ESSAYS ON SOME COMBINATORIAL OPTIMIZATION PROBLEMS WITH INTERVAL DATA

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Hande Yaman
June, 1999

ESSAYS ON SOME COMBINATORIAL OPTIMIZATION PROBLEMS WITH INTERVAL DATA

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL

ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Hande Yaman.

By

Hande Yaman

June, 1999

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Mustafa Ç. Pınar(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Osman Oğuz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Oya Karaşan

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

ESSAYS ON SOME COMBINATORIAL OPTIMIZATION PROBLEMS WITH INTERVAL DATA

Hande Yaman
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Mustafa Ç. Pınar
June, 1999

In this study, we investigate three well-known problems, the longest path problem on directed acyclic graphs, the minimum spanning tree problem and the single machine scheduling problem with total flow time criterion, where the input data for all problems are given as interval numbers. Since optimal solutions depend on the realization of the data, we define new optimality concepts to aid decision making. We present characterizations for these "optimal" solutions and suggest polynomial time algorithms to find them in some special cases.

Key words: Longest Path Problem on Directed Acyclic Graphs, Minimum Spanning Tree Problem, Single Machine Scheduling Problem with Total Flow Time Criterion, Interval Data, Polynomial Time Algorithms

ÖZET

VERİLERİ ARALIK SAYILAR OLAN BAZI EN İYİLEME PROBLEMLERİ ÜZERİNE DENEMELER

Hande Yaman
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Mustafa Ç. Pınar
Haziran, 1999

Bu çalışmada, verileri aralık sayılarla ifade edilen yönlü çevrimsiz çizgelerde en uzun yol problemi, minimum kapsarağaç problemi ve tek makinada toplam akış zamanını azlama problemi incelenmiştir. En iyi çözümler verilere bağlı olduğundan yeni en iyi olma kavramları tanımlanmıştır. Bu "en iyi" çözümler karakterize edilmiş ve bazı özel durumlarda bu çözümleri bulan polinom zamanlı algoritmalar önerilmiştir.

Anahtar sözcükler: Yönlü Döngüsüz Ağlarda En Uzun Yol Problemi, Minimum Kapsarağaç Problemi, Tek Makinada Toplam Akış Zamanını Azlama Problemi, Aralık Sayılar, Polinom Zamanlı Algoritmalar

anneme ve anneanneme

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Mustafa Ç. Pınar for his supervision and encouragement during my graduate study. He has been so kind and patient in all my desperate times. His trust and understanding motivated me and let this thesis come to an end.

I am grateful to Oya Karaşan for her invaluable contribution to this thesis. I owe so much to her interest and suggestions. I also would like to thank her for reading and reviewing my thesis so many times.

I am indebted to Osman Oğuz for accepting to read and review this thesis and for his suggestions.

I would like to take this opportunity to thank Evrim Didem Güneş for being such a good friend in all my school life. Without her friendship and support, I would not be able to bear with all this time. I would also like to thank my officemate Ayten Türkcan for all the NP-complete problems she found for me and for her morale support and encouragement in all my hard times. I am grateful to Deniz Özdemir and Senem Erdem for their friendship, support and all the nice times we spent together. I can not forget the kitchen chats with Seçil Gergün, Özgür Ceyhan, Gonca Yıldırım, Banu Yüksel and Pelin Arun and I would like to thank them all for their helps and encouragement.

I would like to express my gratitude to Tolga Şirin for his love and kindness. I owe so much to him for happiness and peace he brought to my life.

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we investigate three well-known problems, longest path problem on acyclic directed graphs, minimum spanning tree problem and scheduling problem on a single machine with total flow time criterion where the data for all problems are represented by intervals. Deterministic versions of these problems can be solved easily. We mainly focus on the longest path problem on acyclic directed graphs with interval arc lengths and develop some new concepts of optimality to aid decision making in the presence of uncertainty. We give characterizations of these "optimal" solutions and suggest procedures to find them in some special cases. We modify some of these results for minimum spanning tree problem, and derive some basic results for the scheduling problem.

The main motivation for studying the longest path problem on acyclic directed graphs with interval data comes from critical path problem in project management. Each project consists of activities that need to be performed obeying some precedence constraints among these activities. These precedence constraints can be represented by an acyclic directed graph where each arc corresponds to an activity. When the durations of activities are point values, it is direct to determine a critical path, and schedule activities accordingly. A path is critical if and only if it is a longest path, and an activity is critical if

and only if it is on a critical path. However, it is usually hard to determine point values for the activity durations, as there are various uncertain elements that may affect the activity durations during implementation.

We encounter two ways of dealing with uncertainty in project management literature. One is PERT (Project Evaluation and Review Technique), which approximates activity durations with beta distribution and suggests an easy way of determining the critical path based on the mean durations. There are various criticisms and modifications on PERT in the literature, see for example McCahon [7]. The second one is fuzzy project analysis, where activity durations are fuzzy numbers, and critical activities are determined using fuzzy arithmetic, (Rommelfanger [9], Nasution [8] and Chanas and Kamburowski [2]).

Lootsma [6] compares and criticizes these two methods in detail (also gives further references) and concludes his discussion as follows: " In summary, we reject stochastic models in PERT planning when activity durations are estimated by human experts. We hesitate to believe, however, that the fuzzy arithmetic in its present form, should be sufficiently well established to model the vagueness of human judgment."

What we propose in this study differs from these two methods both in the way we structure uncertainty and in the way we define criticality. We structure uncertainty by taking activity durations as intervals defined by known lower and upper bounds, and do not assume any probability distribution. The duration of an activity can take any value in its interval independent of the other activities. This way of defining activity durations is easy to model, since we do not need much information compared to stochastic methods which require the probability distribution functions and the fuzzy methods which require the membership functions, that are hard to know in a nonrepetitive environment.

Since in the presence of uncertainty, the critical path depends on the realization of activity durations, we develop new concepts of criticality. These concepts

are defined by Tansel and Scheuenstuhl [11]. We would like to distinguish paths that are critical for all realizations and paths that are critical for some realizations. We use this information and make a similar analysis for activities. We would like to know which activities are critical for all realizations and which activities are critical for some realizations.

Apart from the critical path problem, this analysis is useful for the longest path problem on acyclic directed graphs with interval data. If there exists a path which is a longest path for all realizations, then this path will solve the problem. In case there does not exist such a path, then we would like to find a path such that the error we make by picking that path as the solution will be the smallest. Such a path is referred to as a *robust path.* We deal with two common robustness measures. The first one called absolute robustness considers the worst case length of a path. We would like to find a path whose shortest length among all possible realizations is the maximum. The second measure is relative robustness. This time we would like to find a path such that the maximum difference between the length of the longest path and the length of this path among all scenarios is minimum. These robustness measures are defined in Kouvelis and Yu [4] for various problems. In the introduction of their book, they motivate robust optimization in comparison with stochastic programming, and state that robust solutions perform much better than the solutions of stochastic programming in unique, nonrepetitive environments. Different from our problem, the authors introduce uncertainty to the problem by a discrete scenario set, where each scenario defines the lengths of all arcs. They prove that the robust shortest path problems are NP-complete for discrete scenario set and conjecture that the problems with interval data are also NP-complete.

