' \setminus A'} s(j) \)?

Karp [15] contains a proof that PARTITION is NP-complete.

We now prove that \( 1 / \min \text{ delays, } k \geq 2, 1,...,1\text{-chains} / C_{\max} \) is NP-hard by showing that the corresponding decision problem is NP-complete.

**Proposition 2.3** The \( 1 / \min \text{ delays, } k \geq 2, 1,...,1\text{-chains} / C_{\max} \leq y \) problem is NP-complete.

**Proof:** Given any sequence of the jobs in \( J \), we can, in polynomial time, verify that the sequence is feasible and that the makespan of the associated schedule does not
exceed the overall deadline. Hence, $1 / \min \text{delays, } k \geq 2, 1, \ldots, 1\text{-chains} / C_{\max} \leq y$ is in NP.

Let $A = \{1, \ldots, a\}$ and $s(j) \in \mathbb{Z}_0^+$ for all $j \in A$ be any instance of PARTITION. We construct a corresponding instance of $1 / \min \text{delays, } k \geq 2, 1, \ldots, 1\text{-chains} / C_{\max} \leq y$ in polynomial time as follows:

\[
\begin{align*}
k &= a + 1; \\
p_{x_1} &= p_{x_2} = 1; \\
l_{x_1} &= l_{x_2} = \frac{1}{2} \sum_{j \in A} s(j); \\
p_j &= s(j - 1), j = 2, \ldots, k; \\
l_j &= 0, j = 2, \ldots, k; \\
p_* &= 0; \\
y &= \sum_{j \in A} s(j) + 2.
\end{align*}
\]
Since \( p_{z_1} + p_{z_2} + \sum_{j=2}^{k} p_j = y \), a schedule can have makespan at most \( y \) if and only if that schedule includes no machine idle time. In any feasible schedule without machine idle time, the sum of the processing requirements of jobs from \( \{J_2, \ldots, J_k\} \) scheduled between jobs \( J_{z_1} \) and \( J_{z_2} (J_{z_2} \text{ and } \ast) \) must be at least \( l_{z_1} = \frac{1}{2} \sum_{j \in A} s(j) \) \( (l_{z_2} = \frac{1}{2} \sum_{j \in A} s(j)) \). Now \( \sum_{j=2}^{k} p_j = \sum_{j \in A} s(j) \), so in any feasible schedule without machine idle time, the sum of the processing requirements of jobs from \( \{J_2, \ldots, J_k\} \) scheduled between jobs \( J_{z_1} \) and \( J_{z_2} \) must equal \( \frac{1}{2} \sum_{j \in A} s(j) \), as must the sum of the processing requirements of jobs from \( \{J_2, \ldots, J_k\} \) scheduled between jobs \( J_{z_2} \) and \( \ast \). Thus, there exists a feasible schedule that meets the overall deadline if and only if there exists a partition of \( \{J_2, \ldots, J_k\} \) into two disjoint subsets such that the sum of the processing requirements of the jobs in each subset equals \( \frac{1}{2} \sum_{j \in A} s(j) \).

Proposition 2.3 does not preclude a pseudo-polynomial time algorithm (i.e., an algorithm with running time bounded by a polynomial in \( \text{Max}[I] \) and \( \text{Length}[I] \), where, for each instance \( I \), \( \text{Max}[I] \) is an integer corresponding to the largest number in \( I \) and \( \text{Length}[I] \) is an integer corresponding to the number of symbols required to describe \( I \) under some reasonable encoding scheme (see [10])) for \( 1/\min \text{ delays, } k \geq 2, 1, \ldots, 1 \text{-chains } / C_{\text{max}} \). Whether or not such an algorithm exists is an open question.

2.1.3 \( 1/\min \text{ delays, } k \geq 2 \text{-chains } / C_{\text{max}} \)

Since \( 1/\min \text{ delays, } k \geq 2, 1, \ldots, 1 \text{-chains } / C_{\text{max}} \) is NP-hard, then so is \( 1/\min \text{ delays, } k \geq 2 \text{-chains } / C_{\text{max}} \). We now prove that \( 1/\min \text{ delays, } k \geq 2 \text{-chains } / C_{\text{max}} \) is NP-hard in the strong sense. A decision problem \( \Pi \) is said to be \( \text{NP-complete in the strong sense} \) if \( \Pi \) is NP-complete even if we permit unary or stroke encoding of numbers (e.g., 4 is encoded as 1111). An optimization problem is said to be NP-hard in the strong sense if the related decision problem is NP-complete in the strong sense. The decision problem version of \( 1/\min \text{ delays, } k \geq 2 \text{-chains } / C_{\text{max}} \), which we refer to as \( 1/\min \text{ delays, } k \geq 2 \text{-chains } / C_{\text{max}} \leq y \), is defined as follows.
INSTANCE: Job set $J = \bigcup_{j=1}^{k} \{ J_{j1}, J_{j2} \} \cup \{ * \}$, processing requirement $p_j \in \mathbb{Z}_0^+ \ \forall \ J_j \in J$, precedence relation $P$ on $J$ of the form $P = \{ < J_{j1}, J_{j2} >, \ j = 1, \ldots, k \} \cup \{ < J_{j2}, * >, \ j = 1, \ldots, k \}$, nonnegative integer minimum delays $l_{j1}$ and $l_{j2}$ for $j = 1, \ldots, k$, and a positive integer $y$ (see Figure 2.4).

QUESTION: Is there a one-machine schedule for $J$ (i.e., a function $a: J \rightarrow \mathbb{Z}^+$, with $a(j) > a(j')$ implying $a(j) \geq a(j') + p_j$) that satisfies the minimum delay precedence constraints (i.e., $a(j') - C_j(a) \geq l_j \ \forall \ < J_j, J_{j'} > \in P$, where $C_j(a) = a(j) + p_j \ \forall \ J_j \in J$) and that meets the overall deadline (i.e., $C_\ast(a) \leq y$)?

Our reduction is from SEQUENCING WITHIN INTERVALS, which is defined as follows.

INSTANCE: Task set $T = \{ 1, \ldots, t \}$ and, for each task $j \in T$, a length $l(j) \in \mathbb{Z}^+$, a release time $r(j) \in \mathbb{Z}_0^+$, and a deadline $d(j) \in \mathbb{Z}^+$.

QUESTION: Is there a one-machine schedule for $T$ (i.e., a function $\phi: T \rightarrow \mathbb{Z}_0^+$, with $\phi(j) > \phi(j')$ implying $\phi(j) \geq \phi(j') + l(j')$) that satisfies the release time...
constraints and meets all the deadlines (i.e., for all \( j \in T \), \( \phi(j) \geq r(j) \) and \( \phi(j) + l(j) \leq d(j) \)?)

See Garey and Johnson [10] for a proof that SEQUENCING WITHIN INTERVALS is NP-complete in the strong sense.

We now establish the computational complexity of \( \text{1/min delays, } k \text{-chains} / C_{\text{max}} \leq y \).

**Proposition 2.4** The \( \text{1/min delays, } k \text{-chains} / C_{\text{max}} \leq y \) problem is NP-complete in the strong sense.

**Proof:** We can show that a decision problem \( \Pi' \) is NP-complete in the strong sense by proving

1. \( \Pi' \) is in NP and
2. there exists a pseudo-polynomial transformation \( f \) from some known strongly NP-complete problem \( \Pi \) to \( \Pi' \).

Let \( D_{\Pi}, D_{\Pi'}, Y_{\Pi}, Y_{\Pi'}, \text{Length}, \text{Length}', \text{Max}, \) and \( \text{Max}' \) be the instance sets, ‘yes’ sets, and specified length and max functions corresponding to problems \( \Pi \) and \( \Pi' \), respectively. Garey and Johnson [10] define a pseudo-polynomial transformation from \( \Pi \) to \( \Pi' \) as a function \( f : D_{\Pi} \rightarrow D_{\Pi'} \) that satisfies

1. \( f \) can be computed in time polynomial in the two variables \( \text{Length}[I] \) and \( \text{Max}[I] \),
2. there exists a polynomial \( q_1 \) such that, for all \( I \in D_{\Pi} \),
   \( q_1(\text{Length}[f(I)]) \geq \text{Length}[I] \),
3. there exists a polynomial \( q_2 \) such that, for all \( I \in D_{\Pi} \),
   \( \text{Max}'[f(I)] \leq q_2(\text{Length}[I], \text{Max}[I]) \), and
4. for all \( I \in D_{\Pi}, I \in Y_{\Pi} \) if and only if \( f(I) \in Y_{\Pi'} \).
The pseudo-polynomial transformation is to the class of strongly NP-complete problems as the polynomial transformation is to the class of NP-complete problems. Since every NP-complete problem can be polynomially transformed to a given NP-complete problem, then the existence of a polynomial algorithm for any NP-complete problem implies the existence of a polynomial algorithm for every NP-complete problem, whence \( P = NP \). Since every strongly NP-complete problem can be pseudo-polynomially transformed to a given strongly NP-complete problem, then the existence of a pseudo-polynomial algorithm for any strongly NP-complete problem implies the existence of a pseudo-polynomial algorithm for every strongly NP-complete problem, whence \( P = NP \).

Given any sequence of the jobs in \( J \), we can, in polynomial time, verify that the sequence is feasible and that the makespan of the associated schedule does not exceed the overall deadline. Thus, \( 1 / \text{min delays, } k \text{ 2-chains} / C_{\text{max}} \leq y \) is in NP.

Let task set \( T = \{1, \ldots, t\} \) and, for each task \( j \in T \), length \( l(j) \in \mathbb{Z}^+ \), release time \( r(j) \in \mathbb{Z}_0^+ \), and deadline \( d(j) \in \mathbb{Z}^+ \) be any instance of SEQUENCING WITHIN INTERVALS. We construct a corresponding instance of \( 1 / \text{min delays, } k \text{ 2-chains} / C_{\text{max}} \leq y \) as follows:

\[
\begin{align*}
k &= t; \\
p_{j1} &= 0, \quad p_{j2} = l(j), \quad j = 1, \ldots, k; \\
p_* &= 0; \\
l_{j1} &= r(j), \quad j = 1, \ldots, k; \\
y &= \max_{j=1, \ldots, t} d(j); \\
l_{j2} &= y - d(j), \quad j = 1, \ldots, k.
\end{align*}
\]

One can easily verify that this mapping from SEQUENCING WITHIN INTERVALS to \( 1 / \text{min delays, } k \text{ 2-chains} / C_{\text{max}} \leq y \) satisfies the computation time and instance size requirements for a pseudo-polynomial transformation.
Suppose there exists a schedule \( \sigma : J \rightarrow \mathbb{Z}_0^+ \) that satisfies the minimum delay precedence constraints and meets the overall deadline. Let us define schedule \( \phi : T \rightarrow \mathbb{Z}_0^+ \) by \( \phi(j) = \sigma(j2) \) for \( j = 1, \ldots, t = k \). Since \( l_j = r(j) \), then \( \sigma(j2) \geq r(j) \) for \( j = 1, \ldots, k \), which implies \( \phi(j) \geq r(j) \) for \( j = 1, \ldots, t \). Now

\[
\sigma(j2) + p_{j2} + 1_{j2} \leq \sigma(*) \leq y \text{ for } j = 1, \ldots, k,
\]

which implies

\[
\sigma(j2) + p_{j2} \leq y - (y - d(j)) = d(j) \text{ for } j = 1, \ldots, k.
\]

Hence, \( \phi(j) + l(j) \leq d(j) \) for \( j = 1, \ldots, t \) and schedule \( \phi \) is feasible for SEQUENCING WITHIN INTERVALS.

Now, suppose there exists a schedule \( \phi : T \rightarrow \mathbb{Z}_0^+ \) such that, for all \( j \in T \), \( \phi(j) \geq r(j) \) and \( \phi(j) + l(j) \leq d(j) \). Let us define schedule \( \sigma : J \rightarrow \mathbb{Z}_0^+ \) by

\[
\begin{align*}
\sigma(j1) &= 0 \text{ for } j = 1, \ldots, k, \\
\sigma(j2) &= \phi(j) \text{ for } j = 1, \ldots, k = t, \text{ and} \\
\sigma(*) &= \max_{m=1, \ldots, k} \{ \sigma(m2) + p_m2 + l_m2 \}.
\end{align*}
\]

Schedule \( \sigma \) satisfies the minimum delay precedence constraints corresponding to \( < J_{j1}, J_{j2} > \) for \( j = 1, \ldots, k \) since

\[
\sigma(j2) - (\sigma(j1) + p_{j1}) = \sigma(j2) = \phi(j) \geq r(j).
\]

By definition of \( \sigma(*) \), schedule \( \sigma \) satisfies the minimum delay precedence constraints corresponding to \( < J_{j2}, * > \) for \( j = 1, \ldots, k \). Finally, schedule \( \sigma \) meets the overall deadline since

\[
\phi(j) + l(j) \leq d(j) \text{ for } j = 1, \ldots, t \Rightarrow \sigma(j2) + p_{j2} \leq d(j) \text{ for } j = 1, \ldots, k,
\]

which implies
\[ C_*(\sigma) = \sigma(*) \]
\[ = \max_{m=1,...,k}\{\sigma(m_2) + p_{m_2} + l_{m_2}\} \]
\[ \leq \max_{m=1,...,k}\{d(m) + l_{m_2}\} \]
\[ = \max_{m=1,...,k}\{d(m) + y - d(m)\} \]
\[ = y. \square \]

2.1.4 \( 1 / \min \) delays, \( k \ n_1, 1, \ldots, 1 \)-chains / \( C_{\max} \)

The \( 1 / \min \) delays, \( k \ n_1, 1, \ldots, 1 \)-chains / \( C_{\max} \) problem is NP-hard, since, as shown in Subsection 2.1.2, \( 1 / \min \) delays, \( k \ 2, 1, \ldots, 1 \)-chains / \( C_{\max} \) is NP-hard. In this subsection, we prove that \( 1 / \min \) delays, \( k \ n_1, 1, \ldots, 1 \)-chains / \( C_{\max} \) is in fact NP-hard in the strong sense.

The decision problem version of \( 1 / \min \) delays, \( k \ n_1, 1, \ldots, 1 \)-chains / \( C_{\max} \), which we refer to as \( 1 / \min \) delays, \( k \ n_1, 1, \ldots, 1 \)-chains / \( C_{\max} \leq y \), is defined as follows.

**INSTANCE:** Job set \( J = \{J_1, \ldots, J_{n_1}\} \cup \{J_2, \ldots, J_k\} \cup \{*, \} \), processing requirement \( p_j \in \mathbb{Z}_+^+ \), \( \forall \ j \in J \), precedence relation \( P \) on \( J \) of the form \( P = \{< J_{j_1}, J_{j_2} >, \ j = 1, \ldots, n_1 - 1\} \cup \{< J_{n_1}, *, >\} \cup \{< J_j, * >, \ j = 2, \ldots, k\} \), nonnegative integer minimum delays \( l_j \) for \( j = 1, \ldots, n_1 \) and \( l_j \) for \( j = 2, \ldots, k \), and a positive integer \( y \) (see Figure 2.5).

**QUESTION:** Is there a one-machine schedule for \( J \) (i.e., a function \( \sigma : J \to \mathbb{Z}_0^+ \), with \( \sigma(j) > \sigma(j') \) implying \( \sigma(j) \geq \sigma(j') + p_{j'} \)) that satisfies the minimum delay precedence constraints (i.e., \( \sigma(j') - C_j(\sigma) \geq l_j \), \( \forall < J_j, J_{j'} > \in P \), where \( C_j(\sigma) = \sigma(j) + p_j \), \( \forall J_j \in J \)) and that meets the overall deadline (i.e., \( C_*(\sigma) \leq y \))?

The problem we use for the reduction is 3-PARTITION, which is defined as follows.

**INSTANCE:** Index set \( T = \{1, \ldots, 3t\} \) and positive integers \( a_1, \ldots, a_{3t} \), and \( b \), with \( a_j \in (\frac{1}{4}b, \frac{1}{2}b) \), \( \forall j \in T \) and \( \sum_{j \in T} a_j = tb \).
Figure 2.5: Instance of $1/\min$ delays, $k \ n_1, 1, \ldots, 1$-chains / $C_{\max} \leq y$. 
QUESTION: Can $T$ be partitioned into $t$ disjoint subsets $T_1, \ldots, T_t$ such that 
$\sum_{j \in T_i} a_j = b$ for $i = 1, \ldots, t$?

Garey and Johnson [10] contains a proof that 3-PARTITION is NP-complete in the strong sense.

We now prove that the decision problem version of 1 / min delays, $k n_1, 1, \ldots, 1$-chains / $C_{\text{max}}$ is strongly NP-complete, and hence, the optimality version is NP-hard in the strong sense.

**Proposition 2.5** The 1 / min delays, $k n_1, 1, \ldots, 1$-chains / $C_{\text{max}} \leq y$ problem is NP-complete in the strong sense.

**Proof:** The 1 / min delays, $k n_1, 1, \ldots, 1$-chains / $C_{\text{max}} \leq y$ problem is in NP since, given any sequence of the jobs in $J$, we can, in polynomial time, verify that the sequence is feasible and that the makespan of the associated schedule does not exceed the overall deadline.

Let $T = \{1, \ldots, 3t\}$ and positive integers $a_1, \ldots, a_{3t}$, and $b$, with $a_j \in (\frac{1}{4}b, \frac{1}{2}b)$ for all $j \in T$ and $\sum_{j \in T} a_j = tb$ be any instance of 3-PARTITION. We construct a corresponding instance of 1 / min delays, $k n_1, 1, \ldots, 1$-chains / $C_{\text{max}} \leq y$ as follows:

$k = 3t + 1, \ n_1 = t$;

$p_{z_j} = 1, \ j = 1, \ldots, n_1$;

$l_{z_j} = b, \ j = 1, \ldots, n_1$;

$p_j = a_{j-1}, \ j = 2, \ldots, k$;

$l_j = 0, \ j = 2, \ldots, k$;

$p_\ast = 0$;

$y = tb + t$. 

25
One can easily verify that this mapping from 3-PARTITION to $1 \text{/ min delays, } k n_1, 1, \ldots, 1$-chains / $C_{\text{max}} \leq y$ satisfies the computation time and instance size requirements for a pseudo-polynomial transformation.

Since $\sum_{j=1}^{n} p_j + \sum_{j=2}^{k} p_j = y$, a schedule can have makespan at most $y$ if and only if that schedule includes no machine idle time. In any feasible schedule without machine idle time, the sum of the processing requirements of jobs from $\{J_2, \ldots, J_k\}$ scheduled between jobs $J_x$, and $J_{x+1}$ ($J_{x_1}$ and *) must be at least $l_x = b$ for $j = 1, \ldots, n_1 - 1 = t - 1$ ($l_{x_1} = b$). Since $\sum_{j=2}^{k} p_j = tb$, then in any feasible schedule without machine idle time, the sum of the processing requirements of jobs from $\{J_2, \ldots, J_k\}$ scheduled between jobs $J_x$, and $J_{x+1}$ must equal $b$ for $j = 1, \ldots, n_1 - 1$ and the sum of the processing requirements of jobs from $\{J_2, \ldots, J_k\}$ scheduled between jobs $J_{x_1}$, and * must equal $b$. Thus, there exists a feasible schedule that meets the overall deadline if and only if there exists a partition of $\{J_2, \ldots, J_k\}$ into $t$ disjoint subsets such that the sum of the processing requirements of the jobs in each subset equals $b$. □

### 2.2 Max Delays Problems

Without loss of generality, solutions to $1 \text{/ max delays, } k n_1, \ldots, n_k$-chains / $C_{\text{max}}$ problems include no machine idle time, since removing machine idle time from a schedule that satisfies the maximum delay precedence constraints results in a feasible schedule with smaller makespan. Schedules without machine idle time are necessarily minimum makespan schedules. Thus, $1 \text{/ max delays, } k n_1, \ldots, n_k$-chains / $C_{\text{max}}$ is the problem of finding a schedule without machine idle time that satisfies the maximum delay precedence constraints.

#### 2.2.1 $1 \text{/ max delays, } k$ 1-chains / $C_{\text{max}}$

We first consider the max delays problem with the simplest $k$ chains precedence relation, that is, $k$ 1-chains. For ease of notation, assume $J = \{J_1, \ldots, J_k\} \cup \{\ast\}$ (see
Figure 2.6: Instance of 1 / max delays, k 1-chains / $C_{\text{max}}$.

Figure 2.6). In addition, let $p(S) = \sum_{J \in S} p_J$ for all $S \subseteq J$. As before, we assume $p_* = 0$.

Suppose that for some $S \subseteq J \setminus \{\ast\}$, $u_j < p(S \setminus \{J_j\})$ for all $J_j \in S$. Then, there can be no feasible schedule since, in any schedule, the maximum delay precedence constraint corresponding to the earliest scheduled job in $S$ will be violated. We will refer to a subset $S \subseteq J \setminus \{\ast\}$ having the property $u_j < p(S \setminus \{J_j\})$ for all $J_j \in S$ as a blocking subset.

In any feasible schedule without machine idle time, job $\ast$ starts at time $p(J)$. Thus, the completion time of job $J_j$ in any feasible schedule without idle time must be at least $p(J) - u_j$ for all $j = 1, \ldots, k$. Equivalently, the start time of job $J_j$ in any feasible schedule without idle time must be at least $p(J \setminus \{J_j\}) - u_j$ for all $j = 1, \ldots, k$. Define release dates $r_j = p(J \setminus \{J_j\}) - u_j$ for all $j = 1, \ldots, k$. We now show that the 1 / max delays, k 1-chains / $C_{\text{max}}$ problem is solved by sequencing the jobs $J_1, \ldots, J_k$ in order of nondecreasing release date.

**Proposition 2.6** Assume $r_1 \leq \cdots \leq r_k$. Then, either the instance of 1 / max delays, $k$ 1-chains / $C_{\text{max}}$ is infeasible or the schedule without machine idle time corresponding to the sequence $J_1 \rightarrow \cdots \rightarrow J_k \rightarrow *$ is optimal.

**Proof:** Let $\sigma$ be the schedule without machine idle time corresponding to $J_1 \rightarrow \cdots \rightarrow J_k \rightarrow *$. Suppose there exists $i \in \{1, \ldots, k\}$ such that $\sigma(i) < r_i$. Let
$S = \{ J_j : \sigma(j) \geq \sigma(i), j = 1, \ldots, k \}$. Since $r_1 \leq \cdots \leq r_k$, then

$$r_j \geq r_i \geq \sigma(i) = p(J \setminus S) \forall J_j \in S.$$  

Thus,

$$p(J \setminus \{ J_j \}) - u_j = r_j > p(J \setminus S) \forall J_j \in S,$$

which implies

$$u_j < p(S \setminus \{ J_j \}) \forall J_j \in S.$$

By definition, $S$ is a blocking subset and the instance is infeasible.

On the other hand, suppose $\sigma(j) \geq r_j$ for $j = 1, \ldots, k$. Then, for each $j = 1, \ldots, k$,

$$0 \geq r_j - \sigma(j) = p(J \setminus \{ J_j \}) - u_j - \sigma(j) = \sigma(*) - p_j - u_j - \sigma(j),$$

which implies $\sigma(*) - C_j(\sigma) \leq u_j$ for each $j = 1, \ldots, k$. Therefore, schedule $\sigma$ is feasible and hence optimal. $\square$

Sorting the jobs according to release date is the most time consuming step in solving $1 / \text{max delays, } k \text{ 1-chains } / C_{\text{max}}$. Thus, the $1 / \text{max delays, } k \text{ 1-chains } / C_{\text{max}}$ problem can be solved in time $O(k \lg k)$.

### 2.2.2 $1 / \text{max delays, } k \text{ 2, 1, . . . , 1-chains } / C_{\text{max}}$

We proved in the previous subsection that $1 / \text{max delays, } k \text{ 1-chains } / C_{\text{max}}$ is solvable in polynomial time. We now show that the problem obtained from it by allowing one of the chains to include two jobs is NP-hard. In other words, we prove that determining whether or not there exists a schedule without machine idle time that satisfies the maximum delay precedence constraints, where the precedence relation is $k \text{ 2, 1, . . . , 1-chains}$, is NP-complete. This decision problem, which we refer to as $1 / \text{max delays, } k \text{ 2, 1, . . . , 1-chains } / C_{\text{max}} \leq p(J)$, is defined as follows.

**INSTANCE:** Job set $J = \{ J_{x_1}, J_{x_2} \} \cup \{ J_2, \ldots, J_k \} \cup \{ \ast \}$, processing requirement $p_j \in Z_0^+ \forall J_j \in J$, precedence relation $P$ on $J$ of the form $P = \{ < J_{x_1}, J_{x_2} > \}$
Figure 2.7: Instance of $1/\text{max delays, } k \geq 2, 1,..., 1\text{-chains} / C_{\text{max}} \leq p(J)$.

$\cup \{< J_{x_0}, * >\} \cup \{< J_j, * >, j = 2, \ldots, k\}$, maximum delays $u_{x_1}, u_{x_2}$, and $u_j$ for $j = 2, \ldots, k$, where each maximum delay is either infinite or a nonnegative integer (see Figure 2.7).

**QUESTION:** Is there a one-machine schedule for $J$ (i.e., a function $\sigma: J \rightarrow \mathbb{Z}_0^+$, with $\sigma(j) > \sigma(j')$ implying $\sigma(j) \geq \sigma(j') + p_{j'}$) that satisfies the maximum delay precedence constraints (i.e., $\sigma(j') - C_j(\sigma) \leq u_j \forall < J_j, J_{j'} > \in P$, where $C_j(\sigma) = \sigma(j) + p_j \forall J_j \in J$) and that meets the overall deadline (i.e., $C_*(\sigma) \leq p(J)$)?

We now show that the decision problem version of $1/\text{max delays, } k \geq 2, 1,..., 1\text{-chains} / C_{\text{max}}$ is NP-complete.

**Proposition 2.7** The $1/\text{max delays, } k \geq 2, 1,..., 1\text{-chains} / C_{\text{max}} \leq p(J)$ problem is NP-complete.

**Proof:** The $1/\text{max delays, } k \geq 2, 1,..., 1\text{-chains} / C_{\text{max}} \leq p(J)$ problem is in NP since, given any sequence of the jobs in $J$, we can, in polynomial time, verify that
Figure 2.8: Feasible schedule for constructed instance of $1 / \text{max delays, } k, 2, 1, \ldots, 1$-chains / $C_{\text{max}} \leq p(J)$.

the sequence is feasible and that the associated schedule without machine idle time satisfies the maximum delay precedence constraints.