In the present thesis, we derive some basic results for robust path problems with interval data and give a mixed integer programming formulation for the relative robust path problem. We show that, knowing which arcs are always on longest paths and which arcs are never on longest paths, we can preprocess a given graph for robust path problems. So, the analysis we make for project

management is useful for longest path problem. Since the longest path problem is more general, we derive all results for the longest path problem, but give interpretations of the results in terms of project management when necessary.

We do a similar analysis for the minimum spanning tree problem and the scheduling problem on a single machine with total flow time criterion. We define similar optimality concepts and give basic characterizations.

Kozina and Perepelista [5] have studied the minimum spanning tree problem with interval edge costs. They have defined a relation order on the set of feasible solutions and generated a Pareto set.

The rest of the thesis is organized as follows: Chapter 2 is devoted to the longest path problem with interval arc lengths on directed acyclic graphs. We determine which paths are longest for all realizations and which paths are longest for some realizations. We also define absolute robust and relative robust paths and derive some results for them. Then we investigate arcs, and determine whether a given arc is on longest paths for all realizations or is on longest paths for some realizations. We present mixed integer programming formulations to solve both problems. In Chapter 3, we study the longest path problem on layered graphs. We give polynomial time algorithms to solve arc problems when the width of the layered graph is 2. In Chapter 4, we extend our results to the minimum spanning tree problem with interval edge lengths. We study the single machine scheduling problem with total flow time criterion where processing times are intervals in Chapter 5. Finally, we give a conclusion in Chapter 6.

Chapter 2

Longest Path Problem with Interval Data

In this chapter, we consider the longest path problem on a directed acyclic graph $G = (V, A)$. There are n nodes in the graph, where node 1 is the start node and node n is the target node. We assume that G is topologically sorted, that is for all $(i, j) \in A$, we have $i < j$. In a directed acyclic graph, topological sorting can be achieved in $O(m)$ time [1], where m is the number of arcs. Also, deterministic version of the longest path problem on directed acyclic graphs can be solved in $O(m)$ time [1].

We structure uncertainty by interval arc lengths. Arc (i, j) has length l_{ij} within the interval $[\underline{l}_{ij}, \bar{l}_{ij}]$ where $\underline{l}_{ij} \leq \bar{l}_{ij}$. We assume that each value in the interval can be realized by some positive probability. No probability distribution is assumed for the arc lengths. l_{ij} is an arbitrary value in the interval $[\underline{l}_{ij}, \bar{l}_{ij}]$. A scenario s is a specification of all arc lengths. For scenario s, l_{ij}^s denotes the length of arc (i, j).

P is the set of all paths from node 1 to node n. $\underline{l}_p$ denotes the length of path p when the lengths of all arcs on it are at lower bounds and $\bar{l}_p$ denotes the length of path p when the lengths of all arcs on p are at upper bounds. We

denote by l_p^s the length of path p in scenario s.

A project consists of activities that need to be performed obeying some precedence constraints. The project is completed when all activities are completed. A project network, which specifies the precedence constraints, can be represented by a directed acyclic graph, where arcs correspond to activities and nodes correspond to events. Activity (i,j) can be started when all activities in $\{(k,i) : k \in V\}$ have been completed. This type of a project network is called an AOA (Activity on Arc) network. An activity represented by arc (i,j) has duration $l_{ij} \in [\underline{l}_{ij}, \bar{l}_{ij}]$.

We first analyze paths and check whether there exists a path in the graph which is a longest path for all possible realizations of arc lengths. Then we decide whether there exists a scenario for which a given path is a longest path. We also define two robustness measures based on the worst case performances of paths.

Next, we distinguish arcs which are always on longest paths, and arcs that are never on a longest path. We give mixed integer programming formulations to check whether a given arc is always on a longest path and a given arc is never on a longest path. We show that this analysis can be used to preprocess the graph for robust path problems.

2.1 Paths

When interval arc lengths are introduced to the problem, the concept of being a longest path is weakened, since the longest path depends on the realizations of arc lengths. It is necessary to distinguish paths that are longest for all realizations, paths that are longest for some realizations and paths that are longest for no realization. When we think in terms of the longest path problem, if there exists a path which is longest for all realizations, we should pick that path. But if such a path does not exist, we may still need to choose

a "best" path. In this case, we evaluate paths according to their worst case performances.

2.1.1 Permanent Paths

The first question we address is whether there exists a path which is longest for all realizations or not. We call such a path *permanent* and give a necessary and sufficient condition for a path to be permanent. We also present polynomial time procedures to decide whether there exists a permanent path or not.

Definition 2.1 *A path is a* **permanent path** *if it is a longest path for all realizations of arc lengths.*

The following theorem characterizes permanent paths.

Theorem 2.1 *A path is a permanent path if and only if it is one of the longest paths when the lengths of all arcs on this path are at their lower bounds and the lengths of all other arcs are at their upper bounds.*

Proof

If a path is a permanent path then it is one of the longest paths for the stated realization by definition.

If a path p is a longest path when the lengths of all arcs on p are at their lower bounds and the lengths of all other arcs are at their upper bounds, for an arbitrary path $p' \in P$, we have:

$$\sum_{(i,j)\in p\setminus p'} \underline{l}_{ij} + \sum_{(i,j)\in p\cap p'} \underline{l}_{ij} \geq \sum_{(i,j)\in p'\setminus p} \overline{l}_{ij} + \sum_{(i,j)\in p\cap p'} \underline{l}_{ij}$$

If we consider arbitrary values in the given intervals for the lengths of arcs $(i,j) \in p \cap p'$, we obtain:

$$\sum_{(i,j)\in p\setminus p'} \underline{l}_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij} \geq \sum_{(i,j)\in p'\setminus p} \overline{l}_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij}$$

Since $\sum_{(i,j)\in p\setminus p'} l_{ij} \geq \sum_{(i,j)\in p\setminus p'} \underline{l}_{ij}$ and $\sum_{(i,j)\in p'\setminus p} \bar{l}_{ij} \geq \sum_{(i,j)\in p'\setminus p} l_{ij}$, we have:

$$\sum_{(i,j)\in p\setminus p'} l_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij} \geq \sum_{(i,j)\in p'\setminus p} l_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij}$$

So $l_p \geq l_{p'}$ for all realizations. Since p' is an arbitrary path, this is true for all paths in P. Thus, p is a permanent path. □

So given a path we can check whether it is permanent or not in polynomial time. We simply set the lengths of all arcs on this path at lower bounds and the lengths of the remaining arcs at upper bounds and find a longest path in this graph. If the length of this path coincides with the length of the longest path, then this path is permanent. Otherwise, it is not.

Proposition 2.1 *Suppose there exists a permanent path p. Consider the scenario in which all arc lengths are at lower bounds. If there exists a path p' such that $\underline{l}_p = \underline{l}_{p'}$, then $\underline{l}_{ij} = \bar{l}_{ij}$ for all $(i,j) \in p'\setminus p$. Moreover p' is also a permanent path if and only if $\underline{l}_p = \underline{l}_{p'}$ and $\underline{l}_{ij} = \bar{l}_{ij}$ for all $(i,j) \in p\setminus p'$.*

Proof

Suppose there exists a path p' such that $\underline{l}_p = \underline{l}_{p'}$ and assume there exists an arc $(i,j) \in p'\setminus p$ whose length is non degenerate, that is $\underline{l}_{ij} < \bar{l}_{ij}$. By setting the length of arc (i,j) to its upper bound, we obtain a scenario in which p' is longer than p. This contradicts that p is a permanent path. So for each arc $(i,j) \in p'\setminus p$, we have $\underline{l}_{ij} = \bar{l}_{ij}$.

If there exists a path p' which is also a permanent path, for all realizations $l_p = l_{p'}$ and $\underline{l}_{ij} = \bar{l}_{ij}$ for all $(i,j) \in p'\setminus p$. Then, for all possible realization of arc lengths we have:

$$\sum_{(i,j)\in p\setminus p'} l_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij} = \sum_{(i,j)\in p'\setminus p} l_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij}$$

This implies that:

$$\sum_{(i,j)\in p\setminus p'} l_{ij} = \sum_{(i,j)\in p'\setminus p} l_{ij}$$

Since the right hand side of the inequality is a constant for all realizations, we should have $\underline{l}_{ij} = \bar{l}_{ij}$ for all $(i,j) \in p \backslash p'$.

If there exists a path p' such that $\underline{l}_p = \underline{l}_{p'}$ and $\underline{l}_{ij} = \bar{l}_{ij}$ for all $(i,j) \in p \backslash p'$, then $\sum_{(i,j) \in p \backslash p'} l_{ij} = \sum_{(i,j) \in p' \backslash p} l_{ij}$ for all realizations. This implies that $l_p = l_{p'}$ for all realizations. Thus p' is also a permanent path. □

Corollary 2.1 *If all arc lengths are non degenerate and there exists a permanent path, it is the unique longest path when the lengths of all arcs are at their lower bounds.*

Corollary 2.2 *If all arc lengths are non degenerate and there exists a permanent path, it is unique.*

Corollary 2.3 *If there exists a permanent path p and if all arc lengths on p are non degenerate, then p is the unique permanent path.*

In case when all arc lengths are non degenerate, the previous corollaries 2.1 and 2.2 lead us to the following simple procedure to find a permanent path when one exists.

Procedure FindPermanentPath

1. Set all arc lengths at their lower bounds.
2. Find a longest path.
3. Check whether this path is permanent or not using the characterization in Theorem 2.1. If this path is permanent, then it is the unique permanent path. Otherwise, there exists no permanent path in the graph due to Corollary 2.1.

Next, we relax the assumption of non degenerate arc lengths. We present a procedure that can find a permanent path if it exists, when some of the arcs

have degenerate lengths. We use the fact that if there exists a permanent path p in the graph, then the partial path of p from node 1 to any node i on p is a permanent partial path between these nodes.

We investigate each node i in the topological ordering, starting from node 3 and check whether there exists a permanent partial path from node 1 to node i. If there does not exist such a permanent partial path, we remove node i and all arcs incident at it from the graph, since this node cannot be on a permanent path. If there exists a unique permanent partial path from node 1 to node i, then this path uses an arc (k,i), we keep this arc and remove all other arcs coming into node i. If there are more than one permanent partial path, then all arcs on these paths which have non degenerate arc lengths should be common in these paths, and the lengths of these paths are equal when the lengths of all arcs are at lower bounds. Proposition 2.1 implies that one of these paths is permanent if and only if the others are permanent, so we can pick any of these paths. Say we pick the path using arc (k,i), then we keep this arc and remove all other arcs coming into node i. So to each node processed, there exists a unique incoming arc. This implies that if we are about to process node i, we consider at most $i-1$ paths in the worst case. We either stop with a disconnected graph at some iteration with the conclusion that there exists no permanent path, or we end up with a unique path p and check whether this path is permanent or not. Each path eliminated is either not permanent or is permanent if and only if path p is permanent. So, if path p is not permanent, we conclude that there exists no permanent path in the graph.

In checking whether there exists a permanent partial path to node i from node 1, we do not need to check each path separately. Let $\{p_1, p_2, .., p_j\}$ denote the set of paths from node 1 to node i. The cardinality of this set is bounded above by $i-1$. We first consider paths p_1 and p_2. If $\sum_{(m,n)\in p_1\setminus p_2} \underline{l}_{mn} \geq \sum_{(m,n)\in p_2\setminus p_1} \bar{l}_{mn}$, then we remove path p_2 from the set. We continue with paths p_1 and p_3. Otherwise, if $\sum_{(m,n)\in p_2\setminus p_1} \underline{l}_{mn} \geq \sum_{(m,n)\in p_1\setminus p_2} \bar{l}_{mn}$, then we remove path p_1 from the set and continue with paths p_2 and p_3. If none of the inequalities holds, then we remove both paths from the set and continue with p_3 and p_4. So we

make at most $i-2$ comparisons for node i in the worst case. If at the end, the set becomes empty then none of these paths can be a permanent partial path between nodes 1 and i and thus node i cannot be on a permanent path. Then we remove node i with all arcs incident at node i from the graph.

Proposition 2.2 *Procedure FindPermanentPath can decide whether there exists a permanent path in an arbitrary graph in $O(n^2)$ time.*

Proof
In the worst case, for each node i, we make $i-2$ comparisons and $\sum_{i=3}^{n} i-2 = \sum_{i=1}^{n-2} i = (n-2)(n-1)/2 \leq n^2$. □

2.1.2 Weak Paths

When there does not exist a permanent path in the graph, we would like to find a path whose worst case performance is the best among all paths. Before doing that, we would like to see which paths are never longest, since such paths cannot be candidates for having a good worst case performance.

Definition 2.2 *A path is a* **weak path** *if it is one of the longest paths at least for one realization of arc lengths.*

We can check whether a given path is a weak path or not in polynomial time by the following characterization:

Theorem 2.2 *A path is a weak path if and only if it is one of the longest paths when the lengths of all arcs on this path are at their upper bounds and the lengths of the other arcs are at their lower bounds.*

Proof
If a path is longest for the stated realization of arc lengths, it is a weak path by definition.