Let $A = \{1, \ldots, a\}$ and $s(j) \in \mathbb{Z}_0^+$ for all $j \in A$ be any instance of PARTITION (see page 16). We construct a corresponding instance of $1 / \text{max delays, } k, 2, 1, \ldots, 1$-chains / $C_{\text{max}} \leq p(J)$ in polynomial time as follows:

\[
\begin{align*}
\hat{k} &= a + 1; \\
p_{x_1} &= p_{x_2} = 1; \\
u_{x_1} &= u_{x_2} = \frac{1}{2} \sum_{j \in A} s(j); \\
p_j &= s(j - 1), j = 2, \ldots, k; \\
u_j &= \sum_{i \in A, i \neq j-1} s(i) + 1, j = 2, \ldots, k; \\
p_* &= 0.
\end{align*}
\]

Suppose there exists a subset $A' \subseteq A$ such that $\sum_{j \in A'} s(j) = \sum_{j \in A \setminus A'} s(j) = \frac{1}{2} \sum_{j \in A} s(j)$. Let $S = \{J_i \in \{J_2, \ldots, J_k\}: s(j - 1) \in A'\}$ and let $T = \{J_2, \ldots, J_k\} \setminus S$. The schedule illustrated in Figure 2.8 satisfies the maximum delay precedence constraints and meets the overall deadline.

On the other hand, suppose there exists no subset $A' \subseteq A$ such that $\sum_{j \in A} s(j) = \sum_{j \in A \setminus A'} s(j)$. Let $\sigma$ be a schedule that satisfies the maximum delay precedence constraints and meets the overall deadline. Since $u_{x_1} = u_{x_2} = \frac{1}{2} \sum_{j \in A} s(j)$ and by
our hypothesis concerning the nonexistence of a partition for \( A \), the sum of the processing requirements of jobs from \( \{ J_2, \ldots, J_k \} \) scheduled either between jobs \( J_{x_1} \) and \( J_{x_2} \) or between jobs \( J_{x_2} \) and \( * \) must be less than \( \sum_{j \in A} s(j) \). Thus, some job \( J_j \in \{ J_2, \ldots, J_k \} \) must precede job \( J_{x_1} \) in schedule \( \sigma \). Of all such jobs, let \( J_{j'} \) be the job scheduled earliest. Now

\[
\sigma(*) - C_j'(\sigma) \geq p(J \setminus \{ J_{j'} \}) = \sum_{i \in A, i \neq j'-1} s(i) + 2 > u_{j'},
\]

which contradicts our assumption that schedule \( \sigma \) satisfies the maximum delay precedence constraints. Hence, there exists a schedule that satisfies the maximum delay precedence constraints and meets the overall deadline if and only if there exists a subset \( A' \subseteq A \) such that \( \sum_{j \in A'} s(j) = \sum_{j \in A \setminus A'} s(j) \). \( \square \)

Proposition 2.7 notwithstanding, there might exist a pseudo-polynomial time algorithm for 1 / max delays, \( k = 2, 1, \ldots, 1 \)-chains / \( C_{max} \). Whether or not such an algorithm exists is an open question. As a result of Proposition 2.7, the 1 / max delays, \( k = 2 \)-chains / \( C_{max} \) problem is NP-hard. Whether or not 1 / max delays, \( k = 2 \)-chains / \( C_{max} \) is NP-hard in the strong sense, as is 1 / min delays, \( k = 2 \)-chains / \( C_{max} \), is also an open question.

2.2.3 1 / max delays, \( k = n_1, 1, \ldots, 1 \)-chains / \( C_{max} \)

The 1 / max delays, \( k = n_1, 1, \ldots, 1 \)-chains / \( C_{max} \) problem is NP-hard, since, as shown in Subsection 2.2.2, 1 / max delays, \( k = 2, 1, \ldots, 1 \)-chains / \( C_{max} \) is NP-hard. In this subsection, we prove that 1 / max delays, \( k = n_1, 1, \ldots, 1 \)-chains / \( C_{max} \) is in fact NP-hard in the strong sense. The decision problem version of 1 / max delays, \( k = n_1, 1, \ldots, 1 \)-chains / \( C_{max} \), which we refer to as 1 / max delays, \( k = n_1, 1, \ldots, 1 \)-chains / \( C_{max} \leq p(J) \), is defined as follows.

**INSTANCE:** Job set \( J = \{ J_{x_1}, \ldots, J_{x_{n_1}} \} \cup \{ J_2, \ldots, J_k \} \cup \{ * \} \), processing requirement \( p_j \in \mathbb{Z}_0^+ \forall J_j \in J \), precedence relation \( P \) on \( J \) of the form \( P = \)
Figure 2.9: Instance of $1 / \text{max delays}$, $k$ $n_1, 1, \ldots, 1$-chains $/ C_{\text{max}} \leq p(J)$.

$\{< J_{x_j}, J_{x_{j+1}} >, j = 1, \ldots, n_1 - 1 \} \cup \{< J_{x_{n_1}}, * > \} \cup \{< J_{j}, * >, j = 2, \ldots, k \}$

maximum delays $u_{x_j}$ for $j = 1, \ldots, n_1$ and $u_j$ for $j = 2, \ldots, k$, where each maximum delay is either infinite or a nonnegative integer (see Figure 2.9).

**QUESTION:** Is there a one-machine schedule for $J$ (i.e., a function $\sigma : J \rightarrow Z^+_0$, with $\sigma(j) > \sigma(j')$ implying $\sigma(j) \geq \sigma(j') + p_{j'}$) that satisfies the maximum delay precedence constraints (i.e., $\sigma(j') - C_j(\sigma) \leq u_j \ \forall \ < J_j, J_{j'} > \in P$,

where $C_j(\sigma) = \sigma(j) + p_j \ \forall \ J_j \in J$) and that meets the overall deadline (i.e., $C_*(\sigma) \leq p(J)$)?

We now establish the computational complexity of $1 / \text{max delays}$, $k$ $n_1, 1, \ldots, 1$-chains $/ C_{\text{max}} \leq p(J)$. 

32
Proposition 2.8 The 1 / max delays, k n1,1, ... , k-chains / Cmax ≤ p(J) problem is NP-complete in the strong sense.

Proof: Given any sequence of the jobs in J, we can, in polynomial time, verify that the sequence is feasible and that the associated schedule without machine idle time satisfies the maximum delay precedence constraints. Thus, 1 / max delays, k n1,1, ... , k-chains / Cmax ≤ p(J) is in NP.

Let index set T = \{1, ..., 3t\} and positive integers a1, ..., a3t, and b, with a_j ∈ (1/4b, 1/2b) ∀ j ∈ T and \(\sum_{j \in T} a_j = tb\) be any instance of 3-PARTITION (see page 23). We construct a corresponding instance of 1 / max delays, k n1,1, ... , k-chains / Cmax ≤ p(J) as follows:

\[
\begin{align*}
k &= 3t + 1, \quad n_1 = t; \\
p_{x_j} &= 1, \quad j = 1, \ldots, n_1; \\
u_{x_j} &= b, \quad j = 1, \ldots, n_1; \\
p_j &= a_{j-1}, \quad j = 2, \ldots, k; \\
u_j &= \sum_{i \in T, i \neq j-1} a_i + t - 1, \quad j = 2, \ldots, k; \\
p_* &= 0.
\end{align*}
\]

One can easily verify that this mapping from 3-PARTITION to 1 / max delays, k n1,1, ... , k-chains / Cmax ≤ p(J) satisfies the computation time and instance size requirements for a pseudo-polynomial transformation.

Suppose that T can be partitioned into t disjoint subsets T1, ..., Tt such that \(\sum_{j \in T_i} a_j = b\) for i = 1, ..., t. Let \(S_i = \{J_j \in \{J_2, \ldots, J_k\} : a_{j-1} \in T_i\}\) for i = 1, ..., t. The schedule illustrated in Figure 2.10 satisfies the maximum delay precedence constraints and meets the overall deadline.

Suppose, on the other hand, that T cannot be partitioned into t disjoint subsets T1, ..., Tt such that \(\sum_{j \in T_i} a_j = b\) for i = 1, ..., t. Let σ be a schedule that satisfies
the maximum delay precedence constraints and meets the overall deadline. Since $u_{x_j} = b$ for $j = 1, \ldots, n_1$ and by our hypothesis concerning the nonexistence of a partition for $T$, the sum of the processing requirements of jobs from $\{J_2, \ldots, J_k\}$ scheduled either between jobs $J_{x_j}$ and $J_{x_{j+1}}$ for $j = 1, \ldots, n_1 - 1$ or between jobs $J_{x_{n_1}}$ and * must be less than $tb$. Thus, some job $J_j \in \{J_2, \ldots, J_k\}$ must precede job $J_{x_1}$ in schedule $\sigma$. Of all such jobs, let $J_{j'}$ be the job scheduled earliest. Now

$$\sigma(*) - C_{j'}(\sigma) \geq p(J \setminus \{J_{j'}\}) = \sum_{i \in T, i \neq j-1} a_i + t > u_{j'},$$

which contradicts our assumption that schedule $\sigma$ satisfies the maximum delay precedence constraints. Hence, there exists a schedule that satisfies the maximum delay precedence constraints and meets the overall deadline if and only if there exists a partition of $T$ into $t$ disjoint subsets $T_1, \ldots, T_t$ such that $\sum_{j \in T_i} a_j = b$ for $i = 1, \ldots, t$. \hfill \Box

### 2.3 Min and Max Delays Problems

The problems of Sections 2.1 and 2.2 are special cases of minimum makespan problems subject to both minimum and maximum delay precedence constraints. Thus, min and max delays problems with all but the simplest precedence relation are NP-hard.

In this section, we establish the complexity of two problems subject to both minimum and maximum delay precedence constraints with precedence relation $k$ $1$-chains. The first, $1 / \text{min or max delays}, k$ $1$-chains / $C_{\text{max}}$ is solvable in polynomial
time. Recall that the 'or' of 'min or max delays' is an exclusive or so that \( l_j = 0 \) or \( u_j = \infty \) for all \( j = 1, \ldots, k \). The second problem, 1 / min and max delays, \( k \) 1-chains / \( C_{\text{max}} \), is NP-hard in the strong sense. In fact, even for \( k \)-chains, the problem of determining whether or not a feasible schedule exists is strongly NP-complete.

### 2.3.1 1 / min or max delays, \( k \) 1-chains / \( C_{\text{max}} \)

For convenience, assume \( J = \{J_1, \ldots, J_k\} \cup \{\ast\} \). Let \( S \subseteq \{J_1, \ldots, J_k\} \) consist of those jobs for which \( l_j > 0 \), and let \( T = \{J_1, \ldots, J_k\} \setminus S \). Using the algorithm below, we generate an optimal solution for the 1 / min or max delays, \( k \) 1-chains / \( C_{\text{max}} \) problem by combining the solution of the 1 / min delays, \( k \) 1-chains / \( C_{\text{max}} \) problem on jobs in \( S \cup \{\ast\} \), and the solution of the 1 / max delays, \( k \) 1-chains / \( C_{\text{max}} \) problem on jobs in \( T \cup \{\ast\} \).

**Proposition 2.9** The following algorithm solves the 1 / min or max delays, \( k \) 1-chains / \( C_{\text{max}} \) problem.

**1 / min or max delays, \( k \) 1-chains / \( C_{\text{max}} \) Algorithm**

**Step 1:** Solve the 1 / max delays, \( k \) 1-chains / \( C_{\text{max}} \) problem on jobs in \( T \cup \{\ast\} \) (see Subsection 2.2.1). If there is no feasible schedule for this max delays problem, then STOP: The instance of the min or max delays problem is INFEASIBLE.

**Step 2:** Solve the 1 / min delays, \( k \) 1-chains / \( C_{\text{max}} \) problem on jobs in \( S \cup \{\ast\} \) (see Subsection 2.1.1). Let \( C \) be the optimal makespan.

**Step 3:** Combine the solutions from Steps 1 and 2 as in Figure 2.11. Gap [\( C - p(J) \)] + is the minimum amount of idle time which must be inserted between jobs in \( S \) and jobs in \( T \) so as to satisfy the minimum delay precedence constraints corresponding to jobs in \( S \).

**Proof:** In Step 1, we either determine a feasible optimal schedule for jobs in \( T \cup \{\ast\} \), or we find a blocking subset of \( T \), whence the instance of the min or max delays problem is infeasible.
If $[C - p(J)]^+ = 0$, the schedule obtained in Step 3 includes no machine idle time and hence is optimal. On the other hand, if $[C - p(J)]^+ > 0$, the schedule obtained in Step 3 has makespan $C$, which, from Step 2, provides a lower bound on the optimal makespan. □

Sorting the jobs in $T$ by release date and the jobs in $S$ by precedence delay are the algorithm's most time consuming tasks. Therefore, the $1 / \min$ or $\max$ delays, $k$ 1-chains / $C_{\max}$ problem can be solved in time $O(k \lg k)$.

### 2.3.2 1 / min and max delays, $k$ 1-chains / $C_{\max}$

In the previous subsection, we proved that $1 / \min$ or $\max$ delays, $k$ 1-chains / $C_{\max}$ is solvable in polynomial time. In this subsection, we show that $1 / \min$ and $\max$ delays, $k$ 1-chains / $C_{\max}$ is NP-hard in the strong sense. In fact, we prove the stronger result that determining whether or not there exists a schedule that satisfies the minimum and maximum delay precedence constraints, where the precedence relation is $k$ 1-chains, is strongly NP-complete. The min and max delays, $k$ 1-chains feasibility problem is formally defined as follows.

**INSTANCE:** Job set $J = \{J_1, \ldots, J_k\} \cup \{\star\}$, processing requirement $p_j \in \mathbb{Z}_0^+ \forall J_j \in J$, precedence relation $P$ on $J$ of the form $P = \{< J_j, \star >, j = 1, \ldots, k\}$, minimum delays $l_j$ and maximum delays $u_j$, where $0 \leq l_j \leq u_j$, $l_j$ is a nonnegative integer, and $u_j$ is either infinite or a nonnegative integer for $j = 1, \ldots, k$. 

![Figure 2.11: Optimal schedule for the 1 / min or max delays, $k$ 1-chains / $C_{\max}$ problem.](image-url)
QUESTION: Is there a one-machine schedule for $J$ (i.e., a function $\sigma: J \to Z_0^+$, with $\sigma(j) > \sigma(j')$ implying $\sigma(j) \geq \sigma(j') + p_{j'}$) that satisfies the minimum and maximum delay precedence constraints (i.e., $l_j \leq \sigma(\ast) - C_j(\sigma) \leq u_j$, where $C_j(\sigma) = \sigma(j) + p_j \forall j = 1, \ldots, k$)?

We now establish the computational complexity of the min and max delays, $k$ 1-chains feasibility problem.

Proposition 2.10 The min and max delays, $k$ 1-chains feasibility problem is NP-complete in the strong sense.

Proof: Given a schedule $\sigma: J \to Z_0^+$, we can, in polynomial time, verify that $\sigma$ satisfies the minimum and maximum delay precedence constraints. Thus, min and max delays, $k$ 1-chains feasibility is in NP.

Let $T = \{1, \ldots, 3t\}$ and positive integers $a_1, \ldots, a_{3t}$, and $b$, with $a_j \in (\frac{1}{4}b, \frac{1}{2}b)$ for all $j \in T$ and $\sum_{j \in T} a_j = tb$ be any instance of 3-PARTITION (see page 23). We construct a corresponding instance of the min and max delays, $k$ 1-chains feasibility problem as follows:

- $k = 4t$;
- $p_j = a_j$, $l_j = 0$, $u_j = \sum_{i \in T, i \neq j} a_i + t - 1$ for $j = 1, \ldots, 3t$;
- $p_j = 1$, $l_j = u_j = (4t + 1 - j)b + 4t - j$ for $j = 3t + 1, \ldots, k$;
- $p_\ast = 0$.

One can easily verify that this mapping from 3-PARTITION to the min and max delays, $k$ 1-chains feasibility problem satisfies the computation time and instance size requirements for a pseudo-polynomial transformation.

By definition of $l_j$ and $u_j$ for $j = 3t + 1, \ldots, k$, any feasible schedule $\sigma$ satisfies

$$\sigma(J_{3t+1}) < \sigma(J_{3t+2}) < \cdots < \sigma(J_k) < \sigma(\ast)$$

and

37
\[
\sigma(J_{3t+2}) - C_{3t+1}(\sigma) = \sigma(J_{3t+3}) - C_{3t+2}(\sigma) = \ldots \\
= \sigma(J_k) - C_{k-1}(\sigma) = \sigma(*) - C_k(\sigma) = b.
\]

No job \( J_j \in \{J_1, \ldots, J_{3t}\} \) can precede job \( J_{3t+1} \) in any feasible schedule since
\[
p_{3t+1} + l_{3t+1, \ast} = tb + t > u_j \text{ for } j = 1, \ldots, 3t.
\]

Now, since \( \sum_{j=1}^{3t} p_j = tb \), then in any feasible schedule, the sum of the processing requirements of jobs from \( \{J_1, \ldots, J_{3t}\} \) scheduled between jobs \( J_j \) and \( J_{j+1} \) must equal \( b \) for \( j = 3t + 1, \ldots, k - 1 \) and the sum of the processing requirements of jobs from \( \{J_1, \ldots, J_{3t}\} \) scheduled between jobs \( J_k \) and \( \ast \) must equal \( b \). Thus, there exists a feasible schedule if and only if there exists a partition of \( \{J_1, \ldots, J_{3t}\} \) into \( t \) disjoint subsets such that the sum of the processing requirements of the jobs in each subset equals \( b \). \( \Box \)

To conclude this chapter, we summarize the computational complexity results obtained thus far. The complexities of min delays, max delays, and min and max delays problems are given in Tables 2.1, 2.2, and 2.3, respectively.
Table 2.1: Complexity classification of min delays, minimum makespan problems.

<table>
<thead>
<tr>
<th>Precedence Relation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$ 1-chains</td>
<td>$O(k \lg k)$</td>
</tr>
<tr>
<td>$k$ 2, 1, ..., 1-chains</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$k$ 2-chains</td>
<td>NP-hard in the strong sense</td>
</tr>
<tr>
<td>$k$ $n_1$, 1, ..., 1-chains</td>
<td>NP-hard in the strong sense</td>
</tr>
</tbody>
</table>

Table 2.2: Complexity classification of max delays, minimum makespan problems.

<table>
<thead>
<tr>
<th>Precedence Relation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$ 1-chains</td>
<td>$O(k \lg k)$</td>
</tr>
<tr>
<td>$k$ 2, 1, ..., 1-chains</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$k$ 2-chains</td>
<td>NP-hard</td>
</tr>
<tr>
<td>$k$ $n_1$, 1, ..., 1-chains</td>
<td>NP-hard in the strong sense</td>
</tr>
</tbody>
</table>

Table 2.3: Complexity classification of min and max delays, $k$ 1-chains, minimum makespan problems.

<table>
<thead>
<tr>
<th>Type of Delays</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>min or max delays</td>
<td>$O(k \lg k)$</td>
</tr>
<tr>
<td>min and max delays</td>
<td>NP-hard in the strong sense</td>
</tr>
</tbody>
</table>
CHAPTER 3

1 / min and max delays, 
\( k \) 1-chains / \( \sum C_j \) or \( \sum w_j C_j \)

In this chapter, we draw the line between easy and hard total completion time 
and total weighted completion time problems with precedence relation \( k \) 1-chains. 
In Section 3.1, minimum delay precedence constraints only are allowed, while in 
Section 3.2, maximum delay precedence constraints only are allowed.

3.1 Min Delays Problems

Recall from Chapter 2 that associated with each feasible sequence is an active 
schedule that schedules not only job \(*\), but also each job as early as possible so 
as to respect the sequence and to satisfy the machine capacity and the minimum 
delay precedence constraints. Thus, 1 / min delays, \( k \) 1-chains / \( \sum C_j \) (\( \sum w_j C_j \)) is 
the problem of finding, among all feasible sequences, a sequence that has associated 
active schedule with minimum total (weighted) completion time.

3.1.1 1 / min delays, \( k \) 1-chains / \( \sum C_j \)

In this subsection, we show that 1 / min delays, \( k \) 1-chains / \( \sum C_j \) can be 
solved in time \( O(k^3 \log k) \). For ease of notation, assume \( J = \{J_1, \ldots, J_k\} \cup \{\ast\} \). Let 
schedule MM be the schedule associated with the sequence obtained by ordering 
jobs \( J_1, \ldots, J_k \) by nonincreasing precedence delay. As proved in Subsection 2.1.1, 
schedule MM has minimum makespan (i.e., minimum completion time for job \(*\))
among all feasible schedules. Let schedule SPT be the schedule associated with the sequence obtained by ordering jobs $J_1, \ldots, J_k$ using the shortest processing time rule (i.e., ordering the jobs by nondecreasing processing requirement), with ties broken in favor of the job with the largest precedence delay. As shown by Smith [23], schedule SPT solves the problem of minimizing total completion time for jobs $J_1, \ldots, J_k$.

We now show that the makespans of the MM and SPT schedules bound the optimal makespan for $1 / \min \delays, k$ 1-chains $/ \sum C_j$.

**Proposition 3.1** Let $\sigma^*$ be an optimal schedule for $1 / \min \delays, k$ 1-chains $/ \sum C_j$. Then $C_\ast(MM) \leq C_\ast(\sigma^*) \leq C_\ast(SPT)$.

**Proof:** Schedule MM has minimum makespan among all feasible schedules, so $C_\ast(MM) \leq C_\ast(\sigma^*)$. Suppose $C_\ast(\sigma^*) > C_\ast(SPT)$. Among all feasible schedules, schedule SPT has minimum total completion time for jobs $J_1, \ldots, J_k$, which implies $\sum_{j=1}^k C_j(\sigma^*) \geq \sum_{j=1}^k C_j(SPT)$. Thus,

$$\sum_{j=1}^k C_j(\sigma^*) + C_\ast(\sigma^*) > \sum_{j=1}^k C_j(SPT) + C_\ast(SPT),$$

a contradiction of the fact that $\sigma^*$ is an optimal schedule. $\square$

Let $J_{e_1} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow *$ be any feasible sequence. The schedule, $\sigma$, associated with this sequence has

$$\sigma(e_1) = 0,$$

$$\sigma(e_j) = \sum_{i=1}^{j-1} p_{e_i} \text{ for } j = 2, \ldots, k, \text{ and}$$

$$\sigma(*) = \max_{j=1, \ldots, k} \{ \sigma(e_j) + p_{e_j} + l_{e_j} \}.$$

Hence,

$$C_\ast(\sigma) = \max_{j=1, \ldots, k} \{ C_j(\sigma) + l_j \} + p_*.$$
For simplicity, we assume \( p_* = 0 \) since the total completion time of a schedule with \( p_* > 0 \) differs from the total completion time of the corresponding schedule with \( p_* = 0 \) by precisely \( c \).

Let \( C \in \{ C_*(MM), C_*(MM) + 1, \ldots, C_*(SPT) \} \). The following proposition characterizes those schedules that have makespan at most \( C \) (see [24]).

**Proposition 3.2** If \( \sigma \) is any schedule, then \( C_*(\sigma) \leq C \) if and only if \( \sigma \) meets the individual job deadlines \( d_j = C - l_j \) for all \( j = 1, \ldots, k \).

**Proof:** If \( C_*(\sigma) = \max_{j=1,\ldots,k} \{ C_j(\sigma) + l_j \} \leq C \), then
\[
C_j(\sigma) \leq C - l_j = d_j \quad \text{for all } j = 1, \ldots, k.
\]

If \( C_j(\sigma) \leq d_j = C - l_j \) for all \( j = 1, \ldots, k \), then
\[
C_*(\sigma) = \max_{j=1,\ldots,k} \{ C_j(\sigma) + l_j \} \leq C. \quad \Box
\]

Proposition 3.2 implies that if we can solve \( 1 / \bar{d}_j / \sum_{j=1}^k C_j \), the problem of minimizing the total completion time of jobs \( J_1, \ldots, J_k \) subject to individual job deadlines, then we can solve \( 1 / \min \text{ delays}, k \text{ 1-chains} / \sum C_j \) by varying \( C \) from \( C_*(MM) \) to \( C_*(SPT) \) (or from \( C_*(SPT) \) down to \( C_*(MM) \)).

We now present an algorithm for \( 1 / \bar{d}_j / \sum_{j=1}^k C_j \) first proposed by Smith [23].

**Proposition 3.3** The following algorithm solves the \( 1 / \bar{d}_j / \sum_{j=1}^k C_j \) problem.

\[
1 / \bar{d}_j / \sum_{j=1}^k C_j \text{ Algorithm}
\]

**Step 1:** Number jobs \( J_1, \ldots, J_k \) such that \( p_1 \geq \cdots \geq p_k \). Sort jobs \( J_1, \ldots, J_k \) in order of nonincreasing deadline.

**Step 2:** \( U \leftarrow \{ J_1, \ldots, J_k \}; T \leftarrow \sum_{j=1}^k p_j. \)

**Step 3:** \( V \leftarrow \{ J_j \in U : \bar{d}_j \geq T \}. \) If \( V = \emptyset \), STOP: The instance is INFEASIBLE.
Step 4: $J_m \leftarrow \text{argmax}\{p_j : J_j \in V\}$. Break ties in favor of the job with the largest deadline.

Step 5: $C_m(\sigma) \leftarrow T (\sigma(m) \leftarrow T - p_m); U \leftarrow U \setminus \{J_m\}; T \leftarrow T - p_m$.

Step 6: If $T > 0$, then go to Step 3. Otherwise, STOP: Schedule $\sigma$ is OPTIMAL.

Proof: The $1 / \bar{d}_j / \sum_{j=1}^{k} C_j$ algorithm terminates with either $V = \emptyset$ or $T = 0$. In the former case, $\bar{d}_j \leq T = \sum_{J_j \in U} p_j$ for all $J_j \in U$. Hence, in any schedule, the job in $U$ scheduled latest exceeds its deadline and the instance is infeasible. In the latter case, the output schedule is feasible since, at each iteration, we scheduled next to last a job which, when so scheduled, met its deadline.