If a path p is a weak path, then there exists a scenario s for which

$$\sum_{(i,j)\in p} l_{ij}^s \geq \sum_{(i,j)\in p'} l_{ij}^s \quad \forall p' \in P$$

Let p' be an arbitrary path in P. We have:

$$\sum_{(i,j)\in p\setminus p'} l_{ij}^s + \sum_{(i,j)\in p\cap p'} l_{ij}^s \geq \sum_{(i,j)\in p'\setminus p} l_{ij}^s + \sum_{(i,j)\in p\cap p'} l_{ij}^s$$

When we set the lengths of all arcs in $p \cap p'$ at their upper bounds, we still have:

$$\sum_{(i,j)\in p\setminus p'} l_{ij}^s + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij} \geq \sum_{(i,j)\in p'\setminus p} l_{ij}^s + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij}$$

When we consider the scenario in which the lengths of all arcs on p are at upper bounds and the lengths of the remaining arcs are at lower bounds, since $\sum_{(i,j)\in p\setminus p'} \bar{l}_{ij} \geq \sum_{(i,j)\in p\setminus p'} l_{ij}^s$ and $\sum_{(i,j)\in p'\setminus p} \underline{l}_{ij} \leq \sum_{(i,j)\in p'\setminus p} l_{ij}^s$, we obtain:

$$\sum_{(i,j)\in p\setminus p'} \bar{l}_{ij} + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij} \geq \sum_{(i,j)\in p'\setminus p} \underline{l}_{ij} + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij}$$

Since p' is picked arbitrarily, this is true for all paths in P. Thus p is also a longest path when the lengths of arcs on p are at their upper bounds and the lengths of all other arcs are at their lower bounds. □

If the number of paths is not large, the above characterization shows us an easy way to find all weak paths. But in a complete directed acyclic graph with n nodes, there are 2^{n-2} paths from node 1 to node n.

The next proposition gives a necessary and sufficient condition for a given path to be weak if there exists a permanent path in the graph.

Proposition 2.3 *If there exists a permanent path p, then a path p' is a weak path if and only if $\underline{l}_p = l_{p'}^s$ in scenario s, in which the lengths of all arcs on p are at their lower bounds and the lengths of all other arcs are at their upper bounds.*

Proof

If $\underline{l}_p = l^s_{p'}$ for scenario s, then p' is a longest path for this scenario. Thus p' is a weak path.

If p' is weak path, then p' is a longest path when the lengths of all arcs on p' are at upper bounds and the lengths of the remaining arcs are at lower bounds. So we have:

$$\sum_{(i,j)\in p\setminus p'} \underline{l}_{ij} + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij} = \sum_{(i,j)\in p'\setminus p} \bar{l}_{ij} + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij}$$

When we set the lengths of arcs in $p \cap p'$ at lower bounds, we still have:

$$\sum_{(i,j)\in p\setminus p'} \underline{l}_{ij} + \sum_{(i,j)\in p\cap p'} \underline{l}_{ij} = \sum_{(i,j)\in p'\setminus p} \bar{l}_{ij} + \sum_{(i,j)\in p\cap p'} \underline{l}_{ij}$$

So $\underline{l}_p = l^s_{p'}$. □

So, if there exists a permanent path p, we can find all weak paths by finding all longest paths when the lengths of all arcs on p are at lower bounds and the lengths of all other arcs are at upper bounds.

Corollary 2.4 *If there exists a permanent path, p, and if* $\underline{l}_p > l^s_{p'}$ $\forall p' \in P$ *in scenario s in which the lengths of all arcs on p are at lower bounds and the lengths of all other arcs are at upper bounds, then no other path can be weak.*

Thus, if there exists a permanent path, we can check whether there exist other weak paths or not simply by finding the second longest path.

Weak paths and permanent paths help us in determining critical activities in project networks. If an activity is on a permanent path, then for all realizations of activity durations, this activity will be critical. So delaying such an activity will cause the project to last longer. On the other hand, if an activity is on a weak path, there exists a scenario in which this activity is critical. If no weak path uses an arc, then this activity will not be critical for any realization. However, we do not know exactly how long this activity can be delayed without delaying the project.

2.1.3 Robust Paths

Next we give a formal definition to what we mean by worst case performance. We call a path whose worst case performance is best among all paths a *robust* path. We define two different robustness measures. The first measure is absolute robustness. We would like to find a path whose minimum length over all scenarios is maximum. The second one is relative robustness. In this one, we would like to find a path for which the maximum difference between the length of the longest path and the length of this path among all scenarios is minimum. It is clear that if there exists a permanent path in the graph, this path will both be an absolute robust path and a relative robust path. Though the names may be different, both robustness concepts are due to Kouvelis and Yu [4].

Let S denote the set of all possible scenarios.

Definition 2.3 *Given a path p, the* **absolute worst case scenario** s_p^a *is the scenario in which the length of this path is the smallest. That is,* $s_p^a = \arg\min_{s \in S} l_p^s$.

It is easy to see that the absolute worst case scenario for a path p corresponds to the scenario in which all arcs on this path are at lower bounds.

Definition 2.4 *The path whose length is the maximum for the absolute worst case scenario is called an* **absolute robust path**. *So the absolute robust path* $p^a = \arg\max_{p \in P} \min_{s \in S} l_p^s = \arg\max_{p \in P} l_p^{s_p^a}$.

Kouvelis and Yu [4] have studied the absolute robust shortest path problem, where the scenario set is finite. They have shown that the absolute robust shortest path problem is NP-complete even in layered networks of width 2 and with only 2 scenarios. Moreover, they have also proved that the problem becomes strongly NP-hard when the scenario set is unbounded. However, in our case the absolute worst case scenarios for all paths correspond to the scenario

in which all arc lengths are at lower bounds. So, it is enough to consider this unique scenario. Then, the absolute robust path is the longest path for this scenario. Thus, the absolute robust path problem with interval arc lengths can be solved in polynomial time.

Definition 2.5 *Given a path p, the* **relative worst case scenario** s_p^r *is the scenario in which the difference between the length of the longest path and the length of path p is the maximum. That is,* $s_p^r = \arg\max_{s\in S} l^s_{p^*(s)} - l^s_p$, *where* $p^*(s)$ *is the longest path for scenario s. We call the difference* $d_p = l^{s_p^r}_{p^*(s_p^r)} - l^{s_p^r}_p$ *the* **robust deviation** *for path p.*

Definition 2.6 *The path whose robust deviation is the minimum is called a* **relative robust path.** *In other words, the relative robust path* $p^r = \arg\min_{p\in P} d_p = \arg\min_{p\in P}\max_{s\in S} l^s_{p^*(s)} - l^s_p$.

It is clear that an absolute robust path is a weak path. We show that a relative robust path is also a weak path.

Proposition 2.4 *A relative robust path is a weak path.*

Proof

Let p be a path, which is not weak. Consider path p' which is a longest path for the scenario in which the lengths of all arcs on p are at upper bounds and the lengths of the remaining arcs are at lower bounds. Then $l_p < l_{p'}$ for all scenarios. Consider the relative worst case scenario for path p' . We have:

$$d_{p'} = l^{s^r_{p'}}_{p^*(s^r_{p'})} - l^{s^r_{p'}}_{p'} < l^{s^r_{p'}}_{p^*(s^r_{p'})} - l^{s^r_{p'}}_{p} \leq \max_{s\in S} l^s_{p^*(s)} - l^s_p = d_p$$

So p cannot be a relative robust path. □

The following proposition states a relative worst case scenario for a given path.