To complete the proof, we need to show that the schedule produced by the algorithm is optimal. Let $\sigma$ be the schedule produced by the algorithm and let $J_{e_1} \rightarrow \cdots \rightarrow J_{e_k}$ be the corresponding sequence. Let $\sigma^*$ be any optimal schedule for $1 / \bar{d}_j / \sum_{j=1}^{k} C_j$ and let $J_{e_1} \rightarrow \cdots \rightarrow J_{e_k}$ be the corresponding sequence. Define $b = \text{argmax}_{j=1,\ldots,k}\{j : J_j \neq J_{e_j}^*\}$. Assume such a $b$ exists since, if not, then $\sigma = \sigma^*$ and we are done. By definition of $b$, $\sigma^*(e_b) < \sigma^*(e_b^*)$ (i.e., job $J_{e_b}$ is scheduled earlier in schedule $\sigma^*$ than is job $J_{e_b}^*$). We now prove, through a series of three claims, that the schedule $\sigma^*$ obtained by interchanging jobs $J_{e_b}$ and $J_{e_b}^*$ in schedule $\sigma^*$ (see Figure 3.1) meets the individual job deadlines and has total completion time no greater than the total completion time of schedule $\sigma^*$. 

Figure 3.1: Obtaining schedule $\bar{\sigma}$ from schedule $\sigma$. 

43
Proof: By definition of $b$, job $J_{e_b}$ was not scheduled in schedule $\sigma$ prior to the iteration in which job $J_{e_b}$ was scheduled. Now, $C_{e_b}(\sigma)$ equals $T$ in the iteration in which job $J_{e_b}$ was scheduled in schedule $\sigma$. Since $C_{e_b}(\sigma) = C_{e_b}^{*}(\sigma^{*})$ and $C_{e_b}^{*}(\sigma^{*}) \leq \bar{d}_{e_b}$, then $\bar{d}_{e_b}$ is greater than or equal to $T$ in the iteration in which job $J_{e_b}$ was scheduled in schedule $\sigma$. Thus, both jobs $J_{e_b}$ and $J_{e_b}^{*}$ must have been in $V$ in the iteration in which job $J_{e_b}$ was scheduled in schedule $\sigma$. Job $J_{e_b}$ was selected over job $J_{e_b}^{*}$, which implies $p_{e_b} \geq p_{e_b}^{*}$. □

Claim 3.3.2 Schedule $\sigma^{*}$ is a feasible schedule.

Proof: We consider each deadline constraint in turn.

1. $C_{e_b}(\sigma^{*}) = C_{e_b}^{*}(\sigma^{*}) = C_{e_b}(\sigma) \leq \bar{d}_{e_b}$.

2. $C_{e_b}^{*}(\sigma^{*}) < C_{e_b}^{*}(\sigma^{*}) \leq \bar{d}_{e_b}$.

3. The completion time of each job scheduled either before job $J_{e_b}$ or after job $J_{e_b}^{*}$ in schedule $\sigma^{*}$ is unchanged from schedule $\sigma^{*}$ to schedule $\bar{\sigma}^{*}$. Hence, these "initial" and "terminal" jobs meet their deadlines in schedule $\bar{\sigma}^{*}$.

4. For each job $J_{e}$, scheduled between jobs $J_{e_b}$ and $J_{e_b}^{*}$ in schedule $\sigma^{*}$,

$$C_{e}(\sigma^{*}) = C_{e}(\sigma^{*}) - p_{e_b} + p_{e_b}^{*}.$$ 

Since, by Claim 3.3.1, $-p_{e_b} + p_{e_b}^{*} \leq 0$, then $C_{e}(\sigma^{*}) \leq C_{e}(\sigma^{*})$. Thus, these "middle" jobs meet their deadlines in schedule $\bar{\sigma}^{*}$. □

44
Claim 3.3.3 $\sum_{j=1}^{k} C_j(\sigma^*) \leq \sum_{j=1}^{k} C_j(\sigma^*)$.

Proof: We consider each completion time in turn.

1. $C_{e_b}(\sigma^*) + C_{e_b}(\sigma^*) = C_{e_b}(\sigma^*) + C_{e_b}(\sigma^*) - [p_{e_b} - p_{e_b}]^+ \leq C_{e_b}(\sigma^*) + C_{e_b}(\sigma^*)$.

2. The completion time of each job scheduled either before job $J_{e_b}$ or after job $J_{e_t}$ in schedule $\sigma^*$ is unchanged from schedule $\sigma^*$ to schedule $\sigma^*$.

3. The completion time of each job scheduled between jobs $J_{e_b}$ and $J_{e_t}$ in schedule $\sigma^*$ is reduced by $p_{e_b} - p_{e_t} \geq 0$ from schedule $\sigma^*$ to schedule $\sigma^*$.

Claims 3.3.2 and 3.3.3 imply that $\sigma^*$ is an optimal schedule. Starting with optimal schedule $\sigma^*$, we can repeat the process of identifying the largest index, if any, in which the sequences corresponding to schedules $\sigma$ and $\sigma^*$ differ. We can then interchange a pair of jobs in schedule $\sigma^*$ to obtain a new optimal schedule. Continuing in this manner, we will eventually obtain an optimal schedule that does not differ from schedule $\sigma$. $\square$

We now show that the $1 / \bar{d}_j / \sum_{j=1}^{k} C_j$ algorithm can be implemented in time $O(k \lg k)$ using a specialized data structure known as a 2-3 tree. A 2-3 tree is a tree in which every vertex which is not a leaf has 2 or 3 children, and all leaves have the same depth [1]. To our knowledge, the proof of the following widely cited complexity result (see [13] and [25] for example) appears nowhere else.

Proposition 3.4 The $1 / \bar{d}_j / \sum_{j=1}^{k} C_j$ algorithm requires time $O(k \lg k)$.

Proof: Steps 1 and 2 require time $O(k \lg k)$ and $O(k)$, respectively. Steps 3-6 are repeated $k$ times. Steps 5 and 6 require constant time per iteration. If $V$ is represented by a 2-3 tree by assigning the jobs to the leaves of the tree in increasing number order from left to right, then Step 4, which consists of identifying the longest (i.e., lowest numbered) job in $V$ (and deleting that job from $V$) requires time $O(\lg k)$.
per iteration (see [1]). Step 3 consists of identifying the jobs in $V$ and inserting each job in $V$ into the 2-3 tree. Observe that each job is inserted into $V$ only once and remains in $V$ until the job is scheduled. Hence, inserting jobs into the 2-3 tree requires time $O(k \lg k)$ over all iterations (reference [1]). Notice also that jobs enter $V$ in order of nonincreasing deadline. The task of identifying the jobs in $V$ can be accomplished in linear time over all iterations using a pointer together with the list of jobs sorted according to deadline from Step 1. Therefore, the $1/\sum_{j=1}^k C_j$ algorithm can be implemented in time $O(k \lg k)$. □

In Step 5 of the $1/\sum_{j=1}^k C_j$ algorithm, let $\text{slack}[m] = T - \bar{d}_m$. By definition of $V$ in Step 3, $\text{slack}[j] \geq 0$ for $j = 1, \ldots, k$. Define $s = \min_{j=1,\ldots,k} \text{slack}[j]$. The following proposition limits the number of completion times for job $*$ which must be considered in solving the $1/\min C_j$ problem.

**Proposition 3.5** The schedule that solves $1/\bar{d}_j = \bar{C} - l_j / \sum_{j=1}^k C_j$ also solves $1/\bar{d}_j = C - l_j / \sum_{j=1}^k C_j$ for all $C \in \{\bar{C} - s, \bar{C} - s + 1, \ldots, \bar{C}\}$.

**Proof:** For all $C$ in the given interval, the order of job selection is unaffected by changes in the individual job deadlines. □

As a result of Proposition 3.5, the next completion time for job $*$ to consider after $C$ is $\bar{C} - s - 1$.

We now present the main result of this subsection, a polynomial algorithm for $1/\min C_j$.

**Proposition 3.6** The following algorithm solves the $1/\min C_j$ problem.

**Algorithm**

**Initialization:** Compute $C_*^{MM}$ and $C_*^{SPT}$. If $C_*^{MM} = C_*^{SPT}$, then STOP: The SPT schedule is OPTIMAL. Otherwise, $\bar{C} \leftarrow C_*^{SPT}$; Incumbent $\leftarrow$ nil. Compute deadlines $\bar{d}_j = \bar{C} - l_j$ for $j = 1, \ldots, k$. 46
Step 1: Solve the $1/\bar{d}_j/\sum C_j$ problem on jobs $J_1, \ldots, J_k$. Compute $s = \min_{j=1, \ldots, k} slack[j]$ and let $\sigma(*) = C - s$.

Step 2: If schedule $\sigma$ has total completion time less than the total completion time of the incumbent solution, then replace the incumbent solution with schedule $\sigma$.

Step 3: $C \leftarrow C - s - 1$. If $C < C^*_{MM}$, then STOP: The current incumbent solution is OPTIMAL. Otherwise, $\bar{d}_j \leftarrow \bar{d}_j - s - 1$ for $j = 1, \ldots, k$. Go to Step 1.

Proof: The correctness of the $1/\min \text{ delays}, k \text{ 1-chains}/\sum C_j$ algorithm follows immediately from Propositions 3.1, 3.3, and 3.5. We should point out that each $1/\bar{d}_j/\sum C_j$ instance encountered is feasible since schedule MM meets the deadlines $\bar{d}_j = C^*_{MM} - l_j$ for $j = 1, \ldots, k$, which implies schedule MM meets the deadlines $\bar{d}_j = C - l_j$ for $j = 1, \ldots, k$ and for any $C \geq C^*_{MM}$. $\square$

Before analyzing the complexity of the $1/\min \text{ delays}, k \text{ 1-chains}/\sum C_j$ algorithm, let us consider an example. For the instance shown in Figure 3.2,

$$C_*(MM) = 111,$$

$$\sum_{j=1}^{6} C_j(MM) = 151,$$

$$\sum C_j(MM) = 262,$$

$$C_*(SPT) = 127,$$

$$\sum_{j=1}^{6} C_j(SPT) = 123,$$

and

$$\sum C_j(SPT) = 250.$$

One optimal schedule, $\sigma^*$, corresponds to the sequence $J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_6 \rightarrow J_5 \rightarrow \ast$. For schedule $\sigma^*$,
Figure 3.2: Instance of $1 / \text{min delays, } k\text{ 1-chains } / \sum C_j$.

$$C_*(\sigma^*) = 116,$$

$$\sum_{j=1}^{6} C_j(\sigma^*) = 126,$$

and

$$\sum C_j(\sigma^*) = 242.$$

This example illustrates the tradeoff between minimizing $\sum_{j=1}^{k} C_j$ and minimizing $C_*$ in solving $1 / \text{min delays, } k\text{ 1-chains } / \sum C_j$. An optimal schedule is not necessarily a schedule that minimizes either $\sum_{j=1}^{k} C_j$ or $C_*$, but instead is a schedule that balances the two objectives.

A schedule $\sigma$ is Pareto optimal with respect to the objective functions $\sum_{j=1}^{k} C_j$ and $C_*$ if there exists no feasible schedule $\pi$ with

$$\sum_{j=1}^{k} C_j(\pi) \leq \sum_{j=1}^{k} C_j(\sigma) \text{ and } C_*(\pi) \leq C_*(\sigma),$$

where at least one of these two inequalities is strict. Clearly, the optimal schedule for $1 / \text{min delays, } k\text{ 1-chains } / \sum C_j$ is Pareto optimal. The remainder of this subsection, which follows closely the exposition in Hoogeveen and van de Velde [13], consists of showing that the complexity of the $1 / \text{min delays, } k\text{ 1-chains } / \sum C_j$ algorithm is $O(k^3 \lg k)$ by first proving that the schedules produced by the algorithm are Pareto optimal and then showing that the number of Pareto optimal schedules produced by the algorithm is $O(k^2)$. 

48
**Proposition 3.7** The \( 1/\min \) delays, \( k \) 1-chains \( \sum C_j \) algorithm produces Pareto optimal schedules with respect to \( \sum_{j=1}^{k} C_j \) and \( C_* \).

**Proof:** Let \( \sigma \) be any schedule produced by the algorithm. We must show there exists no feasible schedule \( \pi \) with

1. \( \sum_{j=1}^{k} C_j(\pi) = \sum_{j=1}^{k} C_j(\sigma) \) and \( C_*(\pi) < C_*(\sigma) \),

2. \( \sum_{j=1}^{k} C_j(\pi) < \sum_{j=1}^{k} C_j(\sigma) \) and \( C_*(\pi) = C_*(\sigma) \), or

3. \( \sum_{j=1}^{k} C_j(\pi) < \sum_{j=1}^{k} C_j(\sigma) \) and \( C_*(\pi) < C_*(\sigma) \).

Schedule \( \sigma \) (sans \( \sigma(*) \)) solves \( 1/\overline{d}_j = \overline{C} - l_j / \sum_{j=1}^{k} C_j \) for some \( \overline{C} = C_*(\sigma) + s \). By Proposition 3.5, \( \sigma \) (sans \( \sigma(*) \)) also solves \( 1/\overline{d}_j = C_*(\sigma) - l_j / \sum_{j=1}^{k} C_j \). Now, by Proposition 3.2, a schedule satisfies \( C_* \leq C_*(\sigma) \) if and only if that schedule meets the deadlines \( \overline{d}_j = C_*(\sigma) - l_j \) for \( j = 1, \ldots, k \). Hence, there can exist no feasible schedule \( \pi \) with

\[
\sum_{j=1}^{k} C_j(\pi) < \sum_{j=1}^{k} C_j(\sigma) \quad \text{and} \quad C_*(\pi) \leq C_*(\sigma),
\]

which establishes points 2 and 3 above.

Suppose there exists a schedule with

\[
\sum_{j=1}^{k} C_j = \sum_{j=1}^{k} C_j(\sigma) \quad \text{and} \quad C_* < C_*(\sigma).
\]

Among all such schedules, let \( \sigma^* \) be one with smallest \( C_* \). Let \( J_{e_1} \rightarrow \ldots \rightarrow J_{e_*} \rightarrow * \) and \( J_{e^*_1} \rightarrow \ldots J_{e^*_k} \rightarrow * \) be the sequences corresponding to schedules \( \sigma \) and \( \sigma^* \), respectively. Define \( b = \arg \max_{j=1,\ldots,k} \{ j : J_j \neq J_{e^*_j} \} \). As in Claim 3.3.1, \( p_{e_b} \geq p_{e^*_b} \).

Suppose \( p_{e_b} > p_{e^*_b} \). Then, as in Claims 3.3.2 and 3.3.3, we can show that the schedule obtained by interchanging jobs \( J_{e_b} \) and \( J_{e^*_b} \) in schedule \( \sigma^* \) meets the deadlines \( \overline{d}_j = C_*(\sigma) - l_j \) for each \( j = 1, \ldots, k \) and has \( \sum_{j=1}^{k} C_j \) less than \( \sum_{j=1}^{k} C_j(\sigma^*) = \sum_{j=1}^{k} C_j(\sigma) \), a contradiction of the fact that schedule \( \sigma \) solves \( 1/\overline{d}_j = C_*(\sigma) - l_j / \sum_{j=1}^{k} C_j \). Thus, \( p_{e_b} = p_{e^*_b} \). 

49
By choice of job $J_{eb}$ over job $J_{e*}$ in the algorithm,

$$d_{eb} \geq d_{e*} \Rightarrow C - l_{eb} \geq C - l_{e*} \Rightarrow l_{eb} \leq l_{e*}.$$  

Now, $p_{eb} = p_{e*}$ and $l_{eb} \leq l_{e*}$ imply that the schedule obtained by interchanging jobs $J_{eb}$ and $J_{e*}$ in schedule $\sigma^*$ has $\sum_{j=1}^{k} C_j$ equal to $\sum_{j=1}^{k} C_j(\sigma^*)$ and has $C_*$ less than or equal to $C_*(\sigma^*)$. By definition of schedule $\sigma^*$, this new schedule has $C_*$ equal to $C_*(\sigma^*)$.

Repeating this argument, we see that schedule $\sigma^*$ can be transformed into schedule $\sigma$ without increasing the total completion time for jobs $J_1, \ldots, J_k$, a contradiction of our assumption that $C_*(\sigma^*) < C_*(\sigma)$. Hence, there can exist no feasible schedule $\pi$ with

$$\sum_{j=1}^{k} C_j(\pi) = \sum_{j=1}^{k} C_j(\sigma) \text{ and } C_*(\pi) < C_*(\sigma),$$

which establishes point 1 and completes the proof. □

For each feasible schedule $\sigma$ and for each pair of jobs $J_i \neq J_j$ from $\{J_1, \ldots, J_k\}$, let the indicator function $\delta_{ij}(\sigma)$ be defined by

$$\delta_{ij}(\sigma) = \begin{cases} 
1, & \text{if } C_i(\sigma) < C_j(\sigma) \text{ and } p_i > p_j \\
0, & \text{otherwise.} 
\end{cases}$$

We refer to the interchange of jobs $J_i$ and $J_j$ in schedule $\sigma$ as a positive interchange if the total completion time for jobs $J_1, \ldots, J_k$ of the resultant schedule is less than $\sum_{j=1}^{k} C_j(\sigma)$. A positive interchange is synonymous with $\delta_{ij}(\sigma) = 1$. The interchange of jobs $J_i$ and $J_j$ in schedule $\sigma$ is neutral if the total completion time for jobs $J_1, \ldots, J_k$ of the resultant schedule equals $\sum_{j=1}^{k} C_j(\sigma)$, which occurs if and only if $p_i = p_j$.

For each feasible schedule $\sigma$, let $\Delta(\sigma) = \sum_{i \neq j} \delta_{ij}(\sigma)$. Note that $0 \leq \Delta(\sigma) \leq \frac{1}{2} k(k - 1)$ for all feasible schedules $\sigma$. The following lemma relates the $\Delta$ functions of two feasible schedules, one of which can be obtained from the other via a positive interchange.
Table 3.1: Possible values for $\delta_{ij}(\sigma)$, $\delta_{ij}(\sigma)$, $\delta_{ii}(\pi)$, and $\delta_{ii}(\pi)$.

<table>
<thead>
<tr>
<th>Relationship of $p_i$ and $p_j$ to $p_i$</th>
<th>$\delta_{il}(\sigma)$</th>
<th>$\delta_{lj}(\sigma)$</th>
<th>$\delta_{ij}(\pi)$</th>
<th>$\delta_{ii}(\pi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i &lt; p_j &lt; p_i$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$pi = p_j &lt; p_i$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$p_j &lt; p_i &lt; p_i$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$p_j &lt; p_i = p_i$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$p_j &lt; p_i &lt; p_i$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Lemma 3.8 If schedule $\pi$ can be obtained from schedule $\sigma$ through a positive interchange, then $\Delta(\pi) < \Delta(\sigma)$.

Proof: Suppose schedule $\pi$ can be obtained from schedule $\sigma$ by interchanging jobs $J_i$ and $J_j$, where $p_i > p_j$. The only $\delta$'s which are or might be affected by the interchange are $\delta_{ij}$, $\delta_{li}$, $\delta_{ji}$, and $\delta_{ij}$, where job $J_i$ is any job scheduled between jobs $J_i$ and $J_j$ in schedule $\sigma$.

The change in the $\Delta$ function from schedule $\sigma$ to schedule $\pi$ is given by

$$\delta_{ij}(\sigma) - \delta_{ij}(\pi) + \sum_{J_i}[(\delta_{il}(\sigma) + \delta_{lj}(\sigma)) - (\delta_{il}(\pi) + \delta_{lj}(\pi))].$$

Clearly, $\delta_{ij}(\sigma) = 1$ and $\delta_{ij}(\pi) = 0$.

Table 3.1 shows that for all jobs $J_i$,

$$\delta_{il}(\sigma) + \delta_{lj}(\sigma) \geq \delta_{ij}(\pi) + \delta_{ii}(\pi),$$

which completes the proof. $\square$

The next proposition relates the $\Delta$ functions of two schedules produced by the $1 / \min$ delays, $k$ 1-chains / $\Sigma C_j$ algorithm.

Proposition 3.9 If $\sigma$ and $\pi$ are any two schedules produced by the $1 / \min$ delays, $k$ 1-chains / $\Sigma C_j$ algorithm, where, without loss of generality, $\sum_{j=1}^{k} C_j(\sigma) < \sum_{j=1}^{k} C_j(\pi)$, then $\Delta(\sigma) < \Delta(\pi)$.
Proof: We show that schedule $\sigma$ can be obtained from schedule $\pi$ using only positive and neutral interchanges. Let $J_{e_1} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow \ast$ and $J'_{e_1} \rightarrow \cdots \rightarrow J'_{e_k} \rightarrow \ast$ be the sequences corresponding to schedules $\sigma$ and $\pi$, respectively. Define $b = \arg\max_{j=1,...,k} \{j : J_j \not= J'_j\}$.

By definition of $b$, job $J'_{e_b}$ was not scheduled in schedule $\sigma$ prior to the iteration in which job $J_{e_b}$ was scheduled. Because $\sigma$ and $\pi$ are Pareto optimal, then $\sum_{j=1}^{k} C_j(\sigma) < \sum_{j=1}^{k} C_j(\pi)$ implies $C_*(\sigma) > C_*(\pi)$. Now, $T$ in the iteration in which job $J'_{e_b}$ was scheduled in schedule $\pi$ equals $T$ in the iteration in which job $J_{e_b}$ was scheduled in schedule $\sigma$. Since $C_*(\pi) - l_{e_b}$ is greater than or equal to $T$ in the iteration in which job $J'_{e_b}$ was scheduled in schedule $\pi$ and $C_*(\pi) < C_*(\sigma)$, then $C_*(\sigma) - l_{e_b}$ is greater than or equal to $T$ in the iteration in which job $J_{e_b}$ was scheduled in schedule $\sigma$. Thus, both jobs $J_{e_b}$ and $J'_{e_b}$ must have been in $V$ in the iteration in which job $J_{e_b}$ was scheduled in schedule $\sigma$. By choice of job $J_{e_b}$ over job $J'_{e_b}$ in the algorithm, $p_{e_b} \geq p'_{e_b}$. Thus, the interchange of jobs $J_{e_b}$ and $J'_{e_b}$ in schedule $\pi$ is either positive or neutral.

Repeating this process, we will eventually obtain schedule $\sigma$. Since $\sum_{j=1}^{k} C_j(\sigma) < \sum_{j=1}^{k} C_j(\pi)$, then at least one of the positive or neutral interchanges must have been positive. The result now follows from Lemma 3.8. □

We are finally prepared to establish the running time of the $1 / \min$ delays, $k$ 1-chains / $\sum C_j$ algorithm.

**Proposition 3.10** The $1 / \min$ delays, $k$ 1-chains / $\sum C_j$ problem is solvable in time $O(k^3 \lg k)$.

**Proof:** Proposition 3.7 together with the fact that $0 \leq \Delta(\sigma) \leq \frac{1}{2}k(k - 1)$ for all feasible schedules $\sigma$ implies the number of schedules produced by the algorithm is $O(k^2)$. The $1 / \min$ delays, $k$ 1-chains / $\sum C_j$ algorithm requires time $O(k^3 \lg k)$ overall, since, by Proposition 3.4, each iteration requires time $O(k \lg k)$. □
In [13], Hoogeveen and van de Velde actually considered a more general problem than 1 / min delays, k 1-chains / \(\sum C_j\). Let \(f_j(C_j)\) denote the cost of completing job \(J_j\) at time \(C_j\) for \(j = 1, \ldots, k\). Assume \(f_j\) is nondecreasing in \(C_j\) for \(j = 1, \ldots, k\). Define \(f_{\max} = \max_{j=1,\ldots,k} f_j(C_j)\) and \(p_{\max} = \max_{j=1,\ldots,k} p_j\). Hoogeveen and van de Velde proved that 1 / / \(F(\sum_{j=1}^k C_j, f_{\max})\) is solvable in time \(O(k^3 \min\{k, \lg k + \lg p_{\max}\})\) for any function \(F\) that is nondecreasing in \(\sum_{j=1}^k C_j\) and \(f_{\max}\).

Curiously enough, 1 / min delays, k 1-chains / \(\sum C_j\), a precedence constrained scheduling problem, is a special case of 1 / / \(F(\sum_{j=1}^k C_j, f_{\max})\), a problem not involving precedence constraints. Let \(f_j(C_j) = C_j + l_j\) for \(j = 1, \ldots, k\) and let \(F(\sum_{j=1}^k C_j, f_{\max}) = \sum_{j=1}^k C_j + f_{\max}\). Then,

\[
F(\sum_{j=1}^k C_j, f_{\max}) = \sum_{j=1}^k C_j + \max_{j=1,\ldots,k} \{C_j + l_j\} = \sum_{j=1}^k C_j + C^*_j.
\]

### 3.1.2 1 / min delays, k 1-chains / \(\sum w_jC_j\)

In the previous subsection, we showed that the total completion time problem 1 / min delays, k 1-chains / \(\sum C_j\) is solvable in polynomial time. In this subsection, we show that the closely related total weighted completion time problem is NP-hard. The decision problem version of 1 / min delays, k 1-chains / \(\sum w_jC_j\), which we refer to as 1 / min delays, k 1-chains / \(\sum w_jC_j \leq Y\), is defined as follows.

**INSTANCE:** Job set \(J = \{J_1, \ldots, J_k\} \cup \{\star\}\), processing requirement \(p_j \in \mathbb{Z}_0^+\) and weight \(w_j \in \mathbb{Z}_0^+\ \forall J_j \in J\), precedence relation \(P\) on \(J\) of the form \(P = \{< J_j, \star >, j = 1, \ldots, k\}\), nonnegative integer minimum delays \(l_j\) for \(j = 1, \ldots, k\), and a positive integer \(Y\).

**QUESTION:** Is there a one-machine schedule for \(J\) (i.e., a function \(\sigma : J \rightarrow \mathbb{Z}_0^+\), with \(\sigma(j) > \sigma(j')\) implying \(\sigma(j) \geq \sigma(j') + p_{j'}\)) that satisfies the minimum delay precedence constraints (i.e., \(\sigma(\star) - C_j(\sigma) \geq l_j\) for \(j = 1, \ldots, k\), where \(C_j(\sigma) = \sigma(j) + p_j \ \forall J_j \in J\)) and such that the sum, taken over all \(J_j \in J\), of \(w_jC_j(\sigma)\) is \(Y\) or less?
We now prove that the decision problem version of $1 / \min \text{delays, } k 1\text{-chains} / \sum w_j C_j$ is NP-complete.