Proposition 2.5 *The relative worst case scenario for path p is the scenario in which the lengths of all arcs on p are at lower bounds and the lengths of all other arcs are at upper bounds.*

Proof

Let d_p be the robust deviation for path p. Then:

$$d_p = l^{s_p^r}_{p^*(s_p^r)} - l^{s_p^r}_p = \sum_{(i,j)\in p^*(s_p^r)\setminus p} l^{s_p^r}_{ij} - \sum_{(i,j)\in p\setminus p^*(s_p^r)} l^{s_p^r}_{ij}$$

Let s be the scenario in which the lengths of all arcs on p are at their lower bounds and the lengths of the remaining arcs are at their upper bounds. Since $\sum_{(i,j)\in p^*(s_p^r)\setminus p} l^{s_p^r}_{ij} \leq \sum_{(i,j)\in p^*(s_p^r)\setminus p} \overline{l}_{ij}$ and $\sum_{(i,j)\in p\setminus p^*(s_p^r)} l^{s_p^r}_{ij} \geq \sum_{(i,j)\in p\setminus p^*(s_p^r)} \underline{l}_{ij}$, we have:

$$d_p \leq \sum_{(i,j)\in p^*(s_p^r)\setminus p} l^s_{ij} - \sum_{(i,j)\in p\setminus p^*(s_p^r)} l^s_{ij} = l^s_{p^*(s_p^r)} - l^s_p$$

Certainly $l^s_{p^*(s)} \geq l^s_{p^*(s_p^r)}$. So we have:

$$d_p \leq l^s_{p^*(s_p^r)} - l^s_p \leq l^s_{p^*(s)} - l^s_p$$

Since $d_p = \max_{s'\in S} l^{s'}_{p^*(s')} - l^{s'}_p$, we have $d_p = l^s_{p^*(s)} - l^s_p$. So s is a relative worst case scenario for path p. □

Kouvelis and Yu [4] have also proved that relative robust shortest path problem is NP-complete even in layered networks of width 2 and with only 2 scenarios and is strongly NP-hard with unbounded number of scenarios. In case of interval arc lengths, Proposition 2.5 implies that we need to consider a finite number of scenarios which is equal to the number of paths in the graph. However the number of paths in a graph grows exponentially with the number of nodes in the graph in the worst case. In Figure 2.1, there is the graph on which Kouvelis and Yu [4] proved that the relative robust shortest path problem is NP-complete by a reduction from 2-partition problem.

If we put interval arc lengths on such a graph, the relative robust path problem can be solved easily by considering each pair of arcs separately, since the partial path of the relative robust path from node i to node j is a relative robust path for the subgraph between these nodes. However, this is not necessarily the case in an arbitrary graph.

Kouvelis and Yu [4] conjecture that the relative robust path problem with

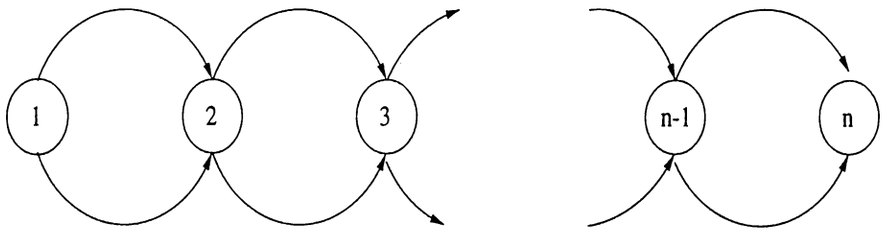

Figure 2.1: The graph on which Kouvelis and Yu proved that relative robust shortest path problem is NP-complete

interval arc lengths is also NP-complete.

Next we give a mixed integer programming formulation by Tansel [10] to find the relative robust path in a graph. Define y_{kl}'s as follows:

$$y_{kl} = \begin{cases} 1 & \text{if arc } (k,l) \text{ is on the path} \\ 0 & \text{otherwise} \end{cases}$$

We have the network flow constraints:

$$-\sum_{k \in \Gamma^-(l)} y_{kl} + \sum_{h \in \Gamma^+(l)} y_{lh} = b_l \quad l = 1, 2, .., n$$

where

$$b_l = \begin{cases} 1 & \text{for } l = 1 \\ 0 & \text{for } l \neq 1, n \\ -1 & \text{for } l = n \end{cases}$$

$\Gamma^-(l) = \{k \in V : (k,l) \in A\}$, and $\Gamma^+(l) = \{h \in V : (l,h) \in A\}$

A vector $\mathbf{y}$ satisfying the above set of constraints defines a path in the graph. The length of an arc (k,l) is defined as $l_{kl} = \bar{l}_{kl} - (\bar{l}_{kl} - \underline{l}_{kl})y_{kl}$ for a given vector $\mathbf{y}$. It can easily be verified that the length of an arc (k,l) on the path defined by $\mathbf{y}$ is at its lower bound since $y_{kl} = 1$, and the length of an arc (k,l) which is not on this path is at its upper bound since $y_{kl} = 0$.

Let x_l be the longest distance from node 1 to node l. We have the following constraints which specifies these distances according to the scenario considered:

$$x_l \geq x_k + \bar{l}_{kl} - (\bar{l}_{kl} - \underline{l}_{kl})y_{kl} \quad \forall (k,l) \in A$$

So x_n is the length of the longest path in the graph for the scenario defined by $\mathbf{y}$. The objective is to find a path for which the difference between the length of the longest path and the length of that path is the smallest for the corresponding scenario, i.e. the scenario in which the lengths of all arcs on this path are at lower bounds and the lengths of the remaining arcs are at upper bounds.

The formulation is as follows:

(RRP)

$$\min x_n - \sum_{(k,l)\in A} \underline{l}_{kl} y_{kl}$$

subject to

$$x_l \geq x_k + \bar{l}_{kl} - (\bar{l}_{kl} - \underline{l}_{kl}) y_{kl} \quad \forall (k,l) \in A$$

$$-\sum_{k\in\Gamma^-(l)} y_{kl} + \sum_{h\in\Gamma^+(l)} y_{lh} = b_l \quad j = 1,2,..,n$$

$$y_{kl} \in \{0,1\} \quad \forall (k,l) \in A \qquad x_k \geq 0 \quad k = 1,2,..,n$$

When we relax the integrality constraints in the above formulation, even when all arc lengths are [0,1], we may not be able to obtain an integral solution.

It is easy to see that the absolute robust path and relative robust path are not necessarily the same paths. Among the two robustness measures, relative robustness makes more sense when interval arc lengths are considered, since absolute robustness gives us information only about the scenario in which all arc lengths are at lower bounds. Consider the extreme case when the lengths of all arcs on the absolute robust path are degenerate, and all other arc lengths correspond to large intervals. Then picking this path as a robust path would not be a good choice.

2.2 Arcs

We have examined the paths in a graph and developed ways to recognize paths that are longest for all realizations, paths that are longest for some realizations and paths that are longest for no realization. Next we would like to analyze arcs and make similar classifications. We would like to find out which arcs are on longest paths for all realizations, which arcs are on longest paths for some realizations and which ones are never on a longest path. In doing this, we use the information we obtain from analyzing paths. Moreover, we can use the information we get from analyzing arcs to preprocess the graph for path problems. For the relative robust path problem, if we can figure out which arcs are not on longest paths for any realization, we can get eliminate paths using these arcs, since a relative robust path is a weak path. On the other hand, if we know which arcs are always on longest paths, we can divide the problem into subproblems by forcing the relative robust path we are looking for to use these arcs.

The analysis we make for arcs is directly related with understanding the criticality of activities in project networks. In terms of project management, it is useful to know which activities are critical for all realizations, and which activities are critical for no realization. Note that, even if no permanent path exists, there may be activities that are critical for all realizations.

2.2.1 Weak Arcs

First we would like to see which arcs are never on longest paths. We call an arc *weak* if it is on a longest path for some realization. We first investigate the case where there exists a permanent path in the graph and present a polynomial time procedure to find the set of weak arcs in this case. We also suggest two polynomial time procedures to detect some of the non weak arcs. Then we give a mixed integer programming formulation to decide whether a given arc is weak or not.

Definition 2.7 *An arc is a* **weak arc** *if it lies on at least one of the weak paths.*

The first proposition gives a necessary condition for an arc to be weak, which is easy to check.

Proposition 2.6 *If arc (i,j) is a weak arc, then arc (i,j) is a weak path between node i and node j.*

Proof

If arc (i,j) is not a weak path between node i and node j when the length of arc (i,j) is at its upper bound and the lengths of all other arcs are at their lower bounds, there exists another path from node i to node j, p_{i-j} for which $\bar{l}_{ij} < \underline{l}_{p_{i-j}}$. Let p be any path using arc (i,j) and p' be the path which has the same arcs as p except for the partial path between node i and node j being p_{i-j}. If we set the lengths of all arcs on p to their upper bounds, we obtain:

$$\sum_{(k,l)\in p\setminus(i,j)} \bar{l}_{kl} + \bar{l}_{ij} < \sum_{(k,l)\in p'\setminus p_{i-j}} \bar{l}_{kl} + \underline{l}_{p_{i-j}}$$

So p' is longer than p for this scenario, and thus p is not a weak path. Since no path using arc (i,j) can be weak, arc (i,j) is not a weak arc. □

We next analyze the case in which there exists a permanent path and give a polynomial time algorithm to find all weak arcs in this case.

Proposition 2.7 *If there exists a permanent path p, arc (i,j) is a weak arc if and only if there exists a path p' using arc (i,j) such that $\underline{l}_p = l^s_{p'}$ in scenario s in which the lengths of all arcs on p are at their lower bounds and the lengths of all other arcs are at their upper bounds.*

Proof

Arc (i,j) is a weak arc if and only if it lies on some weak path, p'. If there exists a permanent path p, p' is a weak path if and only if $\underline{l}_p = l^s_{p'}$ for scenario

s. □

So we can check whether arc (i,j) is weak or not, by finding the longest path using arc (i,j) for the scenario in which all arcs on a permanent path are at lower bounds and the lengths of the remaining arcs are at upper bounds. If the length of this path is equal to the length of the permanent path for this scenario, then arc (i,j) is a weak arc. Otherwise, it is not.

The procedure below can find the set of weak arcs, the weak arc set W_a, when there exists a permanent path p.

Procedure FindWeakArc

1. All arcs on p are weak arcs, so add them to W_a and remove them from A.
2. Set the lengths of all arcs on p to their lower bounds and the lengths of all other arcs to their upper bounds.
3. For each arc $(i,j) \in A$
   (a) Find the longest path from node 1 to node i, p_{1-i}, and the longest path from node j to node n, p_{j-n}. Let $p' = p_{1-i} \cup (i,j) \cup p_{j-n}$.
   (b) If $\underline{l}_p = l_{p'}$, then all arcs on p' are weak arcs, so add them to W_a and remove them from A. If $\underline{l}_p > l_{p'}$, then (i,j) is not a weak arc. Remove it from A.

Proposition 2.8 *If there exists a permanent path in the graph, Procedure FindWeakArc can determine all weak arcs in $O(m^2)$ time.*

Proof

In the worst case, Procedure FindWeakArc solves a longest path problem for each arc in the graph, and for a fixed scenario we can find a longest path in $O(m)$ time. □

So if there exists a permanent path, we can find all weak arcs in polynomial time. In case there does not exist a permanent path, we have two procedures to eliminate arcs which are not weak. The first procedure is a forward pass, which eliminates arcs that cannot be on a weak path. In this procedure, we start with node 2 and consider all nodes one by one. For each node j, we compute a lower bound $\underline{g}_j$ and an upper bound $\overline{g}_j$, such that for all realizations, the length of the longest path from node 1 to node j takes a value in the interval $[\underline{g}_j, \overline{g}_j]$. We compute $\underline{g}_j$ by picking the maximum of $\underline{b}_{ij} = \underline{g}_i + \underline{l}_{ij}$ over all $i \in \Gamma^-(j)$, where $\Gamma^-(j) = \{i \in V : (i,j) \in A\}$. This is equivalent to finding the length of the longest path from node 1 to node j, when the lengths of all arcs are at lower bounds. Similarly, we compute $\overline{g}_j$ by picking the maximum of $\overline{b}_{ij} = \overline{g}_i + \overline{l}_{ij}$ over all $i \in \Gamma^-(j)$. If there exists a node $i \in \Gamma^-(j)$ such that $\overline{b}_{ij} < \underline{g}_j$, then for all realizations the longest path from node 1 to node j will not use arc (i,j). So we drop arc (i,j) from the graph, since this arc cannot be on a weak path.

The second procedure is a backward pass, which is similar to the forward pass. In fact, it is equivalent to the forward pass applied to the mirror version of the original graph.

First, we present the forward pass procedure.

Procedure ForwardPass

1. $\underline{g}_1 = \overline{g}_1 = 0$
2. for $j = 2$ to n
   (a) Compute

$$\underline{b}_{ij} = \underline{g}_i + \underline{l}_{ij} \quad \forall i \in \Gamma^-(j)$$

$$\overline{b}_{ij} = \overline{g}_i + \overline{l}_{ij} \quad \forall i \in \Gamma^-(j)$$

$$\underline{g}_j = \max_{i \in \Gamma^-(j)} \underline{b}_{ij}$$

$$\overline{g}_j = \max_{i \in \Gamma^-(j)} \overline{b}_{ij}$$

(b) if for some $i \in \Gamma^-(j)$ $\overline{b}_{ij} < \underline{g}_j$ then arc (i,j) is not a weak arc and can be eliminated.

The following proposition states that procedure ForwardPass eliminates arcs that are not weak.