**Proposition 3.11** The $1 / \min \text{delays, } k 1\text{-chains} / \sum w_j C_j \leq Y$ problem is NP-complete.

**Proof:** Given any sequence of the jobs in $J$, we can, in polynomial time, verify that the associated schedule satisfies the minimum delay precedence constraints and has total weighted completion time $Y$ or less. Thus, $1 / \min \text{delays, } k 1\text{-chains} / \sum w_j C_j \leq Y$ is in NP.

Let index set $A = \{1, \ldots, a\}$ and size $s(j) \in \mathbb{Z}_0^+$ for all $j \in A$ be any instance of PARTITION (see page 16). We construct a corresponding instance of $1 / \min \text{delays, } k 1\text{-chains} / \sum w_j C_j \leq Y$ in polynomial time as follows:

- $k = a + 1$;
- $p_j = w_j = s(j)$, $l_j = 0$, $j = 1, \ldots, k - 1$;
- $p_k = 1$, $w_k = \frac{1}{2}$, $l_k = \frac{1}{2} \sum_{j \in A} s(j)$;
- $p_\ast = 0$, $w_\ast = 1$;
- $Y = \sum_{1 \leq j \leq k \leq a} s(j) s(k) + \frac{7}{4} \sum_{j \in A} s(j) + \frac{3}{2}$.

With respect to jobs $J_1, \ldots, J_{k-1}$, any nonpreemptive schedule without machine idle time is optimal and has value $\sum_{1 \leq j \leq k \leq a} s(j) s(k)$. Inserting the unit-time job $J_k$ in a schedule for jobs $J_1, \ldots, J_{k-1}$ increases the contribution of jobs $J_1, \ldots, J_{k-1}$ by the sum of the processing requirements of jobs from $\{J_1, \ldots, J_{k-1}\}$ completed after job $J_k$.

Suppose there exists a subset $A' \subseteq A$ such that $\sum_{j \in A'} s(j) = \sum_{j \in A \setminus A'} s(j) = \frac{1}{2} \sum_{j \in A} s(j)$. Let $S = \{J_j \in \{J_1, \ldots, J_{k-1}\} : j \in A'\}$ and $T = \{J_1, \ldots, J_{k-1}\} \setminus S$. The schedule illustrated in Figure 3.3 satisfies the minimum delay precedence constraints and has total weighted completion time equal to $Y$. 

54
<table>
<thead>
<tr>
<th>Jobs in $S$ in any order</th>
<th>$J_k$</th>
<th>Jobs in $T$ in any order</th>
<th>$*$</th>
</tr>
</thead>
</table>

Figure 3.3: Feasible schedule for constructed instance of $1 / \min$ delays, $k$ 1-chains / $\sum w_j C_j \leq Y$.

On the other hand, suppose there exists no subset $A' \subseteq A$ such that $\sum_{j \in A'} s(j) = \sum_{j \in A \setminus A'} s(j)$. Then, in any schedule that satisfies the minimum delay precedence constraints, the sum of the processing requirements of jobs from $\{J_1, \ldots, J_{k-1}\}$ completed after job $J_k$ is either strictly less than or strictly greater than $\frac{1}{2} \sum_{j \in A} s(j)$.

Suppose this sum of processing requirements is strictly less than $\frac{1}{2} \sum_{j \in A} s(j)$. Figure 3.4 illustrates an arbitrary schedule that satisfies the minimum delay precedence constraints and for which the sum of the processing requirements of jobs from $\{J_1, \ldots, J_{k-1}\}$ completed after job $J_k$ equals $\frac{1}{2} \sum_{j \in A} s(j) - \Delta$ for some $\Delta > 0$. The contribution of jobs $J_1, \ldots, J_{k-1}$ to the value of this schedule is given by

$$\sum_{1 \leq j \leq k \leq a} s(j)s(k) + \frac{1}{2} \sum_{j \in A} s(j) - \Delta.$$  

The contributions of jobs $J_k$ and $*$ are given by

$$\frac{1}{2}\left(\frac{1}{2} \sum_{j \in A} s(j) + \Delta + 1\right)$$

and

$$\frac{1}{2} \sum_{j \in A} s(j) + \Delta + p_k + l_k = \sum_{j \in A} s(j) + \Delta + 1,$$

respectively. Summing these contributions, we see that this schedule has total weighted completion time $Y + \frac{1}{2}\Delta > Y$.  

55
Figure 3.4: Schedule with sum of processing times of jobs from \( \{J_1, \ldots, J_{k-1}\} \) completed after job \( J_k \) less than \( \frac{1}{2} \sum_{j \in A} s(j) \).

Now, suppose the sum of processing requirements of jobs from \( \{J_1, \ldots, J_{k-1}\} \) completed after job \( J_k \) is strictly greater than \( \frac{1}{2} \sum_{j \in A} s(j) \). Figure 3.5 shows an arbitrary schedule that satisfies the minimum delay precedence constraints and for which the sum of the processing requirements of jobs from \( \{J_1, \ldots, J_{k-1}\} \) completed after job \( J_k \) equals \( \frac{1}{2} \sum_{j \in A} s(j) + \Delta \) for some \( \Delta > 0 \). The contribution of jobs \( J_1, \ldots, J_{k-1} \) to the value of this schedule is given by

\[
\sum_{1 \leq j \leq k < a} s(j)s(k) + \frac{1}{2} \sum_{j \in A} s(j) + \Delta.
\]

The contributions of jobs \( J_k \) and * are given by

\[
\frac{1}{2}(\frac{1}{2} \sum_{j \in A} s(j) - \Delta + 1)
\]

and

\[
\sum_{j \in A} s(j) + p_k = \sum_{j \in A} s(j) + 1,
\]

respectively. Summing these contributions, we see that this schedule also has total weighted completion time equal to \( Y + \frac{1}{2} \Delta > Y \). Thus, there exists a schedule that satisfies the minimum delay precedence constraints and has total weighted completion time \( Y \) or less if and only if there exists a partition of \( \{J_1, \ldots, J_{k-1}\} \)
Figure 3.5: Schedule with sum of processing times of jobs from \( \{J_1, \ldots, J_{k-1}\} \) completed after job \( J_k \) greater than \( \frac{1}{2} \sum_{j \in A} s(j) \).

into two disjoint subsets such that the sum of the processing requirements of the jobs in each subset equals \( \frac{1}{2} \sum_{j \in A} s(j) \).

Proposition 3.11 does not preclude a pseudo-polynomial time algorithm for 1/min delays, \( k \)-1-chains / \( \sum w_j C_j \). Whether or not such an algorithm exists is an open question.

3.2 Max Delays Problems

Removing idle time from a schedule that satisfies the maximum delay precedence constraints results in a feasible schedule with smaller total completion time and no larger total weighted completion time. Thus, 1/max delays, \( k \)-1-chains / \( \sum C_j \) (\( \sum w_j C_j \)) is the problem of finding, among all schedules without machine idle time that satisfy the maximum delay precedence constraints, a schedule that has minimum total (weighted) completion time.

3.2.1 1/max delays, \( k \)-1-chains / \( \sum C_j \)

In Subsection 3.1.1, we showed that 1/min delays, \( k \)-1-chains / \( \sum C_j \) can be solved in time \( O(k^3 \lg k) \). In this subsection, we show that the corresponding max delays problem is NP-hard in the strong sense. The decision problem version of 1/max delays, \( k \)-1-chains / \( \sum C_j \), which we refer to as 1/max delays, \( k \)-1-chains / \( \sum C_j \) \( \leq Y \), is defined as follows.
INSTANCE: Job set \( J = \{J_1, \ldots, J_k\} \) \( \cup \{\star\} \), processing requirement \( p_j \in \mathbb{Z}_{\geq 0}^+ \) \( \forall J_j \in J \), precedence relation \( P \) on \( J \) of the form \( P = \{< J_j, \star >, \ j = 1, \ldots, k\} \), maximum delays \( u_j \) for \( j = 1, \ldots, k \), where each maximum delay is either infinite or a nonnegative integer, and a positive integer \( Y \).

QUESTION: Is there a one-machine schedule for \( J \) (i.e., a function \( \sigma : J \rightarrow \mathbb{Z}_0^+ \), with \( \sigma(j) > \sigma(j') \) implying \( \sigma(j) \geq \sigma(j') + p_j \) that satisfies the maximum delay precedence constraints (i.e., \( \sigma(\star) - C_j(\sigma) \leq u_j \), where \( C_j(\sigma) = \sigma(j) + p_j \) \( \forall j = 1, \ldots, k \)) and such that the sum, taken over all \( J_j \in J \), of \( C_j(\sigma) \) is \( Y \) or less?

We now show that \( 1 / \max \text{ delays, } k \text{-chains} / \sum C_j \leq Y \) is strongly NP-complete. The proof is adapted from and follows closely the complexity proof for \( F_2 / / \sum C_j \), the problem of minimizing total completion time in a two-machine flowshop, in Garey, Johnson, and Sethi [11].

**Proposition 3.12** The \( 1 / \max \text{ delays, } k \text{-chains} / \sum C_j \leq Y \) problem is NP-complete in the strong sense.

**Proof:** Given any sequence of the jobs in \( J \), we can, in polynomial time, verify that the associated schedule without machine idle time satisfies the maximum delay precedence constraints and has total completion time \( Y \) or less. Thus, \( 1 / \max \text{ delays, } k \text{-chains} / \sum C_j \leq Y \) is in NP.

The problem we use for the reduction is 3-PARTITION (see page 23). We start with \( t + 1 \) unit-length jobs. These jobs have associated maximum precedence delays such that, if they are the only jobs to be scheduled, then they can be scheduled in such a manner as to leave \( t \) identical slots. We add other jobs having maximum delays large enough so that the associated maximum delay precedence constraints are always satisfied and exactly fill the slots with these jobs and meet the target total completion time if and only if the 3-PARTITION instance has a solution.

Let \( T = \{1, \ldots, 3t\} \) and positive integers \( a_1, \ldots, a_{3t} \), and \( b \), with \( a_j \in ( \frac{1}{2} b, \frac{1}{2} b ) \) for all \( j \in T \) and \( \sum_{j \in T} a_j = tb \) be any instance of 3-PARTITION. We construct a corresponding instance of \( 1 / \max \text{ delays, } k \text{-chains} / \sum C_j \) as follows:
\[ z = 3tb + 1; \]
\[ v = z + 3tb + tz + \frac{t(t-1)}{2} \cdot z(b + 1); \]
\[ c = zv + b + 1; \]
\[ x = 2(t + 2)c + v; \]
\[ k = t + 1 + v + tz; \]

\[ ps_j = 1, \ u_{s_j} = (t - j)c + xv, \ j = 0, 1, \ldots, t; \]
\[ px_j = x, \ u_{x_j} = tc + xv + 1, \ j = 1, \ldots, v; \]
\[ pv_{i,j} = v, \ u_{v_{i,j}} = tc + xv + 1, \ i = 1, \ldots, t, \ j = 1, \ldots, z - 3; \]
\[ pw_j = v + a_j, \ u_{w_j} = tc + xv + 1, \ j = 1, \ldots, 3t; \]
\[ p* = 0; \]

\[ Y = \hat{S} + \hat{X} + \hat{Z} + tc + xv + 1, \text{ where} \]
\[ \hat{S} = \sum_{j=0}^{t} (jc + 1), \]
\[ \hat{X} = \sum_{j=1}^{t} (tc + 1 + jx), \text{ and} \]
\[ \hat{Z} = 3tb + \sum_{i=0}^{t-1} \left[ \sum_{j=1}^{i} (jv + ic + 1) \right] \]
\[ \quad = 3tb + tz + \frac{t(t-1)}{2} \cdot z(b + 1) + \frac{t(tz+1)}{2} \cdot v. \]

Bearing in mind that \( Y \), the largest number produced under this mapping from 3-PARTITION to \( 1/ \text{max delays}, k \) 1-chains / \( \sum C_j \leq Y \), is bounded by a polynomial in \( t \) and \( b \), one can easily verify that the mapping satisfies the computation time and instance size requirements for a pseudo-polynomial transformation.
We refer to jobs $J_s$, for $j = 0, 1, \ldots, t$ as 'S' jobs. Similarly, we refer to jobs with $X$, $V$, and $W$ subscripts as 'X', 'V', and 'W' jobs, respectively. We refer to the $V$ and $W$ jobs collectively as 'Z' jobs.

Suppose that $T$ can be partitioned into $t$ disjoint subsets $T_1, \ldots, T_t$ such that $\sum_{j \in T_i} a_j = b$ for $i = 1, \ldots, t$. Assume $T_i = \{g(i, 1), g(i, 2), g(i, 3)\}$ for $i = 1, \ldots, t$ and consider the schedule, $\sigma$, illustrated in Figure 3.6, wherein jobs are identified by their subscripts only.

We first show that schedule $\sigma$ satisfies the maximum delay precedence constraints. The sum of the processing requirements of the $Z$ jobs scheduled between jobs $J_s$ and $J_{s+1}$ in schedule $\sigma$ is given by

$$(z - 3)v + (3v + b) = zv + b = c - 1.$$  

for $j = 0, 1, \ldots, t - 1$. Now,

$$\sigma(*) - C_{J_s}(\sigma) = (t - j)c + xv = u_s,$$

for $j = 0, 1, \ldots, t$, so schedule $\sigma$ satisfies the maximum delay precedence constraints corresponding to the S jobs. Any schedule without machine idle time satisfies the maximum delay precedence constraints corresponding to the X and Z jobs since $tc + xv + 1 = p(J)$. Hence, schedule $\sigma$ satisfies the maximum delay precedence constraints.

We now verify that schedule $\sigma$ has total completion time $Z$ or less. One can easily show that $\sum_{j=0}^{t} C_{J_s}(\sigma) = \hat{S}$, $\sum_{j=1}^{u} C_{X_j}(\sigma) = \hat{X}$, and $C_*(\sigma) = tc + xv + 1$. The total completion time of the $Z$ jobs in schedule $\sigma$ is given by

$$Z = \sum_{i=0}^{t-1} \left( \sum_{j=1}^{i} (jv + ic + 1) + 3a_g(i+1,1) + 2a_g(i+1,2) + a_g(i+1,3) \right),$$

which is less than

$$\hat{Z} = \sum_{i=0}^{t-1} \left( \sum_{j=1}^{i} (jv + ic + 1) + 3 \sum_{j=1}^{3t} a_j \right).$$
Therefore, 

\[ \sum C_j(\sigma) = \hat{S} + \hat{X} + Z + tc + xv + 1 < Y \]

and schedule \( \sigma \) is feasible for \( 1 / \text{max delays, } k \text{-chains} / \sum C_j \).

We now show, through a series of six claims, that if there exists a "good" schedule, that is, a schedule that satisfies the maximum delay precedence constraints and has total completion time \( Y \) or less, then there exists a partition of \( T \) into \( t \) disjoint subsets \( T_1, \ldots, T_t \) such that \( \sum_{j \in T_i} a_j = b \) for \( i = 1, \ldots, t \). The first and second claims address the relative ordering of the \( S \) and \( X \) jobs.

**Claim S** If there exists a good schedule, then there exists a good schedule that satisfies 

\[ \sigma(S_0) < \sigma(S_1) < \cdots < \sigma(S_t). \]  

**Proof:** Jobs \( J_{S_0}, J_{S_1}, \ldots, J_{S_t} \) can be interchanged in any good schedule so that job \( J_{S_j} \) precedes job \( J_{S_{j+1}} \) for \( j = 0, 1, \ldots, t - 1 \) without affecting feasibility or value. \( \square \)

**Claim X1** If there exists a good schedule, then there exists a good schedule that satisfies 

\[ \sigma(X_1) < \cdots < \sigma(X_v). \]  

**Proof:** Jobs \( J_{X_1}, \ldots, J_{X_v} \) can be interchanged in any good schedule so that job \( J_{X_j} \) precedes job \( J_{X_{j+1}} \) for \( j = 1, \ldots, v - 1 \) without affecting feasibility or value. \( \square \)
The third claim gives a lower bound for the start time of the X jobs in some good schedule that satisfies (3.1) and (3.2).

Claim X2 If there exists a good schedule that satisfies conditions (3.1) and (3.2), then there exists a good schedule that satisfies these conditions that also satisfies

\[ \sigma(X_j) \geq tc + 1 \text{ for } j = 1, \ldots, v. \]  

Proof: Among all good schedules that satisfy (3.1) and (3.2), let schedule \( \sigma \) be one with minimum total completion time. Suppose that schedule \( \sigma \) does not satisfy (3.3), so that \( \sigma(X_i) < tc + i \) for some \( i \). Since \( P_X > tc + 1 \) for each \( j = 1, \ldots, v \) and since \( \sigma \) satisfies (3.2), then \( i \) must be equal to 1.

We now show that the jobs scheduled after job \( J_{X_1} \) in schedule \( \sigma \) can be reordered without violating the maximum delay precedence constraints. Observe that reordering affects neither the start time of job \( * \) nor the completion time of job \( J_{S_j} \) for any job \( J_{S_j} \) scheduled before job \( J_{X_1} \) in schedule \( \sigma \). Since \( \sigma \) satisfies (3.1), then the sum of the processing requirements of jobs scheduled between jobs \( J_{S_j} \) and \( * \) in the reordered schedule is at most

\[ P(J) - P_{X_1} - \sum_{m=0}^{j} P_{S_m} = tc + xv - x - j \]

for any job \( J_{S_j} \) scheduled after job \( J_{X_1} \) in schedule \( \sigma \). Now,

\[ u_{S_j} - [tc + xv - x - j] = (t - j)c + xv - [tc + xv - x - j] \]
\[ = -jc + x + j \]
\[ = -jc + 2(t + 2)c + v + j \]
\[ = (2t + 4 - j)c + v + j \]
\[ \geq (t + 4)c + v + j \]
\[ > 0. \]

Hence, the reordered schedule satisfies the maximum delay precedence constraints.

Since the jobs scheduled after job \( J_{X_1} \) can be reordered and since \( \sigma \) is a minimum total completion time schedule, then the jobs scheduled after job \( J_{X_1} \) in schedule
\( \sigma \) must be ordered by nondecreasing processing requirement. In particular, jobs \( J_{x_2}, \ldots, J_{x_v} \) must be scheduled last before job \( * \) in schedule \( \sigma \). Since \( \sigma(X_1) \geq p_{x_1} = x \), then
\[
\sum_{j=1}^{v} C_{x_j}(\sigma) \geq X = x + \sum_{j=2}^{v} (tc + 1 + jx) = X - tc - 1.
\]
The start time of job \( * \) in schedule \( \sigma \) is at least \( p(J) = tc + xv + 1 \). Thus
\[
C_{s_1}(\sigma) \geq p(J) - us_1 = jc + 1
\]
for each \( j = 0, 1, \ldots, t - 1 \). Job \( J_{s_1} \) must be scheduled after job \( J_{x_1} \) in schedule \( \sigma \) since, otherwise,
\[
\sigma(X_1) < tc + 1 \Rightarrow \sigma(X_1) \leq tc \Rightarrow C_{s_1}(\sigma) \leq tc \Rightarrow
\]
\[
\sigma(*) - C_{s_1}(\sigma) \geq p(J) - tc = xv + 1 > us_1,
\]
a contradiction of the fact that \( \sigma \) is a good schedule. Hence, \( C_{s_1}(\sigma) \geq p(X_1) = x \)
and
\[
\sum_{j=0}^{t} C_{s_1}(\sigma) \geq \bar{S} = \sum_{j=0}^{t-1} (jc + 1) + x
\]
\[
= \hat{S} - (tc + 1) + x
\]
\[
= \hat{S} + tc + 4c + v - 1
\]
\[
> \hat{S} + tc + 1 + v.
\]
By treating the \( Z \) jobs as jobs each having processing requirement \( v \) scheduled one after the other, we obtain the lower bound
\[
\bar{Z} = \sum_{j=1}^{t_z} jv = \frac{t_z(t_z + 1)}{2} \cdot v
\]
for the total completion time of the \( Z \) jobs in schedule \( \sigma \). Note that
\[
\bar{Z} = \hat{Z} - 3tb - tz - \frac{t(t - 1)}{2} \cdot z(b + 1).
\]
Thus,

\[ \sum C_j(\sigma) \geq S + X + Z + tc + xv + 1 \]

\[ > S + X + Z + tc + xv + 1 + tc + 1 + v - tc - 1 - 3tb - tz - \frac{u(t-1)}{2} \cdot z(b+1) \]

\[ = S + X + Z + tc + xv + 1 + v - 3tb - tz - \frac{u(t-1)}{2} \cdot z(b+1). \]

Now, \( v > 3tb + tz + \frac{u(t-1)}{2} \cdot z(b+1) \), so \( \sum C_j(\sigma) > Y \), a contradiction of the fact that \( \sigma \) is a good schedule. Therefore, schedule \( \sigma \) must satisfy (3.3). □

The fourth claim specifies the start time of job \( J_{X_1} \) in some good schedule that satisfies (3.1)-(3.3).

**Claim X3** *If there exists a good schedule that satisfies conditions (3.1)-(3.3), then there exists a good schedule that satisfies these conditions that also satisfies* \( \sigma(X_1) = tc + 1 \) and \( \sigma(j) < tc + 1 \) for all \( S \) and \( Z \) jobs \( J_j \).*

**Proof:** Among all good schedules that satisfy (3.1)-(3.3), let schedule \( \sigma \) be one with minimum total completion time. Suppose that schedule \( \sigma \) does not satisfy (3.4), so that, since \( \sigma \) satisfies (3.3), \( \sigma(X_1) > tc + 1 \). Note that the processing requirements of the \( S \) and \( Z \) jobs sum to \( tc + 1 \). Since \( \sigma \) satisfies (3.2), then jobs \( J_{X_2}, \ldots, J_{X_t} \) are scheduled after job \( J_{X_1} \) in schedule \( \sigma \). Thus, \( \sigma \) must include machine idle time prior to time \( \sigma(X_1) \), a contradiction of the fact that \( \sigma \) is a minimum total completion time schedule. Therefore, \( \sigma \) satisfies (3.4). □

The fifth claim addresses the number of \( Z \) jobs preceding each \( S \) job in some good schedule that satisfies the preceding four conditions.

**Claim Z1** *If there exists a good schedule that satisfies conditions (3.1)-(3.4), then there exists a good schedule that satisfies these conditions that also satisfies* \( job J_{S_i} \) is preceded by exactly \( iz \) \( Z \) jobs for each \( i = 0, 1, \ldots, t \).*

**Proof:** Among all good schedules that satisfy (3.1)-(3.4), let schedule \( \sigma \) be one with minimum total completion time. Suppose that schedule \( \sigma \) does not satisfy
condition (3.5), so that in \( \sigma \), some job \( J_s \) is preceded by either fewer than or more than \( iz \) \( Z \) jobs.

First, suppose that in schedule \( \sigma \), some job \( J_s \) is preceded by fewer than \( iz \) \( Z \) jobs. Then, the sum of processing requirements of jobs preceding job \( J_s \) in schedule \( \sigma \) is no more than

\[
i + (iz - 1)v + tb = i + izv - v + tb \\
= i(1 + zv + b) - v + tb - ib \\
= ic - v + tb - ib \\
< ic,
\]

where the last inequality follows since \( v > 3tb > tb \). By the definition of \( u_s \) and since \( \sigma \) satisfies (3.1), then \( \sigma(S_i) \geq ic \), so \( \sigma \) must include machine idle time prior to time \( \sigma(S_i) \), a contradiction of the fact that \( \sigma \) is a minimum total completion time schedule.

On the other hand, suppose that in schedule \( \sigma \), some job \( J_s \) is preceded by more than \( iz \) \( Z \) jobs. Then, the sum of processing requirements of jobs preceding job \( J_s \) in schedule \( \sigma \) is at least

\[
i + (iz + 1)v = ic + v - ib,
\]

so \( \sigma(S_i) \geq ic + v - ib \). Thus,

\[
\sum_{j=0}^{t} C_{S_j}(\sigma) \geq \sum_{j=0}^{t} (jc + 1) + v - ib = \hat{S} + v - ib.
\]

Moreover, the total completion time of the \( X \) and \( Z \) jobs in schedule \( \sigma \) is at least

\[
\sum_{j=1}^{v} (tc + 1 + jx) = \hat{X}
\]

and

\[
\sum_{j=1}^{t} jv = \hat{Z} - 3tb - tz - \frac{t(t - 1)}{2} \cdot z(b + 1),
\]
respectively. Hence,

\[ \sum C_j(\sigma) \geq \hat{S} + \hat{X} + \hat{Z} + v - ib - 3tb - tz - \frac{t(t-1)}{2} \cdot z(b+1). \]

Since \( v = z + (3tb + tz + \frac{t(t-1)}{2} \cdot z(b+1)) \) and \( z > ib \), then \( \sum C_j(\sigma) > Y \), a contradiction of the fact that \( \sigma \) is a good schedule. Therefore, \( \sigma \) satisfies condition (3.5). \( \square \)

The sixth and final claim in the proof of Proposition 3.12 specifies start times for the S jobs in some good schedule that satisfies (3.1)-(3.5).

Claim Z2. If there exists a good schedule that satisfies conditions (3.1)-(3.5), then there exists a good schedule that satisfies these conditions that also satisfies

\[ \sigma(S_j) = j \cdot c \text{ for each } j = 0, 1, \ldots, t. \]  \hspace{1cm} (3.6)

Proof: Among all good schedules that satisfy (3.1)-(3.5), let schedule \( \sigma \) be one with minimum total completion time. Suppose that schedule \( \sigma \) does not satisfy (3.6), so that \( \sigma(S_i) > ic \) for some \( i \) (since \( \sigma(*) \geq p(J) = tc + xv + 1 \), then \( \sigma(S_i) \geq p(J) - us_i - ps_i = ic \)). Since schedule \( \sigma \) satisfies (3.4), then \( \sigma(X_1) = tc + 1 \) and hence \( 0 \leq i \leq t - 1 \). Since \( \sigma \) satisfies (3.5), then the number of Z jobs scheduled between jobs \( J_{s_i} \) and \( J_{s_{i+1}} \) in schedule \( \sigma \) is \( z \). The total completion time of these \( z \) jobs in schedule \( \sigma \) is at least \( \sum_{j=1}^{z} (ic + 2 + jv) \). The total completion time of the \( Z \) jobs between jobs \( J_{s_n} \) and \( J_{s_{n+1}} \) in schedule \( \sigma \) is at least \( \sum_{j=1}^{z} (mc + 1 + jv) \) for any \( m = 0, 1, \ldots, t - 1 \). Thus, the total completion time of the \( Z \) jobs in schedule \( \sigma \) is at least

\[ \sum_{m=0}^{t-1} (mc + 1 + jv) + z = \hat{Z} - 3tb + z = \hat{Z} + 1. \]

The total completion time of the S and X jobs in schedule \( \sigma \) is at least \( \hat{S} + \hat{X} \). Now,

\[ \sum C_j(\sigma) \geq \hat{S} + \hat{X} + \hat{Z} + tc + xv + 2 = Y + 1. \]

a contradiction of the fact that \( \sigma \) is a good schedule. Therefore, schedule \( \sigma \) satisfies (3.6). \( \square \)
Table 3.2: Complexity classification of $1 / \text{min}$ and $\text{max}$ delays, $k$ 1-chains / $\sum C_j$ or $\sum w_j C_j$ problems.

<table>
<thead>
<tr>
<th>Type of Delays</th>
<th>Objective Function</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>min delays</td>
<td>$\sum C_j$</td>
<td>$O(k^3 \log k)$</td>
</tr>
<tr>
<td>min delays</td>
<td>$\sum w_j C_j$</td>
<td>NP-hard</td>
</tr>
<tr>
<td>max delays</td>
<td>$\sum C_j$</td>
<td>NP-hard in the strong sense</td>
</tr>
</tbody>
</table>

To complete the proof, suppose that there exists a good schedule. Then, by Claims S, X1, X2, X3, Z1, and Z3, there exists a good schedule $\sigma$ that satisfies conditions (3.1)-(3.6). Let $Z_i$ consist of those $Z$ jobs scheduled between jobs $J_{S_{i-1}}$ and $J_{S_i}$ in schedule $\sigma$ for $i = 1, \ldots, t$. The preceding six claims imply

$$|Z_i| = z \text{ and } \sum_{j \in Z_i} p_j = c - 1 = zv + b$$

for each $i = 1, \ldots, t$.

Since every $V$ job in $Z_i$ has processing requirement $v$, and every $W$ job in $Z_i$ has processing requirement between $v + \frac{b}{4}$ and $v + \frac{b}{2}$, then $Z_i$ must contain exactly three $W$ jobs with processing requirements that sum to $3v + b$. Thus, if we let

$$T_i = \{j : \sigma(S_{i-1}) < \sigma(W_j) < \sigma(S_i), j = 1, \ldots, 3t\}$$

for $i = 1, \ldots, t$, then $\sum_{j \in T_i} a_j = b$ for each $i = 1, \ldots, t$. \Box

In this chapter, we have investigated the computational complexity of total completion time and total weighted completion time problems with precedence relation $k$ 1-chains. The complexity results we have obtained are summarized in Table 3.2.
CHAPTER 4

1 / min and max delays,
2 $n_1, n_2$-chains / $C_{max}$

Chapter 2 addressed the computational complexity of minimum makespan problems for which the number of chains was a parameter, $k$. In this chapter, we fix $k = 2$ and investigate the complexity of two resultant problems. These two problems, 1 / min delays, 2 $n_1, n_2$-chains / $C_{max}$ and 1 / max delays, 2 $n_1, n_2$-chains / $C_{max}$, are the topics of Sections 4.1 and 4.2, respectively.

4.1 1 / min delays, 2 $n_1, n_2$-chains / $C_{max}$

Recall from Chapter 2 that associated with each feasible sequence (i.e., with each sequence that satisfies the ordinary precedence constraints underlying 2 $n_1, n_2$-chains) is an active schedule that schedules each job, and job * in particular, as early as possible so as to respect the sequence and to satisfy the machine capacity and the minimum delay precedence constraints. Thus, 1 / min delays, 2 $n_1, n_2$-chains / $C_{max}$ is the problem of finding, among all feasible sequences, a sequence that has associated active schedule with minimum makespan. In this section, we provide a characterization of the feasible sequences. Starting from this characterization, we develop a pseudo-polynomial time dynamic programming algorithm for 1 / min delays, 2 $n_1, n_2$-chains / $C_{max}$.

The 2 $n_1, n_2$-chains precedence relation imposes strict ordering requirements on jobs $J_{1,1}, \ldots, J_{1,n_1}$ and $J_{2,1}, \ldots, J_{2,n_2}$. Consequently, we can equate with each feasible
sequence a string of \(n_1-1\)'s and \(n_2-2\)'s. The \(r\)th symbol in the string is a 1 (2) if the first subscript of the \(r\)th job in the given sequence is a 1 (2) for \(r = 1, \ldots, n_1 + n_2\). The number of ways to choose \(n_1\) of \(n_1+n_2\) symbols to be 1's, and hence the number of feasible sequences, is equal to

\[
\binom{n_1 + n_2}{n_1}
\]

(see [8]), which is not bounded by any polynomial in \(n_1\) and \(n_2\). Thus, \(1 / \min\) delays, 2 \(n_1, n_2\)-chains / \(C_{\max}\) cannot be solved in polynomial time by explicitly enumerating all feasible sequences.

In the following discussion, we refer to job * as \(J_{1,n_1+1}\) and we let \(J_{1,0} (J_{2,0})\) be a job with zero processing requirement which must precede job \(J_{1,1} (J_{2,1})\) (see Figure 4.1). We define \(l_{1,0} = 0\) and \(l_{2,0} = 0\). We refer to jobs \(J_{1,0}, J_{1,1}, \ldots, J_{1,n_1+1}\) as the 1-jobs and to jobs \(J_{2,0}, J_{2,1}, \ldots, J_{2,n_2}\) as the 2-jobs.

Without loss of generality, \(J_{2,0}\) is the first job and \(J_{2,0}\) is the second job in any feasible sequence. Since \(J_{1,n_1+1}\) is necessarily the last job, then, in any feasible sequence, the 2-jobs other than \(J_{2,0}\) are interspersed among the 1-jobs. Due to the strict ordering requirements on the 2-jobs, each feasible sequence is characterized by the number of 2-jobs (other than \(J_{2,0}\)) between jobs \(J_{1,i}\) and \(J_{1,i+1}\) for \(i = 0, 1, \ldots, n_1\).

We might imagine there are \(n_1 + 1\) bins, one each between jobs \(J_{1,i}\) and \(J_{1,i+1}\) for \(i = 0, 1, \ldots, n_1\), into which the 2-jobs are placed (see Figure 4.2). Suppose that, having placed \(x_j\) of the 2-jobs in bin \(j\) for \(j = 1, \ldots, t - 1\), so that \(s_{t-1} = \sum_{j=1}^{t-1} x_j\) of the 2-jobs are in bins \(1, \ldots, t - 1\), we decide to place \(x_t\) of the \(n_2 - s_{t-1}\) remaining 2-jobs in bin \(t\). This decision corresponds to appending jobs \(J_{2,s_{t-1}+1}, \ldots, J_{2,s_{t-1}+x_t}\), and \(J_{1,t}\), in that order, to the end of the sequence

\[
J_{2,0} \rightarrow J_{1,0} \rightarrow J_{2,1} \rightarrow \cdots \rightarrow J_{2,x_1} \rightarrow J_{1,1} \rightarrow J_{2,x_1+1} \rightarrow \cdots \rightarrow J_{2,x_1+x_2} \rightarrow J_{1,2} \\
\rightarrow \cdots \rightarrow J_{1,t-2} \rightarrow J_{2,x_1+\ldots+x_{t-2}+1} \rightarrow \cdots \rightarrow J_{2,s_{t-1}} \rightarrow J_{1,t-1}.
\]

Let \(f_t\) be the contribution of these additional jobs to the makespan of the active schedule associated with the appended sequence. In other words, \(f_t\) is equal to

69
Figure 4.1: Strict ordering requirements imposed by $2n_1, n_2$-chains.

Figure 4.2: Bins into which the 2-jobs are placed.
the difference between the completion times of jobs $J_{1,t}$ and $J_{1,t-1}$ in the schedule associated with the appended sequence. Then, for each solution $x_1, \ldots, x_{n_1+1}$ of

$$x_1 + \cdots + x_{n_1+1} = n_2, \; x_j \in \mathbb{Z}_0^+ \text{ for } j = 1, \ldots, n_1 + 1,$$

the makespan of the schedule associated with the feasible sequence defined by $x_1, \ldots, x_{n_1+1}$ is equal to $\sum_{t=1}^{n_1+1} f_t$. For reasons which will soon be apparent, we define $w_{t-1}$ to be the difference between the start time of job $J_{1,t-1}$ and the completion time of job $J_{2,n_{t-1}}$ in the schedule associated with the original sequence.

We now exhibit formulas for $f_t$ in terms of $s_{t-1}$, $w_{t-1}$, and $x_t$ for each $t = 1, \ldots, n_1 + 1$ and each feasible combination of $x_1, \ldots, x_{t-1}$, and $x_t$. These formulas depend on $x_1, \ldots, x_{t-1}$ only through the roles these variables play in determining $s_{t-1}$ and $w_{t-1}$. Subsequently, we show that for $t = 1, \ldots, n_1 + 1$, both $s_t$ and $w_t$ can be computed given only $s_{t-1}$, $w_{t-1}$, and $x_t$, so that we can treat $1 / \min$ delays, $2 n_1, n_2$-chains / $C_{\max}$ as a discrete-time sequential decision process modeled as

$$z = \min_{x_1, \ldots, x_{n_1+1}} \sum_{t=1}^{n_1+1} f_t(s_{t-1}, w_{t-1}, x_t)$$

$$(s_t, w_t) = \phi_t(s_{t-1}, w_{t-1}, x_t) \text{ for } t = 1, \ldots, n_1 + 1$$

$$(s_0, w_0) \text{ given.}$$

Then, we prove that the dynamic programming recursion that arises from this model allows us to solve $1 / \min$ delays, $2 n_1, n_2$-chains / $C_{\max}$ in pseudo-polynomial time.

In exhibiting formulas for $f_t$, we consider four cases, the first and second with $t \in \{1, \ldots, n_1\}$ and either $x_t = 0$ or $x_t \in \{1, \ldots, n_2 - s_{t-1}\}$, and the third and fourth with $t = n_1 + 1$ and either $s_{t-1} = s_{n_1} = n_2$ or $s_{n_1} \in \{0, 1, \ldots, n_2 - 1\}$. From Figure 4.3, we see that for each $t = 1, \ldots, n_1$ and each feasible combination of $x_1, \ldots, x_{t-1}$, and $x_t$ with $x_t = 0,$

$$f_t = l_{1,t-1} + p_{1,t}. \quad (4.1)$$
Define

\[ PLP_{m,m'} = \sum_{r=m}^{m'-1} (p_{2,r} + l_{2,r}) + p_{2,m'}, \quad 1 \leq m \leq m' \leq n_2. \]

We see from Figure 4.4 that for each \( t = 1, \ldots, n_1 \) and each feasible combination of \( x_1, \ldots, x_{t-1}, \) and \( x_t \) with \( x_t \in \{1, \ldots, n_2 - s_{t-1}\} \), where \( s_{t-1} = \sum_{j=1}^{t-1} x_j \),

\[ f_t = g_1 + PLP_{x_{t-1}+1, x_{t-1}+x_t} + g_2 + p_{1,t}, \tag{4.2} \]

where

\[ g_2 = [l_{1,t-1} - (g_1 + PLP_{x_{t-1}+1, x_{t-1}+x_t})^+] \tag{4.3} \]

and

\[ g_1 = \{l_{2,x_{t-1}} - (w_{t-1} + p_{1,t-1})\}^+. \tag{4.4} \]

Examining Figure 4.5, we see that for each feasible combination of \( x_1, \ldots, x_{n_1}, \)

and \( x_{n_1+1} \) such that \( s_{n_1} = \sum_{j=1}^{n_1} x_j = n_2, \)

\[ f_{n_1+1} = \max\{l_{1,n_1}, [l_{2,n_2} - (w_{n_1} + p_{1,n_1})]^+\} + p_* \tag{4.5} \]

Finally, we see from Figure 4.6 that for each feasible combination of \( x_1, \ldots, x_{n_1}, \) and \( x_{n_1+1} \) such that \( s_{n_1} = \sum_{j=1}^{n_1} x_j \in \{0, 1, \ldots, n_2 - 1\}, \)

\[ f_{n_1+1} = g_1 + PLP_{x_{n_1+1}, n_2} + g_2 + p_*, \tag{4.6} \]
where
\[ g_2 = \max\{\left[l_1,n_1 - (g_1 + PLP_{s_n_1+n_2})\right]^+, l_2,n_2\} \] (4.7)

and
\[ g_1 = \{l_2,s_n_1 - (w_n_1 + p_1,n_1)\}^+. \] (4.8)

Equations (4.1)-(4.8) give formulas for \( f_t \) in terms of \( s_{t-1}, w_{t-1}, \) and \( x_t \) for each \( t = 1, \ldots, n_1 + 1 \) and each feasible combination of \( x_1, \ldots, x_{t-1}, \) and \( x_t. \) These formulas depend on \( x_1, \ldots, x_{t-1} \) only through the roles these variables play in determining \( s_{t-1} \) and \( w_{t-1}. \) We now show that, given \( s_{t-1}, w_{t-1}, \) and \( x_t, \) we can compute both \( s_t \) and \( w_t \) for \( t = 1, \ldots, n_1 + 1, \) so that the model on page 71 is of the correct form.

By definition,
\[ s_t = s_{t-1} + x_t \] for \( t = 1, \ldots, n_1 + 1, \) \( s_{t-1} = 0, 1, \ldots, n_2, \) and \( x_t = 0, 1, \ldots, n_2 - s_{t-1}, \)

where \( s_0 = 0. \) Thus, given \( s_{t-1} \) and \( x_t, \) we can compute \( s_t \) for \( t = 1, \ldots, n_1 + 1. \)

For each \( t = 1, \ldots, n_1 + 1 \) and each feasible combination of \( x_1, \ldots, x_{t-1}, \) and \( x_t, w_t \) is defined to be the difference between the start time of job \( J_{1,t} \) and the completion time of job \( J_{2,s_t} \) in the schedule associated with the sequence defined by \( x_1, \ldots, x_{t-1}, \) and \( x_t. \) From Figure 4.3 and equation (4.1), we see that for each
\( t = 1, \ldots, n, \) and each feasible combination of \( x_1, \ldots, x_{t-1}, \) and \( x_t \) with \( x_t = 0, \)

\[
w_t = w_{t-1} + p_{1,t-1} + l_{1,t-1}.
\]

(4.9)

We see from Figure 4.4 and equations (4.2)-(4.4) that for each \( t = 1, \ldots, n, \) and each feasible combination of \( x_1, \ldots, x_{t-1}, \) and \( x_t \) with \( x_t \in \{1, \ldots, n_2 - s_{t-1}\}, \)

\[
w_t = g_2 = \left[ l_{1,t-1} - (g_1 + P\L P_{s_{t-1}+1,s_{t-1}+x_t}) \right]^+, \quad (4.10)
\]

where

\[
g_1 = \{l_{2,s_{t-1}} - (w_{t-1} + p_{1,t-1})\}^+. \quad (4.11)
\]

Equations (4.9)-(4.11) make sense only if \( w_t \equiv 0. \) Since there are only \( n_1 + 1 \) decisions to be made, then \( w_{n_1+1} \) is irrelevant. We adopt the convention \( w_{n_1+1} = 0 \) for each feasible combination of \( x_1, \ldots, x_{n_1+1}. \) Therefore, given \( s_{t-1}, w_{t-1}, \) and \( x_t, \)

we can compute both \( s_t \) and \( w_t \) for \( t = 1, \ldots, n_1 + 1. \)

A potential difficulty with the model on page 71 is in the large number of distinct \( w_{t-1}'s \) that might be paired with a given \( s_{t-1}. \) The number of feasible sequences of jobs \( J_{1,1}, \ldots, J_{1,t-1} \) and \( J_{2,1}, \ldots, J_{2,s_{t-1}} \) with job \( J_{1,t-1} \) last is equal to the number of ways to choose \( t - 2 \) of \( t - 2 + s_{t-1} \) symbols to be 1's. In other words, as many as

\[
\binom{t - 2 + s_{t-1}}{t - 2}
\]

distinct \( w_{t-1}'s \) might be paired with \( s_{t-1} \) for each \( t = 2, \ldots, n_1 \) and \( s_{t-1} = 0, 1, \ldots, n_2. \)

Since

\[
C^u = \sum_{i=1}^{n_1} (p_{1,i} + l_{1,i}) + \sum_{j=1}^{n_2} (p_{2,j} + l_{2,j}) + p_*
\]

is an upper bound for the makespan of any schedule, then \( w_{t-1} < C^u. \) Thus, the number of distinct \( w_{t-1}'s \) paired with \( s_{t-1} \) is at most \( C^u \) for each \( t = 2, \ldots, n_1 \) and \( s_{t-1} = 0, 1, \ldots, n_2. \) (As always, we assume that the data are integral.) We now prove that we can replace \( w_{t-1} \) by \( \min\{w_{t-1}, [l_{2,s_{t-1}} - p_{1,t-1}]^+\}, \) so that the number of distinct \( w_{t-1}'s \) paired with \( s_{t-1} \) is at most \( 1 + [l_{2,s_{t-1}} - p_{1,t-1}]^+ \) for each \( t = 2, \ldots, n, \) and \( s_{t-1} = 0, 1, \ldots, n_2. \)
Examining equations (4.2)-(4.8), we see that \( w_{t-1} \geq l_{2,s_{t-1}} - p_{1,t-1} \) is a condition under which each \( f_t \) is in fact independent of \( w_{t-1} \). We see from equations (4.10) and (4.11) that \( w_{t-1} \geq l_{2,s_{t-1}} - p_{1,t-1} \) is also a condition under which, for each \( t = 1, \ldots, n_1 \) and each feasible combination of \( x_1, \ldots, x_{t-1} \), and \( x_t \) with \( x_t \in \{1, \ldots, n_2 - s_{t-1}\} \), \( w_t \) is in fact independent of \( w_{t-1} \). From equation (4.9), we see that for each \( t = 1, \ldots, n_1 \) and each feasible combination of \( x_1, \ldots, x_{t-1} \), and \( x_t \) with \( x_t = 0 \),

\[
 w_{t-1} \geq l_{2,s_{t-1}} - p_{1,t-1} \Rightarrow w_t \geq l_{2,s_{t-1}} + l_{1,t-1} = l_{2,s_t} + l_{1,t-1} \geq l_{2,s_t} - p_{1,t},
\]

that is, \( w_{t-1} \geq l_{2,s_{t-1}} - p_{1,t-1} \) is a condition under which \( w_t \geq l_{2,s_t} - p_{1,t} \). Therefore, we are justified in replacing \( w_{t-1} \) by \( \min\{w_{t-1}, [l_{2,s_{t-1}} - p_{1,t-1}]^+\} \) for each \( t = 2, \ldots, n_1 \) and \( s_{t-1} = 0, 1, \ldots, n_2 \).

We now state the dynamic programming recursion that arises from the model given on page 71. Let \( z_0(s_0, w_0) = z_0(0,0) = 0 \). For each \( m = 1, \ldots, n_1 + 1 \) and each pair \( (s_m, w_m) \), let

\[
 z_m(s_m, w_m) = \min_{x_1, \ldots, x_m} \sum_{t=1}^{m} f_t(s_{t-1}, w_{t-1}, x_t)
\]

\[
 (s_t, w_t) = \phi_t(s_{t-1}, w_{t-1}, x_t) \text{ for } t = 1, \ldots, m
\]

\[
 (s_0, w_0) = (0,0).
\]

Then, \( z = z_{n_1+1}(n_2,0) \). By the principle of dynamic programming optimality (see [4] for example),

\[
 z_m(s_m, w_m) = \min_{\Omega} (f_m(s_{m-1}, w_{m-1}, x_m) + z_{m-1}(s_{m-1}, w_{m-1})) \tag{4.12}
\]

\[
 \Omega = \{s_{m-1}, w_{m-1}, \text{ and } x_m : \phi(s_{m-1}, w_{m-1}, x_m) = (s_m, w_m)\}
\]

for each \( m = 1, \ldots, n_1 + 1 \) and each pair \( (s_m, w_m) \).

The following proposition gives the computational complexity of solving 1 / min delays, 2 \( n_1, n_2 \)-chains / \( C_{\max} \) using (4.12).

**Proposition 4.1** The 1 / min delays, 2 \( n_1, n_2 \)-chains / \( C_{\max} \) problem can be solved using the dynamic programming recursion (4.12) in time \( O(n_1 n_2^3(1 + [L_2 - p_1]^+)) \), where \( L_2 = \max_{j=1,\ldots,n_2} l_{2,j} \) and \( p_1 = \min_{i=2,\ldots,n_1} p_{1,i} \).
Proof: The bottleneck operations in computing $z_m(s_m, w_m)$ are calculating and determining the minimum of at most $s_m + 1$ quantities. The number of distinct $w_m$'s paired with a given $s_m$ is at most $1 + [L_2 - p_1]$. Thus, for each $m = 1, \ldots, n_1 + 1$, computing $z_m(s_m, w_m)$ for each $s_m$ and $w_m$ requires time

$$O((1 + [L_2 - p_1]) \sum_{s_m=0}^{n_2} (s_m + 1)) = O(n_2^2(1 + [L_2 - p_1])).$$

We have tried without success to classify $1 / \min$ delays, $2 n_1, n_2$-chains $/ C_{\max}$ more precisely with respect to its computational complexity. Whether or not this problem is solvable in polynomial time and whether or not this problem is NP-hard are open questions.

4.2 1 / max delays, 2 $n_1, n_2$-chains $/ C_{\max}$

In Section 4.1, we presented a characterization of the feasible sequences, that is, of the sequences that satisfy the ordinary precedence constraints underlying $2 n_1, n_2$-chains. In this section, we show that this characterization leads to a polynomial time dynamic programming algorithm for 1 / max delays, 2 $n_1, n_2$-chains $/ C_{\max}$.

Without loss of generality, solutions to 1 / max delays, 2 $n_1, n_2$-chains $/ C_{\max}$ include no machine idle time, since removing machine idle time from a schedule that satisfies the maximum delay precedence constraints results in a feasible schedule with smaller makespan. Schedules without machine idle time are necessarily minimum makespan schedules. Thus, 1 / max delays, 2 $n_1, n_2$-chains $/ C_{\max}$ is the problem of finding a schedule without machine idle time that satisfies the maximum delay precedence constraints.

As in Section 4.1, we refer to job $*$ as $J_{1, n_1+1}$ and we let $J_{1,0}$ ($J_{2,0}$) be a job with zero processing requirement which must precede job $J_{1,1}$ ($J_{2,1}$). We refer to jobs $J_{1,0}, J_{1,1}, \ldots, J_{1,n_1+1}$ as the 1-jobs and to jobs $J_{2,0}, J_{2,1}, \ldots, J_{2,n_2}$ as the 2-jobs. We
define \( l_{1,0} = \infty \) and \( l_{2,0} = \infty \) so that, without loss of generality, \( J_{2,0} \) is the first job and \( J_{1,0} \) is the second job in any feasible sequence. Then, each feasible sequence is characterized by the number of 2-jobs (other than \( J_{2,0} \)) between jobs \( J_{1,i} \) and \( J_{1,i+1} \) for \( i = 0, 1, \ldots, n_1 \).

We say that a schedule of the jobs in \( J \) is feasible if that schedule satisfies the maximum delay precedence constraint corresponding to each \( < J_r, J_r' > \in P \). In the same vein, we say that a schedule of the jobs in \( J' \subseteq J \) is feasible if that schedule satisfies the maximum delay precedence constraint corresponding to each \( < J_r, J_r' > \in P \) such that \( J_r \in J' \) and \( J_r' \in J' \).

Again as in Section 4.1, we might imagine there are \( n_1 + 1 \) bins, one each between jobs \( J_{1,i} \) and \( J_{1,i+1} \) for \( i = 0, 1, \ldots, n_1 \), into which the 2-jobs are placed (see Figure 4.2). Suppose that, having placed \( x_j \) of the 2-jobs in bin \( j \) for \( j = 1, \ldots, t-1 \), so that \( s_{t-1} = \sum_{j=1}^{t-1} x_j \) of the 2-jobs are in bins \( 1, \ldots, t-1 \), we decide to place \( x_t \) of the \( n_2 - s_{t-1} \) remaining 2-jobs in bin \( t \). Let \( w_{t-1} \) be the difference between the start time of job \( J_{1,t-1} \) and the completion time of job \( J_{2,s_{t-1}} \) in the schedule (without machine idle time) defined by \( x_1, \ldots, x_{t-1} \). Then, from Figure 4.7, we see that for \( t = 1, \ldots, n_1 \), the schedule defined by \( x_1, \ldots, x_{t-1}, \) and \( x_t \) is feasible if and only if

1. the schedule defined by \( x_1, \ldots, x_{t-1} \) is feasible and

2. if \( x_t > 0 \), then \( w_{t-1} + p_{1,t-1} \leq u_{2,s_{t-1}} \) and

\[
\sum_{r=s_{t-1}+1}^{s_{t-1}+x_t} p_{2,r} \leq u_{1,t-1}.
\]

Moreover, we see from Figure 4.8 that the schedule defined by \( x_1, \ldots, x_{n_1}, \) and \( x_{n_1}+1 \) is feasible if and only if

1. the schedule defined by \( x_1, \ldots, x_{n_1} \) is feasible and
Figure 4.7: Schedule defined by $x_1, \ldots, x_{t-1}$, and $x_t$.

\[
\begin{array}{c|c|c}
J_{2,s_{t-1}} & J_{1,t-1} & J_{2,s_{t-1}+1} \\
\end{array}
\begin{array}{c|c|c}
J_{1,t} & J_{2,s_{t-1}+x_t}
\end{array}
\]

Figure 4.8: Schedule defined by $x_1, \ldots, x_{n_1}$, and $x_{n_1+1}$.

\[
\begin{array}{c|c|c}
J_{2,n_1} & J_{1,n_1} & J_{2,n_1+1} \\
\end{array}
\begin{array}{c|c|c}
J_{2,n_2} & *
\end{array}
\]

2. if $x_{n_1+1} = 0$, then $w_{n_1} + p_{1,n_1} \leq u_{2,n_2}$; otherwise,

\[
\sum_{r=s_{n_1+1}}^{n_2} p_{2,r} \leq u_{1,n_1}.
\]

Thus, given that the schedule defined by $x_1, \ldots, x_{t-1}$ is feasible, we can easily determine whether or not the schedule defined by $x_1, \ldots, x_{t-1}$, and $x_t$ is feasible for $t = 1, \ldots, n_1$.