Proposition 2.9 *If an arc is eliminated by the procedure ForwardPass it is not a weak arc.*

Proof

If arc (i,j) is a weak arc, then there exists a path p, which uses arc (i,j) and which is a longest path, when the lengths of all arcs on p are at their upper bounds and the lengths of all other arcs are at their lower bounds. Then the partial path p_{1-j} of p from node 1 to node j is one of the longest partial paths from node 1 to node j for this scenario. Let k be an arbitrary node in $\Gamma^-(j)$ and p^k_{1-j} denote an arbitrary partial path from node 1 to node j visiting node k. We have:

$$\sum_{(m,n)\in p_{1-j}} \overline{l}_{mn} \geq \sum_{(m,n)\in p^k_{1-j}\setminus p_{1-j}} \underline{l}_{mn} + \sum_{(m,n)\in p^k_{1-j}\cap p_{1-j}} \overline{l}_{mn}$$

This implies that:

$$\sum_{(m,n)\in p_{1-j}} \overline{l}_{mn} \geq \sum_{(m,n)\in p^k_{1-j}\setminus p_{1-j}} \underline{l}_{mn} + \sum_{(m,n)\in p^k_{1-j}\cap p_{1-j}} \underline{l}_{mn}$$

This is equivalent to:

$$\sum_{(m,n)\in p_{1-j}} \overline{l}_{mn} \geq \sum_{(m,n)\in p^k_{1-j}} \underline{l}_{mn}$$

Since p^k_{1-j} is picked arbitrarily, we have:

$$\overline{b}_{ij} \geq \underline{g}_k + \underline{l}_{kj} = \underline{b}_{kj}$$

The above inequality is true for all $k \in \Gamma^-(j)$ since we picked k arbitrarily. Also, since $\underline{g}_j = \max_{k\in\Gamma^-(j)} \underline{b}_{kj}$, we have:

$$\overline{b}_{ij} \geq \underline{g}_j$$

Thus arc (i,j) will not be eliminated by the ForwardPass procedure. □

Next we give the second procedure, which is a backward pass.

Procedure BackwardPass

1. $\underline{f}_n = \overline{f}_n = 0$
2. for $i = n-1$ to 1
   (a) Compute
   $$\underline{a}_{ij} = \underline{f}_j + \underline{l}_{ij} \quad \forall j \in \Gamma^+(i)$$
   $$\overline{a}_{ij} = \overline{f}_j + \overline{l}_{ij} \quad \forall j \in \Gamma^+(i)$$
   $$\underline{f}_i = \max_{j \in \Gamma^+(i)} \underline{a}_{ij}$$
   $$\overline{f}_i = \max_{j \in \Gamma^+(i)} \overline{a}_{ij}$$
   where $\Gamma^+(i) = \{j \in V : (i,j) \in A\}$
   (b) if for some $j \in \Gamma^+(i)$ $\overline{a}_{ij} < \underline{f}_i$ then arc (i,j) is not a weak arc and can be eliminated.

We have a similar proposition, which says that BackwardPass procedure eliminates arcs that are not weak. Since procedure BackwardPass is equivalent to the procedure ForwardPass applied to the mirror version of the graph, we skip the proof.

Proposition 2.10 *If an arc is eliminated by the procedure BackwardPass, it is not a weak arc.*

Since both procedures examine each arc only once, they run in $O(m)$ time.

Backward and ForwardPass procedures do not necessarily eliminate the same arcs. So it is clear that none of the procedures is capable of eliminating all non-weak arcs. Moreover, we are not able to determine all the non-weak arcs

even if we use both procedures.

In the graph in Figure 2.2, the BackwardPass procedure eliminates arc (3,4) and the ForwardPass procedure eliminates arc (2,3). Both arcs are not weak. However, none of the procedures eliminates arc (2,4) which is also not weak.

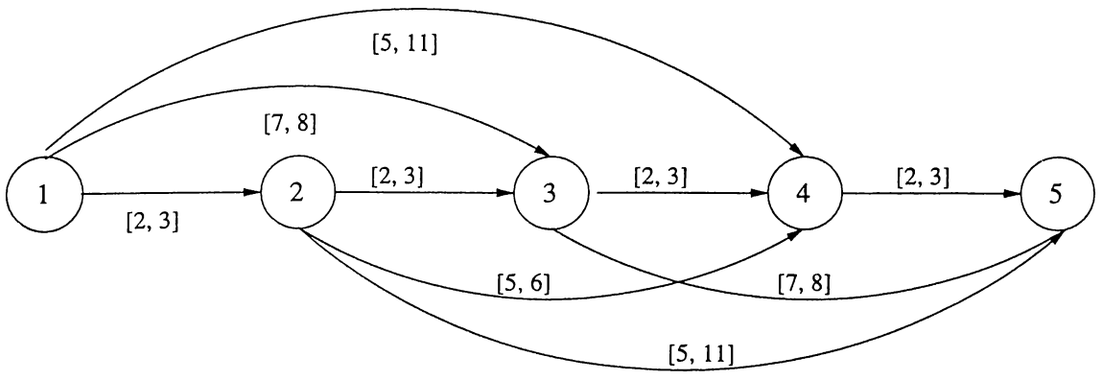

Figure 2.2: The graph on which neither procedure ForwardPass nor procedure BackwardPass can eliminate all non weak arcs

The following lemma characterizes weak arcs.

Lemma 2.1 *An arc* (i,j) *is weak if and only if* $\min_{p \in P_{(i,j)}} \{l^{s_p}_{p^*(s_p)} - \bar{l}_p\} = 0$, *where* $P_{(i,j)}$ *is the set of paths using arc* (i,j), s_p *is the scenario in which the lengths of all arcs on path* p *are at their upper bounds and the lengths of the remaining arcs at lower bounds.*

Proof

Since $p^*(s_p)$ is the longest path in the graph for scenario s_p, $l^{s_p}_{p^*(s_p)} - \bar{l}_p \geq 0 \; \forall p \in P_{(i,j)}$.

An arc (i,j) is weak if and only if there exists a weak path using arc (i,j), if and only if there exists a path $p \in P_{(i,j)}$ such that $l^{s_p}_{p^*(s_p)} - \bar{l}_p = 0$ if and only if $\min_{p \in P_{(i,j)}} \{l^{s_p}_{p^*(s_p)} - \bar{l}_p\} = 0$. □

So, we can check whether a given arc (i,j) is weak or not by the following

mixed integer programming formulation. This formulation is similar to the formulation we have for the relative robust path problem.

(WA)

$$\min x_n - \sum_{(k,l)\in A} \bar{l}_{kl} y_{kl}$$

subject to

$$x_l \geq x_k + \underline{l}_{kl} + (\bar{l}_{kl} - \underline{l}_{kl}) y_{kl} \quad \forall (k,l) \in A$$

$$-\sum_{k\in\Gamma^-(l)} y_{kl} + \sum_{h\in\Gamma^+(l)} y_{lh} = b_l \quad j = 1,2,..,n$$

$$y_{ij} = 1$$

$$y_{kl} \in \{0,1\} \quad \forall (k,l) \in A \qquad x_k \geq 0 \quad k = 1,2,..,n$$

A vector y satisfying the network flow constraints and $y_{ij} = 1$ defines a path in the graph using arc (i,j). The length of arc (k,l) is defined as $l_{kl} = \underline{l}_{kl} + (\bar{l}_{kl} - \underline{l}_{kl}) y_{kl}$ for a given vector y. So, the lengths of all arcs on the path defined by y are at their upper bounds, and the lengths of the remaining arcs are at their lower bounds.