Let $x'_1, \ldots, x'_t$ and $x''_1, \ldots, x''_t$ be distinct solutions of

\[x_1 + \cdots + x_t = s_t, \quad x_j \in \mathbb{Z}_0^+ \text{ for } j = 1, \ldots, t\]

for some $t \in \{2, \ldots, n_1\}$ and $s_t \in \{1, \ldots, n_2\}$. Assume that both $\sigma'$ and $\sigma''$, the schedules defined by $x'_1, \ldots, x'_t$ and $x''_1, \ldots, x''_t$, respectively, are feasible. We say that $x'_1, \ldots, x'_t$ dominates $x''_1, \ldots, x''_t$ if the existence of $x''_{t+1}, \ldots, x''_{n_1+1}$ with $\sum_{j=t+1}^{n_1+1} x''_j = n_2 - s_t$ such that the schedule defined by $x''_1, \ldots, x''_{n_1+1}$ is feasible implies the existence of $x'_{t+1}, \ldots, x'_{n_1+1}$ with $\sum_{j=t+1}^{n_1+1} x'_j = n_2 - s_t$ such that the schedule de-
fined by $x'_1, \ldots, x'_{n_1+1}$ is feasible. We now present a sufficient condition for $x'_1, \ldots, x'_t$ to dominate $x''_1, \ldots, x''_t$.

**Proposition 4.2** If $\sigma'(2, s_t) \geq \sigma''(2, s_t)$, then $x'_1, \ldots, x'_t$ dominates $x''_1, \ldots, x''_t$.

**Proof:** Suppose that the schedule, $\sigma''$, defined by $x''_1, \ldots, x''_{n_1+1}$ is feasible. We verify that the schedule, $\sigma'$, defined by $x'_1, \ldots, x'_t, x''_{t+1}, \ldots, x''_{n_1+1}$ is feasible. Since $\sigma'$ is feasible, then $\sigma'$ satisfies the maximum delay precedence constraints corresponding to $<J_r, J_r'> \in P$ such that $J_r \in J'^{*,1} = \{J_{1,0}, J_{1,1}, \ldots, J_{1,t}\} \cup \{J_{2,0}, J_{2,1}, \ldots, J_{2,s_t}\}$ and $J_r' \in J'^{*,2}$. Since $\sigma''$ is feasible, then $\sigma'$ satisfies the maximum delay precedence constraints corresponding to $<J_r, J_r'> \in P$ such that $J_r \in J \backslash J'^{*,1}$ and $J_r' \in J \backslash J'^{*,2}$. Now, $<J_r, J_r'> \in P$ such that $J_r \in J'^{*,1}$ and $J_r' \in J \backslash J'^{*,2}$ consists of $<J_{1,t}, J_{1,t+1}>$ and, if $s_t < n_2$, of $<J_{2,s_t}, J_{2,s_t+1}>$. Schedule $\sigma'$ satisfies the maximum delay precedence constraint corresponding to $<J_{1,t}, J_{1,t+1}>$ since

$$\sigma'(1,t) = \sigma''(1,t),$$

$$C_{1,t+1}(\sigma') = C_{1,t+1}(\sigma''),$$

and

$$C_{1,t+1}(\sigma'') - \sigma''(1,t) \leq u_{1,t}.$$ 

Since

$$\sigma'(2,s_t) = \sigma'(2,s_t) \leq \sigma''(2,s_t) = \sigma''(2,s_t),$$

$$C_{2,s_t+1}(\sigma') = C_{2,s_t+1}(\sigma''),$$

and

$$C_{2,s_t+1}(\sigma'') - \sigma''(2,s_t) \leq u_{2,s_t},$$

then schedule $\sigma'$ satisfies the maximum delay precedence constraint corresponding to $<J_{2,s_t}, J_{2,s_t+1}>$. □

Using Proposition 4.2, we can solve $1 / \text{max delays}, 2 n_1, n_2$-chains $/ C_{\text{max}}$ by dynamic programming as follows. For $t = 1, \ldots, n_1$ and $s_t = 0, 1, \ldots, n_2$, we check
the feasibility of the schedule obtained by adding jobs $J_{2,s_{t-1}+1}, \ldots, J_{2,s_t}$ and $J_{1,t}$, in that order, to the end of a feasible schedule (if any) of jobs $J_{1,0}, J_{1,1}, \ldots, J_{1,t-1}$ and $J_{2,0}, J_{2,1}, \ldots, J_{2,s_{t-1}}$ with job $J_{1,t-1}$ scheduled last for $s_{t-1} = 0, 1, \ldots, s_t$ and we select from among the feasible schedules (if any) one with job $J_{2,s_t}$ scheduled latest. Then, for $t = n_1 + 1$, we check the feasibility of the schedule obtained by adding jobs $J_{2,s_{n_1}+1}, \ldots, J_{2,n_2}$ and $*$, in that order, to the end of a feasible schedule (if any) of jobs $J_{1,0}, J_{1,1}, \ldots, J_{1,n_1}$ and $J_{2,0}, J_{2,1}, \ldots, J_{2,s_{n_1}}$ with job $J_{1,n_1}$ last for $s_{n_1} = 0, 1, \ldots, n_2$. The time required to solve $1 / \max$ delays, $2$ $n_1, n_2$-chains / $C_{\max}$ using this recursive procedure is

$$O(n_1 \sum_{s_t=0}^{n_2} (s_t + 1) + n_2 + 1) = O(n_1 n_2^2).$$

In this chapter, we have investigated the computational complexity of minimum makespan problems for which the number of chains is two. In particular, we showed that $1 / \min$ delays, $2$ $n_1, n_2$-chains / $C_{\max}$ can be solved in pseudo-polynomial time and that $1 / \max$ delays, $2$ $n_1, n_2$-chains / $C_{\max}$ can be solved in time $O(n_1 n_2^2)$. 

80
CHAPTER 5

SPECIAL CASES, HEURISTICS, AND BOUNDS

In this chapter, we present a miscellany of results including polynomially solvable special cases, heuristics, and bounds for two problems which are not known to be solvable in polynomial time. The first problem, 1 / min delays, k 2, 1, ..., 1-chains / C_{max}, was shown to be NP-hard in Chapter 2. The second problem, 1 / min delays, 2 n_1, n_2-chains / C_{max}, was shown to be solvable in pseudo-polynomial time in Chapter 4.

5.1 1 / min delays, k 2, 1, ..., 1-chains / C_{max}

5.1.1 Relative Ordering of Jobs J_2, ..., J_k

In each feasible sequence for k 2, 1, ..., 1-chains, each job from \{J_2, ..., J_k\} appears either before job J_{x_1}, between jobs J_{x_1} and J_{x_2}, or between jobs J_{x_2} and *. The following proposition describes the relative ordering of the jobs from \{J_2, ..., J_k\} which appear before job J_{x_1}, between jobs J_{x_1} and J_{x_2}, and between jobs J_{x_2} and * in the sequence associated with some optimal schedule.

Proposition 5.1 There exists an optimal schedule \sigma^* with associated sequence of the form

\[ J_{e_2} \rightarrow \cdots \rightarrow J_{e_i} \rightarrow J_{x_1} \rightarrow J_{e_{i+1}} \rightarrow \cdots \rightarrow J_{e_m} \rightarrow J_{x_2} \rightarrow J_{e_{m+1}} \rightarrow \cdots \rightarrow J_{e_n} \rightarrow \ast, \]
where $1 \leq i \leq m \leq k$, such that
\[ l_{e_2} \geq \cdots \geq l_{e_1}, \quad l_{e_{m+1}} \geq \cdots \geq l_{e_m}, \quad \text{and} \quad l_{e_{m+1}} \geq \cdots \geq l_{e_k}. \] (5.1)

**Proof:** Let $\sigma$ be any optimal schedule and suppose that the sequence associated with schedule $\sigma$ does not satisfy condition (5.1). Then, there exist adjacent jobs $J_j$ and $J_{j'}$, both in $\{J_2, \ldots, J_k\}$, such that $J_j \rightarrow J_{j'}$ but $l_j < l_{j'}$. Let $\sigma'$ be the schedule obtained from schedule $\sigma$ by interchanging jobs $J_j$ and $J_{j'}$. Let $\Delta = \max_{J \in \{J_j, J_{j'}, \star\}} \{C_r(\sigma') + l_r\} = \max_{J \in \{J_j, J_{j'}, \star\}} \{C_r(\sigma) + l_r\}$. Then
\[
C_{\text{max}}(\sigma') = \max\{\Delta, C_j(\sigma') + l_j, C_{j'}(\sigma') + l_{j'}\} \\
= \max\{\Delta, \sigma(j) + p_j + p_{j'} + l_j, \sigma(j) + p_{j'} + l_{j'}\} \\
\leq \max\{\Delta, \sigma(j) + p_j + p_{j'} + l_j, \sigma(j) + p_{j'} + l_{j'}, \sigma(j) + p_j + p_{j'} + l_{j'}\} \\
= \max\{\Delta, \sigma(j) + p_j + p_{j'} + l_j\} \\
= \max\{\Delta, \sigma(j) + p_j + l_j, \sigma(j) + p_j + p_{j'} + l_{j'}\} \\
= \max\{\Delta, C_j(\sigma) + l_j, C_{j'}(\sigma) + l_{j'}\} \\
= C_{\text{max}}(\sigma).
\]
Repeating this argument, we see that schedule $\sigma$ can be transformed into a schedule that satisfies (5.1) without affecting the makespan. $\square$

The number of sequences of the form
\[ J_{e_2} \rightarrow \cdots \rightarrow J_{e_i} \rightarrow J_{e_1} \rightarrow J_{e_{i+1}} \rightarrow \cdots \rightarrow J_{e_m} \rightarrow J_{e_2} \rightarrow J_{e_{m+1}} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow \star \]
that satisfy (5.1), where $1 \leq i \leq m \leq k$, is equal to the number of ways to distribute $k - 1$ labeled objects (i.e., jobs $J_2, \ldots, J_k$) into three labeled bins, one before job $J_{e_1}$, one between jobs $J_{e_2}$ and $J_{e_3}$, and one between jobs $J_{e_2}$ and $\star$, where bins are allowed to be empty. By a straightforward combinatorial argument, we can show that the number of such distributions is $3^{k-1}$.

**5.1.2 Heuristic with Worst Case Performance Ratio 2**

The worst case performance ratio of a heuristic algorithm for a given minimization (maximization) problem is defined to be the supremum (infimum) taken over
all problem instances of the ratio of the value of the heuristic solution to the optimal value. As a prelude to presenting a heuristic algorithm for $1 / \min$ delays, $k \ 2, 1, \ldots, 1$-chains / $C_{\max}$ with a worst case performance ratio of 2, we describe two relaxations which yield lower bounds for the optimal makespan. The first relaxation of the instance of $1 / \min$ delay, $k \ 2, 1, \ldots, 1$-chains / $C_{\max}$ shown in Figure 5.1 is obtained simply by eliminating job $J_{z_1}$ (see Figure 5.2). The second relaxation is obtained by eliminating job $J_{z_2}$ and imposing a minimum delay of $l_{z_1} + p_{z_2} + l_{z_2}$ between jobs $J_{z_1}$ and $*$ (see Figure 5.3). Clearly, if $\sigma^1 : J \setminus \{J_{z_1}\} \rightarrow \mathbb{Z}_0^+$ is an optimal schedule for the instance of $1 / \min$ delays, $k \ 1$-chains / $C_{\max}$ shown in Figure 5.2 and $\sigma^*$ is an optimal schedule for the instance of $1 / \min$ delays, $k \ 2, 1, \ldots, 1$-chains / $C_{\max}$ shown in Figure 5.1, then

$$C_\ast(\sigma^1) \leq C_\ast(\sigma^*).$$

Similarly, if $\sigma^2 : J \setminus \{J_{z_2}\} \rightarrow \mathbb{Z}_0^+$ is an optimal schedule for the instance of $1 / \min$ delays, $k \ 1$-chains / $C_{\max}$ shown in Figure 5.3, then

$$C_\ast(\sigma^2) \leq C_\ast(\sigma^*).$$

From $\sigma^1$, we can obtain a schedule for $1 / \min$ delays, $k \ 2, 1, \ldots, 1$-chains / $C_{\max}$ with a makespan that exceeds the makespan of schedule $\sigma^1$ by at most $p_{z_1} + l_{z_1}$. In Section 2.1.1, we proved that $1 / \min$ delays, $k \ 1$-chains / $C_{\max}$ is solved by sequencing jobs $J_1, \ldots, J_k$ in order of nonincreasing precedence delay. Thus, we may assume that the sequence associated with schedule $\sigma^1$ has the form

$$J_{e_2} \rightarrow \cdots \rightarrow J_{e_i} \rightarrow J_{z_1} \rightarrow J_{e_{i+1}} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow \ast,$$

where $1 \leq i \leq k$ and $l_{e_2} \geq \cdots \geq l_{e_i} \geq l_{z_1} \geq l_{e_{i+1}} \geq \cdots \geq l_{e_k}$. Let $\tilde{\sigma}^1$ be the schedule obtained from $\sigma^1$ by scheduling job $J_{z_1}$ first, that is, $\tilde{\sigma}^1$ is the schedule associated with the sequence

$$\tilde{J}_{z_1} \rightarrow J_{e_2} \rightarrow \cdots \rightarrow J_{e_i} \rightarrow J_{z_2} \rightarrow J_{e_{i+1}} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow \ast.$$
Figure 5.1: Instance of $1 / \min$ delays, $k = 2, 1, \ldots, 1$-chains / $C_{\text{max}}$.

Figure 5.2: First relaxation of $1 / \min$ delays, $k = 2, 1, \ldots, 1$-chains / $C_{\text{max}}$ instance.
From Figure 5.4, we see that
\[
\hat{\sigma}^1(x_2) = \sigma^1(x_2) + p_{x_1} + [l_{x_1} - \sum_{i=2}^{m} p_{x_i}]^+ \\
\leq \sigma^1(x_2) + p_{x_1} + l_{x_1},
\]
which implies
\[
C_*(\hat{\sigma}^1) - C_*(\sigma^1) \leq p_{x_1} + l_{x_1}.
\]

Similarly, from $\sigma^2$, we can obtain a schedule for 1 / min delays, $k 2, 1, \ldots, 1$-chains / $C_{\text{max}}$ with a makespan that exceeds the makespan of schedule $\sigma^2$ by at most $p_{x_2} + l_{x_2}$. We may assume that the sequence associated with schedule $\sigma^2$ has the form
\[
J_{x_2} \rightarrow \cdots \rightarrow J_{x_m} \rightarrow J_{x_1} \rightarrow J_{x_1}^+ \rightarrow \cdots \rightarrow J_{x_k} \rightarrow *,
\]
where $1 \leq m \leq k$ and $l_{x_2} \geq \cdots \geq l_{x_m} \geq l_{x_1} + p_{x_2} + l_{x_2} \geq l_{x_1}^+ \geq \cdots \geq l_{x_k}$. Let
\( \hat{\sigma}^2 \) be the schedule obtained from \( \sigma^2 \) by inserting job \( J_{x_2} \) as soon as possible so as to satisfy the minimum delay precedence constraint corresponding to \( \prec J_{x_1}, J_{x_2} \), that is, \( \hat{\sigma}^2 \) is the schedule associated with the sequence

\[
J_{x_2} \rightarrow \cdots \rightarrow J_{e_m} \rightarrow J_{x_1} \rightarrow J_{e_{m+1}} \rightarrow \cdots \rightarrow J_{e_r} \rightarrow J_{x_2} \rightarrow J_{e_{r+1}} \rightarrow \cdots \rightarrow J_{e_k} \rightarrow *,
\]

where \( r = \text{argmin}\{s = m + 1, \ldots, k : \sum_{t=m+1}^{t} p_{e_t} \geq l_{x_1}\} \). If no such \( r \) exists, then job \( J_{x_2} \) can be inserted into schedule \( \sigma^2 \) before job \( * \) without affecting the makespan, in which case \( C_*(\hat{\sigma}^2) = C_*(\sigma^2) \). On the other hand, if \( r \) exists, then, from Figure 5.5, we see that

\[
\hat{\sigma}^2(\ast) = \sigma^2(\ast) + p_{x_2} + [l_{x_2} - \sum_{t=r+1}^{k} p_{e_t}]^+.
\]

Thus,

\[
C_*(\hat{\sigma}^2) - C_*(\sigma^2) \leq p_{x_2} + l_{x_2}.
\]

We now prove that the heuristic that involves selecting between \( \hat{\sigma}^1 \) and \( \hat{\sigma}^2 \) the schedule with smaller makespan has a worst case performance ratio of 2.

**Proposition 5.2** Let \( \sigma = \text{argmin}_{\hat{\sigma} = \hat{\sigma}^1, \hat{\sigma}^2} \{C_*(\hat{\sigma})\} \). Then \( C_*(\sigma) \leq 2C_*(\sigma^*) \).

**Proof:** We consider two cases, the first with \( p_{x_1} + l_{x_1} \leq p_{x_2} + l_{x_2} \) and the other with \( p_{x_1} + l_{x_1} > p_{x_2} + l_{x_2} \). If \( p_{x_1} + l_{x_1} \leq p_{x_2} + l_{x_2} \), then

\[
\frac{C_*(\sigma)}{C_*(\sigma^*)} \leq \frac{C_*(\hat{\sigma}^1)}{C_*(\sigma^*)}
\]

86
\[
\textstyle \frac{C_*(\sigma^1)}{C_*(\sigma^0)} \leq \frac{C_*(\sigma^0) + p_{x_1} + l_{x_1}}{C_*(\sigma^0)}
\]
\[
\textstyle \leq \frac{C_*(\sigma^0) + p_{x_2} + l_{x_2}}{C_*(\sigma^0)}
\]
\[
\textstyle \leq \frac{2C_*(\sigma^0)}{C_*(\sigma^0)} = 2,
\]

where the last inequality follows since \( C_*(\sigma^0) \geq p_{x_2} + l_{x_2} \). On the other hand, if \( p_{x_1} + l_{x_1} > p_{x_2} + l_{x_2} \), then
\[
\textstyle \frac{C_*(\sigma^0)}{C_*(\sigma^0)} \leq \frac{C_*(\sigma^2)}{C_*(\sigma^0)}
\]
\[
\textstyle \leq \frac{C_*(\sigma^2) + p_{x_2} + l_{x_2}}{C_*(\sigma^2)}
\]
\[
\textstyle \leq \frac{C_*(\sigma^2) + p_{x_1} + l_{x_1}}{C_*(\sigma^2)}
\]
\[
\textstyle \leq \frac{2C_*(\sigma^2)}{C_*(\sigma^2)} = 2,
\]

where the last inequality follows since \( C_*(\sigma^2) \geq p_{x_1} + l_{x_1} \). 

This heuristic can be accomplished in time \( O(k \log k) \), the time required to sequence \( k \) jobs in order of nonincreasing precedence delay.
The proof of Proposition 5.2 is based on the inequalities
\[ C_*(\sigma^1) - C_*(\sigma^1) \leq p_{x_1} + l_{x_1} \]
and
\[ C_*(\sigma^2) - C_*(\sigma^2) \leq p_{x_2} + l_{x_2}. \]
Proposition 5.2 gives the strongest possible result only in the seemingly unlikely event that for some instance, both of these inequalities are tight. Conceivably, a worst case performance ratio less than 2 could be established. As the following series of examples shows, though, the heuristic’s worst case performance ratio is at least \( \frac{3}{2} \). Let \( k = 4 \), \( p_{x_1} = p_{x_2} = p_s = 1 \), \( p_2 = p_3 = \frac{3}{2} s \), \( p_4 = s \), \( l_{x_1} = s + 1 \), \( l_{x_2} = s \), \( l_2 = l_3 = 1 \), and \( l_4 = 0 \). Then \( \sigma^1 \), the schedule corresponding to sequence \( J_{x_1} \rightarrow J_{x_2} \rightarrow J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow * \), has makespan \( 3s + 4 \) and \( \sigma^2 \), the schedule corresponding to sequence \( J_{x_1} \rightarrow J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_{x_2} \rightarrow * \) has makespan \( 3s + 3 \). The optimal schedule, corresponding to sequence \( J_{x_1} \rightarrow J_2 \rightarrow J_3 \rightarrow J_{x_2} \rightarrow J_4 \rightarrow * \), has makespan \( 2s + 4 \). Thus,
\[ \lim_{s \to \infty} \frac{C_*(\sigma)}{C_*(\sigma^*)} = \lim_{s \to \infty} \frac{3s + 3}{2s + 4} = \frac{3}{2}. \]
Schedules \( \sigma^1 \) and \( \sigma^2 \) belong to the class of schedules associated with sequences obtained by first ordering jobs \( J_2, \ldots, J_k \) by nonincreasing precedence delay and then inserting jobs \( J_{x_1} \) and \( J_{x_2} \), with job \( J_{x_1} \) before job \( J_{x_2} \). The number of such insertions is \( O(k^2) \). Since computing the makespan of the schedule associated with a given sequence requires time \( O(k) \), then finding a schedule with minimum makespan among these insertion schedules can be accomplished in time \( O(k \lg k + k^3) \).

5.1.3 Pseudo-polynomial Algorithm for Special Case with \( l_2 = \cdots = l_k = 0 \)

In proving that \( 1 / \min \text{ delays, } k \ 2, 1, \ldots, 1\)-chains / \( C_{\max} \) is NP-hard, we in fact proved that the special case with \( l_2 = \cdots = l_k = 0 \) is NP-hard. We now prove
Figure 5.6: Obtaining schedule $\sigma^*$ from schedule $\sigma^*$.

that this special case is not NP-hard in the strong sense by exhibiting a pseudo-polynomial time algorithm.

The following lemma specifies the first job scheduled in some optimal schedule.

**Lemma 5.3** There exists an optimal schedule for the special case of $1 / \min_i$ delays, $k 2,1,\ldots, 1$-chains / $C_{\max}$ with $l_2 = \cdots = l_k = 0$ in which the first job scheduled is $J_{z_1}$.

**Proof:** Let $\sigma^*$ be any optimal schedule. If the first job scheduled in $\sigma^*$ is $J_{z_1}$, the proof is complete, so suppose that jobs $J_{z_2}, \ldots, J_{z_k}$ are scheduled before job $J_{z_1}$ and jobs $J_{z_{i+1}}, \ldots, J_{z_m}$ are scheduled between jobs $J_{z_1}$ and $J_{z_2}$ in schedule $\sigma^*$, where $1 \leq i \leq m \leq k$. From Figure 5.6, we see that the schedule, $\sigma^*$, obtained from $\sigma^*$ by interchanging jobs $J_{z_2}$ and $J_{z_1}$, has makespan $\Delta$ less than $C_{\max}(\sigma^*)$, where

$$
\Delta = \left( \sum_{t=2}^{i} p_{z_1} + p_{z_1} + \sum_{t=i+1}^{m} p_{z_t} + [l_{z_1} - \sum_{t=i+1}^{m} p_{z_t}]^+ \right) - \left( p_{z_1} + \sum_{t=2}^{m} p_{z_t} + [l_{z_1} - \sum_{t=2}^{m} p_{z_t}]^+ \right)
$$

$$
= [l_{z_1} - \sum_{t=i+1}^{m} p_{z_t}]^+ - [l_{z_1} - \sum_{t=2}^{m} p_{z_t}]^+
$$

$$
\geq 0. \square
$$

As a result of Lemma 5.3, we can restrict our search for an optimal schedule to those schedules with job $J_{z_1}$ scheduled first.

Let $\sigma$ be any active schedule with job $J_{z_1}$ scheduled first, let $S \subseteq \{J_2, \ldots, J_k\}$ include those jobs scheduled between jobs $J_{z_1}$ and $J_{z_2}$ in $\sigma$, and let $T = \{J_2, \ldots, J_k\} \setminus S$. 

89
$S$ include those jobs scheduled between jobs $J_2$ and $\ast$ in $\sigma$. Since $l_2 = \cdots = l_k = 0$, then the order in which the jobs in $S$ are scheduled in $\sigma$ is immaterial, as is the order in which the jobs in $T$ are scheduled in $\sigma$. Now, either $\sum_{J \in S} p_j < l_x$, in which case schedule $\sigma$ includes idle time between jobs $J_x$ and $J_x$, or $\sum_{J \in S} p_j \geq l_x$, whence $\sigma$ includes no idle time between jobs $J_x$ and $J_x$. Similarly, either $\sum_{J \in T} p_j < l_x$, in which case schedule $\sigma$ includes idle time between jobs $J_x$ and $\ast$, or $\sum_{J \in T} p_j \geq l_x$, whence $\sigma$ includes no idle time between jobs $J_x$ and $\ast$. In other words, schedule $\sigma$ has one of the four forms shown in Figure 5.7.

Since

$$p_{x_1} + l_x + p_x + l_x + p_\ast$$

is a lower bound for the optimal makespan, then a "Form 1" schedule is necessarily optimal. Moreover, since

$$\sum_{J \in J} p_j$$

is a lower bound for the optimal makespan, then a "Form 3" schedule is necessarily optimal as well.

For each $j = 2, \ldots, k$, define

$$x_j = \begin{cases} 1, & \text{job } J_j \text{ is scheduled between jobs } J_x \text{ and } J_x, \\ 0, & \text{job } J_j \text{ is scheduled between jobs } J_x \text{ and } \ast. \end{cases}$$

Recall that all data are assumed to be integral and let $x^{12}$ be an optimal solution for the problem given by

$$\max \{ \sum_{j=2}^{k} p_j x_j : \sum_{j=2}^{k} p_j x_j \leq l_x - 1, \ x \in B^{k-1} \}. \quad (5.2)$$

Problem (5.2) is a special case of the 0-1 knapsack problem known as "subset sum" since, for each variable, the objective and constraint coefficients are the same. If
Figure 5.7: Four possible forms for any schedule with job $J_{x_1}$ scheduled first.

If $\sum_{j=2}^{k} p_j (1 - x_j^{12}) < l_{x_1}$, then the schedule, $\sigma^{12}$, defined by $x^{12}$ is a Form 1 schedule.

Otherwise, $\sigma^{12}$ is a Form 2 schedule with minimum idle time between jobs $J_{x_1}$ and $J_{x_2}$, and hence with minimum makespan among Form 2 schedules.

Let $x^{34}$ be an optimal solution to

$$\min\{\sum_{j=2}^{k} p_j x_j : \sum_{j=2}^{k} p_j x_j \geq l_{x_1}, \ x \in B^{k-1}\}.$$  

Equivalently, $x^{34}$ is the 0-1 complement of an optimal solution for the subset sum problem given by

$$\max\{\sum_{j=2}^{k} p_j x_j : \sum_{j=2}^{k} p_j x_j \leq \sum_{j=2}^{k} p_j - l_{x_1}, \ x \in B^{k-1}\}. \quad (5.3)$$

If $\sum_{j=2}^{k} p_j (1 - x_j^{34}) \geq l_{x_1}$, then the schedule, $\sigma^{34}$, defined by $x^{34}$ is a Form 3 schedule.

Otherwise, $\sigma^{34}$ is a Form 4 schedule with minimum idle time between jobs $J_{x_1}$ and
* and hence with minimum makespan among Form 4 schedules.

In order to solve the special case of $1 / \min$ delays, $k \geq 2, 1, \ldots, 1$-chains / $C_{\max}$ with $l_2 = \cdots = l_k = 0$, we first solve the subset sum problems (5.2) and (5.3). If $\sigma^{12}$ ($\sigma^{34}$) is a Form 1 (3) schedule, then $\sigma^{12}$ ($\sigma^{34}$) is optimal. Otherwise, the schedule with minimum makespan between $\sigma^{12}$ and $\sigma^{34}$ is optimal. Problems (5.2) and (5.3) can be solved by dynamic programming in time $O(k \cdot l_{x_1}^2)$ and $O(k(\sum_{j=2}^{k} p_j - l_{x_1})^2)$, respectively (see Nemhauser and Wolsey [19] for example). Thus, the NP-hard special case with $l_2 = \cdots = l_k = 0$ can be solved in pseudo-polynomial time.

5.1.4 Heuristic with Worst Case Performance Ratio $\frac{5}{4} + \Delta$ for NP-hard Special Case

As an alternative, we could solve (5.2) and (5.3) using the fully polynomial approximation scheme described in Lawler [17] for the subset sum problem. An algorithm for a minimization (maximization) problem is said to be a fully polynomial approximation scheme for that problem if, for any $\varepsilon \in (0,1)$ the algorithm satisfies

1. for any problem instance, the worst case performance ratio (see page 82) is at most $1 + \varepsilon$ (at least $1 - \varepsilon$) and

2. the running time of the algorithm is polynomial in the input length and in $\frac{1}{\varepsilon}$.

We now show how to use Lawler's fully polynomial approximation scheme for the subset sum problem in conjunction with a particular 0-1 knapsack heuristic to produce a heuristic algorithm with a worst case performance ratio of $\frac{5}{4} + \Delta$ for any $\Delta \in (0,\frac{1}{4})$ for an even more specific but still NP-hard special case of $1 / \min$ delays, $k \geq 2, 1, \ldots, 1$-chains / $C_{\max}$.

This special case, with $l_2 = \cdots = l_k = 0$, $p_{x_1} = p_{x_2} = p_* = 0$, $\sum_{j=2}^{k} p_j$ divisible by 2, and $l_{x_1} = l_{x_2} = \frac{1}{2} \sum_{j=2}^{k} p_j$, is illustrated in Figure 5.8. The significance of $l_{x_1} = l_{x_2} = \frac{1}{2} \sum_{j=2}^{k} p_j$ is that in any schedule, either the sum of the processing requirements of the jobs from $\{J_2, \ldots, J_k\}$ scheduled between jobs $J_{x_1}$ and $J_{x_2}$ is $l_{x_1}$ or greater, or the sum of the processing requirements of the jobs from $\{J_2, \ldots, J_k\}$...
scheduled between jobs $J_{x_2}$ and $*$ is $l_{x_2}$ or greater. Due to symmetry, we can assume, without loss of generality, that the latter holds, which implies that the sum of the processing requirements of the jobs from $\{J_2, \ldots, J_k\}$ scheduled between jobs $J_{x_1}$ and $J_{x_2}$ is $l_{x_1}$ or less. Hence, this special case reduces to solving the subset sum problem given by

$$\max\left\{\sum_{j=2}^{k} p_j x_j : \sum_{j=2}^{k} p_j x_j \leq l_{x_1}, x \in B^{k-1}\right\}, \quad (5.4)$$

where, for $j = 2, \ldots, k$,

$$x_j = \begin{cases} 1, & \text{job } J_j \text{ is scheduled between jobs } J_{x_1} \text{ and } J_{x_2} \\ 0, & \text{job } J_j \text{ is scheduled between jobs } J_{x_2} \text{ and } *. \end{cases}$$

We could, of course, solve problem (5.4) by dynamic programming in time $O(k \cdot l_{x_1}^2)$. Instead, we will solve (5.4) using Lawler's fully polynomial approximation scheme, with $\varepsilon$ determined as a function of the greedy heuristic solution of (5.4).
To solve problem (5.4) using the greedy heuristic, we proceed as follows.

**Initialization:** Sort jobs $J_2, \ldots, J_k$ such that $p_{e_2} \geq \cdots \geq p_{e_k}$; $b \leftarrow l_{e_1}$.

**For** $j = 2, \ldots, k$ **do**

If $b \geq p_{e_j}$, then $x_{e_j} = 1$ and $b \leftarrow b - p_{e_j}$; otherwise, $x_{e_j} = 0$.

As a preliminary step, we prove that the greedy heuristic fills the knapsack defined by (5.4) more than half full.

**Proposition 5.4** The greedy heuristic solution, $x^g$, to problem (5.4) satisfies

$$\sum_{j=2}^{k} p_j x^g_{e_j} > \frac{1}{2} l_{e_1} = \frac{1}{4} \sum_{j=2}^{k} p_j.$$  

**Proof:** We can, without loss of generality, assume that $p_{e_2} \leq l_{e_1}$, since otherwise, the solution $x^*$ defined by

$$x^*_{e_j} = \begin{cases} 0, & \text{if } j = 2 \\ 1, & \text{otherwise} \end{cases}$$

is optimal for (5.4) and the corresponding schedule is optimal for the special case illustrated in Figure 5.8. If $p_{e_2} > \frac{1}{4} \sum_{j=2}^{k} p_j$, then we are done, so suppose that $p_{e_2} \leq \frac{1}{4} \sum_{j=2}^{k} p_j$. Let $\delta$ be the smallest index such that

$$p_{e_2} + \cdots + p_{e_\delta} > \frac{1}{4} \sum_{j=2}^{k} p_j.$$  

By definition of $\delta$,

$$p_{e_2} + \cdots + p_{e_{\delta-1}} \leq \frac{1}{4} \sum_{j=2}^{k} p_j.$$  

94
Since \( p_{e_8} \leq p_{e_2} \leq \frac{1}{4} \sum_{j=2}^{k} p_j \), then

\[
p_{e_2} + \cdots + p_{e_{s-1}} + p_{e_s} \leq \frac{1}{4} \sum_{j=2}^{k} p_j + \frac{1}{4} \sum_{j=2}^{k} p_j = \frac{1}{2} \sum_{j=2}^{k} p_j.
\]

Thus, the solution \( x' \) defined by

\[
x'_{e_j} = \begin{cases} 
1, & \text{if } j = 2, \ldots, \delta \\
0, & \text{otherwise}
\end{cases}
\]

is feasible for (5.4). Consequently, \( x'_{e_j} \) must equal 1 for \( j = 2, \ldots, \delta \) and

\[
\sum_{j=2}^{k} p_j x'_{e_j} \geq \sum_{j=2}^{k} p_j x'_j = p_{e_2} + \cdots + p_{e_s} > \frac{1}{4} \sum_{j=2}^{k} p_j. \quad \Box
\]

We now present and establish the worst case performance ratio of a heuristic algorithm for the special case of \( \frac{1}{\text{min delays}}, k \geq 2, 1, \ldots, 1 \)-chains / \( C_{\text{max}} \) illustrated in Figure 5.8.

**Proposition 5.5** The following heuristic algorithm for the special case of \( \frac{1}{\text{min delays}}, k \geq 2, 1, \ldots, 1 \)-chains / \( C_{\text{max}} \) with \( l_2 = \cdots = l_k = 0, p_{e_1} = p_{e_2} = p_* = 0, \) and \( l_{e_1} = l_{e_2} = \frac{1}{2} \sum_{j=2}^{k} p_j \) has a worst case performance ratio of \( \frac{5}{4} + \Delta \) for any \( \Delta \in (0, \frac{1}{4}) \).

**Heuristic Algorithm**

**Step 1:** Solve problem (5.4) using the greedy heuristic. Select \( \Delta \in (0, \frac{1}{4}) \) and let

\[
\varepsilon = 1 - \left( \frac{1}{4} - \Delta \right) \frac{\sum_{j=2}^{k} p_j}{\sum_{j=2}^{k} p_j x'_j}.
\]
Step 2: Solve problem (5.4) using Lawler's fully polynomial approximation scheme with $\varepsilon$ from Step 1.

Proof: By Proposition 5.4,

$$
\varepsilon = 1 - \left(\frac{1}{4} - \Delta\right)\frac{\sum_{j=2}^{k} p_j}{\sum_{j=2}^{k} p_j x_j^*} > 1 - \left(\frac{1}{4} - \Delta\right)4 = 4\Delta > 0.
$$

Thus, Step 2 can be accomplished. Let $x^\varepsilon$ be the heuristic solution and let $x^*$ be an optimal solution for (5.4). The makespans of the schedules corresponding to $x^\varepsilon$ and $x^*$ are $\frac{1}{2} \sum_{j=2}^{k} p_j + \sum_{j=2}^{k} p_j (1 - x_j^*)$ and $\frac{1}{2} \sum_{j=2}^{k} p_j + \sum_{j=2}^{k} p_j (1 - x_j^*)$, respectively. Now

$$
\frac{1}{2} \sum_{j=2}^{k} p_j + \sum_{j=2}^{k} p_j (1 - x_j^*) = \frac{\frac{3}{2} \sum_{j=2}^{k} p_j - \frac{1}{2} \sum_{j=2}^{k} p_j x_j^*}{\sum_{j=2}^{k} p_j} \leq \frac{\frac{3}{2} \sum_{j=2}^{k} p_j - \frac{1}{2} \sum_{j=2}^{k} p_j x_j^*}{\sum_{j=2}^{k} p_j} = \frac{3}{2} \frac{\sum_{j=2}^{k} p_j x_j^*}{\sum_{j=2}^{k} p_j}
$$

$$
\leq \frac{3}{2} \frac{(1 - \varepsilon) \sum_{j=2}^{k} p_j x_j^*}{\sum_{j=2}^{k} p_j} = \frac{3}{2} - \frac{1 - (1 - (\frac{1}{4} - \Delta)) \sum_{j=2}^{k} p_j x_j^*}{\sum_{j=2}^{k} p_j} \leq \frac{3}{2} - \frac{1 - (\frac{1}{4} - \Delta)}{\sum_{j=2}^{k} p_j x_j^*} = \frac{3}{2} - \frac{1 - \Delta}{\sum_{j=2}^{k} p_j x_j^*}.
$$

96
where the first inequality follows since $x^\ast$ is feasible for (5.4), the second inequality follows since

$$\frac{\sum_{j=2}^{k} p_j x_j^\ast}{\sum_{j=2}^{k} p_j x^\ast_j} \geq (1 - \epsilon),$$

and the third inequality follows since

$$\sum_{j=2}^{k} p_j x_j^\ast \geq \sum_{j=2}^{k} p_j x_j^\ast.$$ \(\square\)

Solving problem (5.4) using the greedy heuristic requires time $O(k \log k)$. Solving (5.4) using Lawler's fully polynomial approximation scheme requires time $O(k + \frac{1}{\epsilon^3})$ in general. Since $\epsilon > 4 \Delta$, then $\frac{1}{\epsilon^3} < \frac{1}{(4 \Delta)^3}$. Thus, the heuristic requires time $O(k \log k + \frac{1}{\Delta^3})$ overall.

The proof of Proposition 5.5 goes through with $\Delta = 0$. Unfortunately, though, the heuristic with worst case performance ratio $\frac{5}{4}$ has running time $O(k \log k + \frac{1}{\epsilon^3})$, where $\epsilon$ depends only on the problem data. As the following series of examples shows, $\epsilon$ can go to 0, at least for small values of $k$. Let $k = 5$, $p_2 = 3s + 2$, and $p_3 = p_4 = p_5 = 3s$. Then, $\sum_{j=2}^{5} p_j = 12s + 2$ and $\sum_{j=2}^{5} p_j x_j^\ast = p_2 = 3s + 2$. Thus,

$$\lim_{s \to \infty} \frac{\sum_{j=2}^{5} p_j}{\sum_{j=2}^{5} p_j x_j^\ast} = \lim_{s \to \infty} \frac{12s + 2}{3s + 2} = 4,$$

which implies

$$\lim_{s \to \infty} \epsilon = \lim_{s \to \infty} 1 - \left(\frac{1}{4}\right)4 = 0.$$

By selecting $\Delta > 0$, we bound $\epsilon$ away from 0 and thus establish control over the heuristic's running time.
Proposition 5.5 can be generalized to the special case with $l_2 = \cdots = l_k = 0$, $p_{x_1} = p_{x_2} = p_* = 0$, $l_{x_1} = \alpha \sum_{j=2}^{k} p_j$, and $l_{x_2} = (1 - \alpha) \sum_{j=2}^{k} p_j$, where $\alpha \in [\frac{1}{2}, 1]$ is such that both $l_{x_1}$ and $l_{x_2}$ are integers. For this special case, we execute the heuristic algorithm twice with the same choice of $\Delta$, first solving (5.4) and then solving

$$\max \left\{ \sum_{j=2}^{k} p_j (1 - x_j) : \sum_{j=2}^{k} p_j (1 - x_j) \leq l_{x_2}, x \in B^{k-1} \right\}. \tag{5.5}$$

We can assume, without loss of generality, that the greedy heuristic solution of problem (5.5) has value greater than zero (so that $\epsilon$ is defined) since, otherwise, $x_j = 0$ for $j = 2, \ldots, k$ is optimal for (5.5). We select between the schedules corresponding to the heuristic solutions of (5.4) and (5.5) the schedule with smaller makespan. The ratio of the makespan of this schedule to the optimal makespan is at most $1 + \frac{\alpha}{2} + \Delta$. This heuristic can also be accomplished in time $O(k \log k + \frac{1}{\Delta \gamma})$ provided that the greedy heuristic fills the knapsack defined by (5.5) at least half full.

### 5.2 $\min 1 / \min$ delays, $2 n_1, n_2$-chains / $C_{\text{max}}$

#### 5.2.1 No Schedule Has Makespan Greater Than Twice Optimal

Interestingly enough, every heuristic for $1 / \min$ delays, $2 n_1, n_2$-chains / $C_{\text{max}}$ has a worst case performance ratio of at most 2 since, as we now prove, no schedule has makespan greater than twice the optimal makespan.

**Proposition 5.6** Let $\sigma^*$ be any optimal schedule and let $\sigma$ be an arbitrary schedule. Then $C_*(\sigma) \leq 2C_*(\sigma^*)$.

**Proof:** Assuming that $p_* = 0$, an upper bound for $C_*(\sigma)$ is given by

$$C^u = \sum_{i=1}^{n_1} (p_{1,i} + l_{1,i}) + \sum_{j=1}^{n_2} (p_{2,j} + l_{2,j})$$
and a lower bound for $C_*(\sigma^*)$ is given by

$$C' = \max\{\sum_{i=1}^{n_1}(p_{1,i} + l_{1,i}), \sum_{j=1}^{n_2}(p_{2,j} + l_{2,j})\}.$$  

Now,

$$\frac{C_*(\sigma)}{C_*(\sigma^*)} \leq \frac{C'}{C'} \leq \frac{C'}{C'} \leq 2. \quad \Box$$

### 5.2.2 Polynomially Solvable Special Cases

As mentioned in Chapter 4, the number of feasible sequences for 1 / min delays, 2 $n_1, n_2$-chains / $C_{\max}$ is

\[
\binom{n_1 + n_2}{n_1}.
\]

Suppose that $n_2 \leq c$, where $c$ is a fixed constant. Then

\[
\left(\begin{array}{c} n_1 + n_2 \\ n_1 \end{array}\right) \leq \left(\begin{array}{c} n_1 + c \\ n_1 \end{array}\right) = \frac{(n_1 + c)(n_1 + c - 1) \cdots (n_1 + 1)}{c!} = O(n_1^c).
\]

Since determining the schedule associated with a given sequence can be accomplished in time $O(n_1 + n_2) = O(n_1 + c) = O(n_1)$, then complete enumeration requires time $O(n_1^{c+1})$. Thus, the subset of instances of 1 / min delays, 2 $n_1, n_2$-chains / $C_{\max}$ with $n_2 \leq c$ can be solved in polynomial time.

Recall from Chapter 4 that 1 / min delays, 2 $n_1, n_2$-chains / $C_{\max}$ can be solved by dynamic programming in time $O(n_1 n_2^2(1 + [L_2 - p_1])^+)$, where $L_2 = \max_{j=1,\ldots,n_2} l_{2,j}$ and $p_1 = \min_{i=2,\ldots,n_1} p_{1,i}$. As an immediate consequence, the subset of instances with $L_2 - p_1$ bounded by a polynomial in $n_1$ and $n_2$ can be solved in polynomial time. Reversing the roles of the 1-jobs and the 2-jobs, we can solve 1 / min delays, 2 $n_1, n_2$-chains / $C_{\max}$ in time $O(n_1^2 n_2(1 + [L_2 - p_2])^+)$, where $L_1 = \max_{i=1,\ldots,n_1} l_{1,i}$ and $p_2 = \min_{j=2,\ldots,n_2} p_{2,j}$. Thus, the subset of instances with $L_1 - p_2$ bounded by a polynomial in $n_1$ and $n_2$ can also be solved in polynomial time.
We now show that, assuming \( n_1 \geq n_2 \), the subset of instances of \( 1 / \text{min delays} \), \( 2 n_1, n_2 \)-chains / \( C_{\text{max}} \) with \( L_2 \leq p_1 \) and \( L_1 \leq p_2 \) can be solved in time \( O(n_1 \lg n_1) \). The significance of \( L_2 \leq p_1 \) (\( L_1 \leq p_2 \)) is that if a 2-job (1-job) is sequenced immediately after a 1-job (2-job), then the corresponding active schedule includes no idle time between the end of the 1-job (2-job) and the beginning of the 1-job (2-job).

As described in Chapter 4, we might imagine there are \( n_1 + 1 \) bins, one before job \( J_{1,1} \), one between jobs \( J_{1,i} \) and \( J_{1,i+1} \) for \( i = 1, \ldots, n_1 - 1 \), and one between jobs \( J_{1,n_1} \) and \( * \), into which the 2-jobs are placed (see Figure 5.9). Let \( I_i \), the size of bin \( i \), be the amount of idle time immediately preceding job \( J_{1,i} \) in any schedule corresponding to a sequence with bin \( i \) empty for \( i = 1, \ldots, n_1 \). Let \( I_{n_1+1}^- \) (\( I_{n_1+1}^+ \)) be the amount of idle time immediately preceding job \( * \) in any schedule corresponding to a sequence with bin \( n_1 + 1 \) empty (not empty). Then

\[
I_1 = 0, \\
I_i = l_{1,i-1} \text{ for } i = 2, \ldots, n_1, \\
I_{n_1+1}^- = l_{1,n_1}, \text{ and} \\
I_{n_1+1}^+ = l_{2,n_2}.
\]

Let \( \sigma^1 \) be the schedule corresponding to the sequence with bin \( n_1 + 1 \) empty and jobs \( J_{2,1}, \ldots, J_{2,n_2} \) placed, one each, in the \( n_2 \) largest of bins \( 1, \ldots, n_1 \). Let \( \sigma^2 \) be the schedule corresponding to the sequence with job \( J_{2,n_2} \) placed in bin \( n_1 + 1 \) and jobs \( J_{2,1}, \ldots, J_{2,n_2-1} \) placed, one each, in the \( n_2 - 1 \) largest of bins \( 1, \ldots, n_1 \). We now prove that this special case is solved by selecting between \( \sigma^1 \) and \( \sigma^2 \) the schedule with smaller makespan.

**Proposition 5.7** Let \( \sigma^* = \arg\min_{\sigma = \sigma^1, \sigma^2} \{C_*(\sigma)\} \). Then, schedule \( \sigma^* \) is optimal for the special case of \( 1 / \text{min delays} \), \( 2 n_1, n_2 \)-chains / \( C_{\text{max}} \) with \( L_2 \leq p_1 \) and \( L_1 \leq p_2 \).

**Proof:** If bin \( n_1 + 1 \) is empty, then at least \( n_1 - n_2 \) of bins \( 1, \ldots, n_1 \) must also be empty. Thus, the total idle time in any schedule corresponding to a sequence with
bin $n_1 + 1$ empty is at least $I'$, where $I'$ equals $I_{n_1+1}^-$ plus the sum of the $n_1 - n_2$ smallest of $I_1,\ldots, I_{n_1}$. Since schedule $\sigma^1$ has total idle time equal to $I'$, then $\sigma^1$ has minimum makespan among schedules corresponding to sequences with bin $n_1 + 1$ empty.

On the other hand, if bin $n_1 + 1$ is not empty, then at least $n_1 - (n_2 - 1) = n_1 - n_2 + 1$ of bins $1,\ldots, n_1$ must be empty. Thus, the total idle time in any schedule corresponding to a sequence with bin $n_1 + 1$ not empty is at least $I''$, where $I''$ equals $I_{n_1+1}^+$ plus the sum of the $n_1 - n_2 + 1$ smallest of $I_1,\ldots, I_{n_1}$. Since schedule $\sigma^2$ has total idle time equal to $I''$, then $\sigma^2$ has minimum makespan among schedules corresponding to sequences with bin $n_1 + 1$ not empty.

In the sequence associated with any schedule, bin $n_1 + 1$ is either empty or not empty. Thus, if $I' < I''$, then schedule $\sigma^1$ is optimal and otherwise, schedule $\sigma^2$ is optimal. □

Sorting $I_1,\ldots, I_{n_1}$ in nondecreasing order and identifying the $n_2$ largest of these values requires time $O(n_1 \lg n_1)$. Since computing the start time of each job in the schedule corresponding to a given sequence can be accomplished in time $O(n_1 + n_2)$, then this special case can be solved in time $O(n_1 \lg n_1)$ overall.

### 5.2.3 Disjunctive Graph Representation

In this subsection, we present a technique for representing instances of $1/\min$ delays, $2\ n_1,n_2$-chains / $C_{max}$ which is adapted from the disjunctive graph of Roy and Sussmann as described in [18]. We associate with each instance a weighted,
mixed graph $H = (V, A, E)$. The vertex set $V = V_1 \cup V_2$, where

$$V_1 = \{v_0\} \cup \{v_{1,i}^p, i = 1, \ldots, n_1\} \cup \{v_{2,j}^p, j = 1, \ldots, n_2\}$$

and

$$V_2 = \{v_{1,i}^l, i = 1, \ldots, n_1\} \cup \{v_{2,j}^l, j = 1, \ldots, n_2\}.$$ 

Vertex $v_0$ corresponds to a dummy initial job with zero processing requirement. The remaining vertices in $V_1$ correspond to jobs in $J$. The vertices in $V_2$ correspond to the minimum delays. We assign to each vertex a weight equal to the duration of the corresponding job or delay. Arc set $A = A_1 \cup A_2 \cup A_3 \cup A_4$, where

$$A_1 = \{< v_0, v_{1,1}^p >, < v_0, v_{2,1}^p >\},$$

$$A_2 = \{< v_{1,i}^p, v_{1,i}^l >, i = 1, \ldots, n_1\} \cup \{< v_{1,i}^p, v_{1,i+1}^p >, i = 1, \ldots, n_1 - 1\},$$

$$A_3 = \{< v_{2,j}^p, v_{2,j}^l >, j = 1, \ldots, n_2\} \cup \{< v_{2,j}^p, v_{2,j+1}^p >, j = 1, \ldots, n_2 - 1\},$$

and

$$A_4 = \{< v_{1,n_1}^p, \ast >, < v_{2,n_2}^p, \ast >\}.$$ 

These arcs represent precedence constraints among jobs and minimum delays. The edge set $E$ is given by

$$E = \{(v_{1,i}^p, v_{2,j}^p), i = 1, \ldots, n_1, j = 1, \ldots, n_2\}.$$ 

These edges represent machine capacity constraints. Figure 5.11 shows the weighted, mixed graph $H$ associated with the instance shown in Figure 5.10.

The edges in $E$ join vertices corresponding to pairs of jobs, either one of which is allowed to precede the other. By orienting an edge in one way or the other, we specify the relative order of the pair of jobs corresponding to that edge’s endpoints. By orienting all edges such that the resultant directed graph, $\tilde{H}$, is acyclic, we specify a feasible sequence. The completion time of job $J_c \in J$ in the schedule associated with this sequence is equal to the weight of the maximum weight path in $\tilde{H}$ from
Figure 5.10: Instance of 1 / min delays, 2 $n_1, n_2$-chains / $C_{\text{max}}$.

Figure 5.11: Graph $H$ associated with instance of 1 / min delays, 2 $n_1, n_2$-chains / $C_{\text{max}}$. 
v_0 to the vertex corresponding to J. By weight of a path, we mean the sum of the vertex weights over all vertices in the path. Thus, 1 / \text{min delays, 2 } n_1, n_2\text{-chains } C_{\text{max}} \text{ can be thought of as the problem of finding an orientation of the edges in } E \text{ such that the resultant directed graph is acyclic with maximum weight path from } v_0 \text{ to vertex } * \text{ of minimum weight among all such acyclic directed graphs.}

Using the weighted, mixed graph, we can solve 1 / \text{min delays, 2 } n_1, n_2\text{-chains } C_{\text{max}} \text{ by branch-and-bound as follows. For each problem in which not all edges of the corresponding graph have been oriented, we select an unoriented edge } (v_{1,i}, v_{2,j}) \text{ for some } i \in \{1, \ldots, n_1\} \text{ and } j \in \{1, \ldots, n_2\} \text{ and we consider two subproblems, one with } (v_{1,i}, v_{2,j}) \text{ oriented from } v_{1,i} \text{ to } v_{2,j} \text{ and the other with } (v_{1,i}, v_{2,j}) \text{ oriented from } v_{2,j} \text{ to } v_{1,i}. \text{ For each subproblem, we compute a lower bound for the makespan of any schedule which could be gotten by orienting the remaining edges of the corresponding graph. We eliminate a subproblem from further consideration if this lower bound exceeds a known upper bound for the optimal makespan. One such lower bound is given by the weight of a maximum weight directed path from } v_0 \text{ to vertex } * \text{ in the partially oriented graph.}

If we orient edge } (v_{1,i}, v_{2,j}) \text{ from } v_{1,i} \text{ to } v_{2,j}, \text{ then, in order to ensure that the graph remains acyclic, we must orient edge } (v_{1,i}, v_{2,m}) \text{ from } v_{1,i} \text{ to } v_{2,m} \text{ for } m = j+1, \ldots, n_2. \text{ Observe, though, that the weight of a maximum weight path from } v_0 \text{ to } v_{2,m} \text{ which includes edge } (v_{1,i}, v_{2,j}) \text{ and arcs } < v_{2,j}, v_{2,j} >, < v_{2,j}, v_{2,j+1} >, < v_{2,j+1}, v_{2,j+1} >, \ldots, < v_{2,m-1}, v_{2,m} > \text{ is at least}

\[
\sum_{r=j}^{m-1} \left( p_{2,r} + l_{2,r} \right)
\]

more than the weight of a maximum weight path from } v_0 \text{ to } v_{2,m} \text{ which includes edge } (v_{1,i}, v_{2,m}). \text{ In other words, no maximum weight path from } v_0 \text{ to } v_{2,m} \text{ and hence no maximum weight path from } v_0 \text{ to vertex } * \text{ includes } (v_{1,i}, v_{2,m}). \text{ Thus, having oriented edge } (v_{1,i}, v_{2,j}) \text{ from } v_{1,i} \text{ to } v_{2,j}, \text{ we can eliminate edge } (v_{1,i}, v_{2,m}) \text{ for}
m = j + 1, . . . , n_2. By the same token, if we orient edge \((v_{i,i}^p, v_{i,j}^p)\) from \(v_{i,i}^p\) to \(v_{i,j}^p\), then we can eliminate edge \((v_{i,c}^p, v_{i,j}^p)\) for \(c = i + 1, . . . , n_1\). By eliminating those edges with orientations implied by transitivity, we ensure that the number of arcs and oriented edges and hence the amount of time required to compute the weight of a maximum weight path from \(v_0\) to vertex \(*\) is \(O(n_1 + n_2)\).

The weighted, mixed graph can be generalized in an obvious manner to represent 1 / min delays, \(k\) \(n_1, . . . , n_k\)-chains / \(C_{\text{max}}\). Our success in using this graph to solve 1 / min delays, \(k\) \(n_1, . . . , n_k\)-chains / \(C_{\text{max}}\) by branch-and-bound depends on our ability to generate quality lower bounds for subproblem makespan and upper bounds for the optimal makespan.

### 5.2.4 Lower Bounds

We now present three lower bounds for the optimal makespan of 1 / min delays, 2 \(n_1, n_2\)-chains / \(C_{\text{max}}\) under the assumption \(p_* = 0\). Let \(\sigma^*\) be an optimal schedule. Two obvious lower bounds for \(C_{\text{max}}(\sigma^*) = C_*(\sigma^*)\) are given by

\[
\sum_{J_\epsilon \in J} p_\epsilon
\]

and

\[
\max\{\sum_{i=1}^{n_1}(p_{1,i} + l_{1,i}), \sum_{j=1}^{n_2}(p_{2,j} + l_{2,j})\}.
\]

We refer to these as the processing requirement and longest chain bounds, respectively. The class we now describe includes lower bounds for \(C_*(\sigma^*)\) which are as large as either of these bounds. The following discussion is adapted from Carlier [5].

Obviously, job \(J_{1,i}\) can start no earlier than time

\[
r_{1,i} = \sum_{i=1}^{i-1}(p_{1,i} + l_{1,i})
\]

105
for each \( i = 1, \ldots, n_1 \). Moreover, at least
\[
q_{1,i} = l_{1,i} + \sum_{t=i+1}^{n_1} (p_{1,i} + l_{1,i})
\]
time units must elapse between the end of job \( J_{1,i} \) and the beginning of job \( * \) for \( i = 1, \ldots, n_1 \). We refer to \( r_{1,i} \) and \( q_{1,i} \) as the release date and tail, respectively, of job \( J_{1,i} \) for \( i = 1, \ldots, n_1 \). In a similar manner, we can define \( r_{2,j} \) and \( q_{2,j} \), the release date and tail of job \( J_{2,j} \) for \( j = 1, \ldots, n_2 \). The following proposition defines a class of lower bounds for \( C_*(\sigma^*) \) in terms of these release dates and tails.

**Proposition 5.8** For all \( S \subseteq J \setminus \{ * \} \),
\[
h(S) = \min_{J_e \in S} r_e + \sum_{J_e \in S} p_e + \min_{J_e \in S} q_e \leq C_*(\sigma^*).
\]

**Proof:** Let \( S \subseteq J \setminus \{ * \} \), \( J_m = \arg\min_{J_e \in S} \{ \sigma^*(e) \} \), and \( J_u = \arg\max_{J_e \in S} \{ \sigma^*(e) \} \).

Then
\[
\sigma^*(m) \geq r_m \geq \min_{J_e \in S} r_e,
\]
\[
C_u(\sigma^*) - \sigma^*(m) \geq \sum_{J_e \in S} p_e,
\]

and
\[
C_*(\sigma^*) - C_u(\sigma^*) \geq q_u \geq \min_{J_e \in S} q_e.
\]

Summing these three inequalities gives the result. \( \square \)

Note that
\[
h(J \setminus \{ * \}) = \sum_{J_e \in J} p_e + \min\{q_{1,n_1}, q_{2,n_2}\}
\]
\[
= \sum_{J_e \in J} p_e + \min\{l_{1,n_1}, l_{2,n_2}\}
\]
\[
\geq \sum_{J_e \in J} p_e.
\]

106
Furthermore,

\[ \max \{ h(\{ J_{1,1} \} ), h(\{ J_{2,1} \} ) \} = \max \{ \sum_{i=1}^{n_1} (p_{1,i} + l_{1,i}), \sum_{j=1}^{n_2} (p_{2,j} + l_{2,j}) \}. \]

Thus, the class of lower bounds for \( C_*(\sigma^*) \) defined by Proposition 5.8 includes bounds as large as either the processing requirement or the longest chain bound.

Determining the largest of the lower bounds defined by Proposition 5.8 seemingly involves evaluating \( h(S) \) for each \( S \subseteq J \setminus \{ * \} \). As the following proposition reveals, \( \max_{S \subseteq J \setminus \{ * \}} \{ h(S) \} \) can in fact be determined with relative ease. The Schrage algorithm involves scheduling next the available job with largest tail, with ties broken arbitrarily. The preemptive version of this algorithm also involves stopping the processing of a job if another job with larger tail becomes available.

**Proposition 5.9** The makespan of the schedule generated using the preemptive version of the Schrage algorithm equals \( \max_{S \subseteq J \setminus \{ * \}} \{ h(S) \} \).

**Proof:** We only sketch the proof here. For complete details, see the proof of Proposition 3 in Carlier [5].

Let \( V \) be the makespan of the schedule generated using the preemptive version of the Schrage algorithm. That \( V \geq \max_{S \subseteq J \setminus \{ * \}} \{ h(S) \} \) follows since, for all \( S \subseteq J \setminus \{ * \} \), \( h(S) \) is a lower bound for the makespan of any preemptive schedule. Showing that \( V \leq \max_{S \subseteq J \setminus \{ * \}} \{ h(S) \} \) involves exhibiting a subset \( S_0 \subseteq J \setminus \{ * \} \) such that \( h(S_0) = V \). Such a subset \( S_0 \) can be identified by applying the Schrage algorithm to a modified problem instance obtained by replacing each job \( J_{1,i} (J_{2,j}) \) by \( p_{1,i} (p_{2,j}) \) new jobs, each with a unit processing requirement, a release date of \( r_{1,i} (r_{2,j}) \), and a tail of \( q_{1,i} (q_{2,j}) \) for \( i = 1, \ldots, n_1 \) (\( j = 1, \ldots, n_2 \)) and then invoking the main theorem of [5]. □

We now demonstrate the three lower bounds described in this subsection using the instance shown in Figure 5.10. The processing requirement and longest chain
Table 5.1: Release dates, processing requirements, and tails.

<table>
<thead>
<tr>
<th>e</th>
<th>r_e</th>
<th>p_e</th>
<th>q_e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>0</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>1,2</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>1,3</td>
<td>9</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2,1</td>
<td>0</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>2,2</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 5.12: Schedule generated using the preemptive version of the Schrage algorithm.

bounds for this instance are 12 and 15, respectively. The schedule shown in Figure 5.12 is the result of applying the preemptive version of the Schrage algorithm using the data given in Table 5.1. This schedule has a makespan of 17. Observe that

\[ h(\{J_{1,2}, J_{1,3}, J_{2,2}\}) = 3 + 10 + 4 = 17. \]

Observe also that, although in general, the schedule generated using the preemptive version of the Schrage algorithm includes preemptions, the schedule shown in Figure 5.12 is nonpreemptive, feasible, and hence optimal.

5.2.5 Bicriterion Heuristic

In Section 4.1, we defined \( f_t \) for each \( t = 1, \ldots, n+1 \) and each feasible combination of \( x_1, \ldots, x_{t-1} \), and \( x_t \) to be the difference between the completion times of jobs \( J_{1,t} \) and \( J_{1,t-1} \) in the schedule associated with the sequence obtained by appending
jobs $J_{2,s_{t-1}+1}, \ldots, J_{2,s_{t-1}+x_t}$, and $J_{1,t}$, in that order, to the end of the sequence

$$J_{2,0} \rightarrow J_{1,0} \rightarrow J_{2,1} \rightarrow \cdots \rightarrow J_{2,x_t} \rightarrow J_{1,1} \rightarrow J_{2,x_t+1} \rightarrow \cdots \rightarrow J_{2,x_t+x_2} \rightarrow J_{1,2}$$

$$\rightarrow \cdots \rightarrow J_{1,t-2} \rightarrow J_{2,x_t+1+x_{t-2}+1} \rightarrow \cdots \rightarrow J_{2,s_{t-1}} \rightarrow J_{1,t-1},$$

where $s_{t-1} = \sum_{j=1}^{t-1} x_j \in \{0,1,\ldots,n_2\}$ and $x_t \in \{0,1,\ldots,n_2 - s_{t-1}\}$. In this subsection, we give formulas for $f_t^1$, a lower bound, and $f_t^2$, an upper bound for $f_t$ in terms of $s_{t-1}$ and $x_t$ for each $t = 1, \ldots, n_1 + 1$ and each feasible combination of $x_1, \ldots, x_{t-1}$, and $x_t$. Those formulas depend on $x_1, \ldots, x_{t-1}$ only through the role these variables play in determining $s_{t-1}$. Using these lower and upper bounds, we approximate $1 \big/ \min$ delays, $2 n_1, n_2$-chains / $C_{\max}$ by a bicriterion problem.

In the event $f_t$ can be expressed in terms of only $s_{t-1}$ and $x_t$, we can let $f_t^1 = f_t^2 = f_t$. Thus, for each $t = 1, \ldots, n_1$ and each feasible combination of $x_1, \ldots, x_{t-1}$, and $x_t$ with $x_t = 0$, we can let

$$f_t^1 = f_t^2 = l_{t,t-1} + p_{t,t}.$$

For each $t = 1, \ldots, n_1$ and each feasible combination of $x_1, \ldots, x_{t-1}$, and $x_t$ with $x_t \in \{1, \ldots, n_2 - s_{t-1}\}$, $f_t$ is nondecreasing in

$$g_t = \{l_{2,s_{t-1}} - (w_{t-1} + p_{1,t-1})\}^+.$$

Since

$$0 \leq g_t \leq \{l_{2,s_{t-1}} - p_{1,t-1}\}^+,$$

then, for this case, we can let

$$f_t^1 = PLP_{s_{t-1}+1,s_{t-1}+x_t} + [l_{1,t-1} - PLP_{s_{t-1}+1,s_{t-1}+x_t}]^+ + p_{1,t}$$

and

$$f_t^2 = \{l_{2,s_{t-1}} - p_{1,t-1}\}^+ + PLP_{s_{t-1}+1,s_{t-1}+x_t}$$

$$+ [l_{1,t-1} - (\{l_{2,s_{t-1}} - p_{1,t-1}\}^+ + PLP_{s_{t-1}+1,s_{t-1}+x_t})]^+ + p_{1,t}.$$
By a similar argument, we can justify letting

\[ f_{n_1+1}^1 = l_{1,n_1} + p \]

and

\[ f_{n_1+1}^2 = \max\{l_{1,n_1}, [l_{2,n_2} - p_{1,n_1}]^+\} + p \]

for each feasible combination of \( x_1, \ldots, x_{n_1}, \) and \( x_{n_1+1} \) such that \( s_{n_1} = \sum_{j=1}^{n_1} x_j = n_2 \) and letting

\[ f_{n_1+1}^1 = \text{PLP}_{s_{n_1}+1,n_2} + \max\{[l_{1,n_1} - \text{PLP}_{s_{n_1}+1,n_2}]^+, l_{2,n_2}\} + p \]

and

\[ f_{n_1+1}^2 = \{l_{2,s_{n_1}} - p_{1,n_1}\}^+ + \text{PLP}_{s_{n_1}+1,n_2} \]

\[ + \max\{[l_{1,n_1} - ((l_{2,s_{n_1}} - p_{1,n_1})^+ + \text{PLP}_{s_{n_1}+1,n_2})]^+, l_{2,n_2}\} + p \]

for each feasible combination of \( x_1, \ldots, x_{n_1}, \) and \( x_{n_1+1} \) such that \( s_{n_1} = \sum_{j=1}^{n_1} x_j \in \{0, 1, \ldots, n_2 - 1\} \).

Let \( x'_1, \ldots, x'_{n_1+1} \) be any solution of

\[ x_1 + \cdots + x_{n_1+1} = n_2, \ x_j \in \mathbb{Z}_0^+ \text{ for } j = 1, \ldots, n_1, \tag{5.6} \]

let \( s'_0 = 0, \) and let \( s'_t = \sum_{j=1}^{t} x'_j \) for \( t = 1, \ldots, n_1. \) Then, \( \sum_{t=1}^{n_1+1} f^1_t(s'_{t-1}, x'_t) \) is a lower bound and \( \sum_{t=1}^{n_1+1} f^2_t(s'_{t-1}, x'_t) \) is an upper bound for the makespan of the schedule associated with the sequence defined by \( x'_1, \ldots, x'_{n_1+1}. \) This schedule likely has small makespan if \( \sum_{t=1}^{n_1+1} f^1_t(s'_{t-1}, x'_t) \) is small and \( \sum_{t=1}^{n_1+1} f^2_t(s'_{t-1}, x'_t) \) is not too large. A solution \( x'_1, \ldots, x'_{n_1+1} \) of (5.6) is Pareto optimal with respect to minimizing both \( \sum_{t=1}^{n_1+1} f^1_t \) and \( \sum_{t=1}^{n_1+1} f^2_t \) if there exists no other solution \( x''_1, \ldots, x''_{n_1+1} \) of (5.6) with

\[ \sum_{t=1}^{n_1+1} f^1_t(s''_{t-1}, x''_t) \leq \sum_{t=1}^{n_1+1} f^1_t(s'_{t-1}, x'_t) \]

and

\[ \sum_{t=1}^{n_1+1} f^2_t(s''_{t-1}, x''_t) \leq \sum_{t=1}^{n_1+1} f^2_t(s'_{t-1}, x'_t) \]
where at least one of these inequalities is strict. Schedules with small makespans are likely the schedules associated with feasible sequences defined by Pareto optimal solutions of

$$\min_{x_1, \ldots, x_{n_1+1}} \left\{ \sum_{t=1}^{n_1+1} f_t^1(s_{t-1}, x_t), \sum_{t=1}^{n_1+1} f_t^2(s_{t-1}, x_t) \right\} \quad (5.7)$$

\[ s_t = s_{t-1} + x_t \text{ for } t = 1, \ldots, n_1 + 1, \ s_{t-1} = 0, 1, \ldots, n_2, \text{ and} \]
\[ x_t = 0, 1, \ldots, n_2 - s_{t-1} \]
\[ s_0 = 0. \]

One interpretation of problem (5.7) is as a bicriterion shortest path problem in a certain network. The network has nodes \((t, s_t)\) for \(t = 0\) and \(s_0 = 0\), for \(t = 1, \ldots, n_1\) and \(s_t = 0, 1, \ldots, n_2,\) and for \(t = n_1 + 1\) and \(s_{n_1+1} = n_2.\) In addition, the network has an arc from node \((t-1, s_{t-1})\) to node \((t, s_{t-1} + x_t)\) with weights \(f_t^1(s_{t-1}, x_t)\) and \(f_t^2(s_{t-1}, x_t)\) for \(t = 1, \ldots, n_1 + 1\) and \(x_t = 0, 1, \ldots, n_2 - s_{t-1}\). This network is shown in Figure 5.13. Solutions \(x_1, \ldots, x_{n_1+1}\) of (5.7) correspond to paths from node \((0, 0)\) to node \((n_1 + 1, n_2)\) in the network.

Unfortunately, the number of Pareto optimal paths in a network can be exponential in the number of nodes in the network (see Hansen [12]). Thus, the problem of identifying all Pareto optimal paths is in general intractable. In the absence of a proof that the number of Pareto optimal paths for the network of Figure 5.13 with costs \(f_t^1\) and \(f_t^2\) is bounded by a polynomial in \(n_1\) and \(n_2,\) we suggest the following alternatives. The first alternative is to use a surrogate criterion for \(\min \sum_{t=1}^{n_1+1} f_t^1.\) Note that every path from node \((0, 0)\) to node \((n_1 + 1, n_2)\) includes exactly \(n_1 + 1\) arcs. Thus, instead of simultaneously minimizing \(\sum_{t=1}^{n_1+1} f_t^1\) and \(\sum_{t=1}^{n_1+1} f_t^2,\) we could minimize \(\sum_{t=1}^{n_1+1} f_t^1\) and \(\max_{t=1, \ldots, n_1+1} f_t^2.\) Hansen [12] presents a simple transformation from “MINSUM-MINMAX” to “MINSUM-MAXMIN” and an algorithm for “MINSUM-MAXMIN,” which, for the network of Figure 5.13, requires time \(O(n_1^2 n_2^2 \log n_1 n_2).\) A second, more appealing alternative is to use the fully polynomial approximation scheme (see page 92) for “MINSUM-MINSUM” described in [12].
This scheme, which, for the network of Figure 5.13, requires time $O(\frac{n_1 n_2}{\varepsilon} \log \frac{n_1 n_2}{\varepsilon})$, involves scaling the $f_t^2$'s and then generating approximate Pareto optimal paths.

The number of paths produced using the first (second) alternative is bounded by a polynomial in $n_1$ and $n_2$ ($n_1$, $n_2$, and $\frac{1}{\varepsilon}$). Since evaluating the actual makespan of the schedule associated with the feasible sequence defined by any path from node (0,0) to node $(n_1 + 1, n_2)$ can be accomplished in time $O(n_1 + n_2)$, then producing a set of 'nearly' Pareto optimal paths using the first (second) alternative and selecting from among these paths a path that has associated schedule with minimum makespan can be accomplished in time bounded by a polynomial in $n_1$ and $n_2$ ($n_1$, $n_2$, and $\frac{1}{\varepsilon}$).
6.1 Summary

In this dissertation, we have investigated one-machine scheduling problems subject to generalized precedence constraints. These constraints and minimum delay precedence constraints in particular can arise in the scheduling of athletic competitions. The literature directly related to generalized precedence constrained scheduling (GPCS) is seemingly scant, limited mostly to special cases and related constraints. To our knowledge, this dissertation contains the first explicit identification of generalized precedence constraints as we have defined them and represents the first systematic treatment of GPCS.

As we have seen, all but the simplest of GPCS problems are NP-hard. Among problems for which the precedence relation is $k$ 1-chains, several, including minimizing makespan subject to minimum delays, subject to maximum delays, or subject to minimum or maximum delays but not both, and minimizing total completion time subject to minimum delays, can be solved in polynomial time. Even among these problems with the simplest of precedence relations, though, are hard problems, including minimizing makespan subject to minimum and maximum delays, minimizing total weighted completion time subject to minimum delays, and minimizing total completion time subject to maximum delays, the first and third of which are NP-hard in the strong sense.
The effect on minimum makespan problems of allowing even slightly more complex precedence relations is to make hard those problems which were heretofore solvable in polynomial time. In particular, both the problem subject to minimum delays and the problem subject to maximum delays, and hence the problem subject to minimum or maximum delays but not both, are NP-hard when we allow one of the $k$ chains to include two jobs. Each of these problems is NP-hard in the strong sense when we allow one of the $k$ chains to include any number of jobs. In addition, the problem subject to minimum delays and hence the problem subject to minimum or maximum delays but not both, is NP-hard in the strong sense when we allow each of the $k$ chains to include two jobs.

In contrast to $k$ 1-chains, a “shallow” precedence relation, is $2 n_1, n_2$-chains, a “deep” precedence relation. Our results for minimum makespan problems for which the precedence relation is $2 n_1, n_2$-chains were somewhat inconclusive. The problem subject to minimum delays can be solved in pseudo-polynomial time. We were unable to further classify this problem with respect to computational complexity. On the other hand, the problem subject to maximum delays can be solved in polynomial time.

Just because a particular GPCS problem is NP-hard does not mean we cannot solve that problem with some degree of success. For example, in the case of $1 / \min_{k 2, 1, \ldots, 1}$-chains $/ C_{\max}$, we can easily identify a schedule with makespan no more than twice the optimal makespan from the class of “insertion” schedules. The special case of $1 / \min_{k 2, 1, \ldots, 1}$-chains $/ C_{\max}$ with $l_2 = \cdots = l_k = 0$ can be formulated as two subset sum problems. Consequently, this special case can be solved in pseudo-polynomial time by dynamic programming. Using Lawler’s fully polynomial approximation scheme for subset sum, we can produce a schedule for an even more special case with makespan at most $\frac{5}{4} + \Delta$ for any $\Delta \in (0, \frac{1}{4})$ in time bounded by a polynomial in the length of the problem instance and in $\frac{1}{\Delta}$.

While we do not know whether $1 / \min_{k 2, 1, \ldots, 1}$-chains $/ C_{\max}$ is NP-hard or not, we do know that several special cases, including the one in which the differ-
ence between the largest minimum delay in one chain and the smallest processing requirement in the other chain is bounded by a polynomial in \( n_1 \) and \( n_2 \), can be solved in polynomial time. As we have seen, \( 1 / \text{min delays, } 2 n_1, n_2 \)-chains / \( C_{\max} \) possesses the curious property that no schedule has makespan greater than twice the optimal makespan. The optimal makespan is bounded from below by the processing requirement and the longest chain bounds as well as by the makespan of the schedule generated using the preemptive version of the Schrage algorithm. We can solve \( 1 / \text{min delays, } 2 n_1, n_2 \)-chains / \( C_{\max} \) by branch-and-bound in conjunction with the disjunctive graph representation. Alternatively, we can generate an approximate solution using the bicriterion heuristic. Unfortunately, the former option is likely to be computationally intractable and the latter option provides solutions of unknown quality.

### 6.2 Suggestions for Future Research

Our work falls short of suggesting a general methodology for dealing with generalized precedence constraints. Instead, our results point to the importance of exploiting problem-specific structures, the likelihood that solving GPCS problems will be, in general, computationally expensive, and the need for provably effective heuristic solution techniques.

Our treatment of GPCS problems has been mostly combinatorial in nature. Other approaches such as a polyhedral approach might well prove to be fruitful. As an initial step in a polyhedral treatment of GPCS, we suggest finding the convex hull of the active schedules for \( 1 / \text{min delays, } k 1 \)-chains / \( C_{\max} \).

Whether or not there exist pseudo-polynomial time algorithms for

1. \( 1 / \text{min delays, } k 2, 1, \ldots, 1 \)-chains / \( C_{\max} \),
2. \( 1 / \text{max delays, } k 2, 1, \ldots, 1 \)-chains / \( C_{\max} \),
3. \( 1 / \text{max delays, } k 2 \)-chains / \( C_{\max} \), or
all of which are NP-hard, are open questions. We showed in Section 5.1.3 that the NP-hard special case of 1 / min delays, k 2, 1,..., 1-chains / C_{max} with l_2 = \cdots = l_k = 0 is solvable in pseudo-polynomial time. We hypothesize that the general problem is not solvable in pseudo-polynomial time.

For us, the most intriguing open question concerns the computational complexity of 1 / min delays, 2 n_1, n_2-chains / C_{max}. We hypothesize that, due to the sequence-specific nature of the active schedules, this problem cannot be solved in polynomial time and is in fact NP-hard. As an initial step in proving or disproving this hypothesis, we suggest considering the computational complexity of 1 / min delays, k n_1, \ldots, n_k-chains / C_{max} for k = 3.
BIBLIOGRAPHY


Captain Erick Douglas Wikum was born September 16, 1965 in Edgerton, Wisconsin. He spent his childhood together with his parents and three sisters in East Grand Forks, Minnesota and Plymouth, Massachusetts, as well as in Menomonie, Wisconsin, where he attended high school. Erick received his Bachelor's Degree in Mathematics and Operations Research together with a commission as a 2nd Lieutenant in the U.S. Air Force from the U.S. Air Force Academy in June of 1988. He earned a Masters' Degree in Operations Research from the Georgia Institute of Technology in December of 1989. On April 6, 1991, Erick married Joanne Elizabeth Crane. Erick presently serves as an operations research analyst in the Air Force and hopes to attend law school in the near future.