Similar to the relative robust path problem, we have the following constraints which specify the longest distances from node 1 to each node l:

$$x_l \geq x_k + \underline{l}_{kl} + (\bar{l}_{kl} - \underline{l}_{kl}) y_{kl} \quad \forall (k,l) \in A$$

Thus x_n is the length of the longest path in the graph. The objective is to find a path using arc (i,j), for which the difference between the length of the longest path and the length of that path is the smallest for the corresponding scenario, i.e. the scenario in which the lengths of all arcs on this path are at upper bounds and the lengths of the remaining arcs are at lower bounds.

Hence, we have a theorem which gives a characterization for weak arcs using the formulation WA.

Theorem 2.3 *Arc (i,j) is a weak arc if and only if WA has an optimal objective value of 0.*

Proof
Simply follows from Lemma 2.1. □

2.2.2 Strong Arcs

Having investigated which arcs are never on longest paths, we would like to find arcs that are on longest paths for all realizations. We first consider the case where a permanent path exists. Then we give a mixed integer programming formulation to check whether an arc is always on a longest path or not. We also characterize these arcs using unionwise permanent solutions and show that knowing such arcs can help us in finding a relative robust path.

Definition 2.8 *An arc is a* **strong arc**, *if it lies on a longest path for all realizations of arc lengths.*

We first give a necessary condition for an arc to be strong which we can check in polynomial time.

Proposition 2.11 *If arc (i,j) is a strong arc, then arc (i,j) is a permanent path between node i and node j.*

Proof
If arc (i,j) is not a permanent path between node i and node j, then in scenario s in which the length of arc (i,j) is at its lower bound and the lengths of all other arcs are at their upper bounds there exists another path from node i to node j, p_{i-j} for which $l_{ij} < \bar{l}_{p_{i-j}}$. Let p be any path using arc (i,j) and p' be the path which has the same arcs as p except for the partial path between node i and node j being p_{i-j}. Then for the above scenario, $l_p^s < l_{p'}^s$. Thus there exists a scenario for which no path using arc (i,j) can be longest. Then arc (i,j) is not a strong arc. □

Next we consider the case where a permanent path exists.

Proposition 2.12 *If there exists a permanent path, an arc is strong if and only if it lies on a permanent path.*

Proof

If an arc lies on a permanent path, then it is strong.

Suppose there exists a permanent path p in the graph. Assume arc (i,j) is not on a permanent path and it is a strong arc. Consider the scenario in which the lengths of all arcs on path p are at upper bounds and the lengths of the remaining arcs are at lower bounds. Since (i,j) is a strong arc, there exists a path p' which uses arc (i,j) and which is a longest path for this scenario. Then we have:

$$\sum_{(m,n)\in p\setminus p'} \bar{l}_{mn} + \sum_{(m,n)\in p\cap p'} \bar{l}_{mn} = \sum_{(m,n)\in p'\setminus p} \underline{l}_{mn} + \sum_{(m,n)\in p\cap p'} \bar{l}_{mn}$$

Since p is a permanent path, we should have $\underline{l}_{mn} = \bar{l}_{mn}$ for all arcs $(m,n) \in p'\setminus p$ and for all arcs $(m,n) \in p\setminus p'$. Then $l_p = l_{p'}$ for all realizations. This implies that p' is also a permanent path. But this contradicts that arc (i,j) does not lie on any permanent path. So (i,j) is not a strong arc. □

We give a mixed integer programming formulation to check whether a given arc is strong or not. This formulation is similar to the formulation we had for the weak arc problem. Before we proceed with the formulation, we give another characterization of strong arcs.

Lemma 2.2 *Arc (i,j) is a strong arc if and only if for all scenarios in which the lengths of all arcs on a path p are at upper bounds and the lengths of the remaining arcs are at lower bounds, there exists a longest path using arc (i,j).*

Proof

If an arc is a strong arc then there exist longest paths using that arc for all of the above scenarios by definition.

Let $P_{(i,j)} = \{p_1, p_2, ...p_m\}$ be the set of paths using arc (i,j). Assume arc

(i,j) is not a strong arc. Then there exists a scenario s, for which no path using arc (i,j) is longest. Let path $p \notin P_{(i,j)}$ be one of the longest paths for this scenario. We have:

$$l_p^s > l_{p_k}^s \quad \forall p_k \in P_{(i,j)}$$

For an arbitrary path $p_k \in P_{(i,j)}$, we have:

$$\sum_{(k,l)\in p\setminus p_k} l_{kl}^s + \sum_{(k,l)\in p\cap p_k} l_{kl}^s > \sum_{(k,l)\in p_k\setminus p} l_{kl}^s + \sum_{(k,l)\in p\cap p_k} l_{kl}^s$$

This implies that:

$$\sum_{(k,l)\in p\setminus p_k} l_{kl}^s + \sum_{(k,l)\in p\cap p_k} \bar{l}_{kl} > \sum_{(k,l)\in p_k\setminus p} l_{kl}^s + \sum_{(k,l)\in p\cap p_k} \bar{l}_{kl}$$

If we set the lengths of arcs in $p\setminus p'$ at upper bounds, we get:

$$\sum_{(k,l)\in p\setminus p_k} \bar{l}_{kl} + \sum_{(k,l)\in p\cap p_k} \bar{l}_{kl} > \sum_{(k,l)\in p_k\setminus p} l_{kl}^s + \sum_{(k,l)\in p\cap p_k} \bar{l}_{kl}$$

If we set the lengths of all arcs that are not on p at lower bounds, we obtain:

$$\sum_{(k,l)\in p\setminus p_k} \bar{l}_{kl} + \sum_{(k,l)\in p\cap p_k} \bar{l}_{kl} > \sum_{(k,l)\in p_k\setminus p} \underline{l}_{kl} + \sum_{(k,l)\in p\cap p_k} \bar{l}_{kl}$$

Since p_k is an arbitrary path in $P_{(i,j)}$, the above inequality is valid for all $p_k \in P_{(i,j)}$. Thus, no path $p_k \in P_{(i,j)}$ is longest for the scenario in which the lengths of all arcs on p are at their upper bounds and the lengths of the remaining arcs are at their lower bounds. □

Corollary 2.5 *An arc is a strong arc if and only if* $\max_{p\in P}\{\bar{l}_p - l_{p^*_{(i,j)}}^{s_p}\} = 0$, *where* s_p *is the scenario in which the lengths of all arcs on* p *are at their upper bounds and the lengths of the remaining arcs at lower bounds and* $p^*_{(i,j)}$ *is the longest path using arc* (i,j) *for scenario* s_p.

Now, we proceed with the formulation of the strong arc problem.

(SA)

$$\max \sum_{(k,l)\in A} \bar{l}_{kl} y_{kl} - (x_i + x_j' + \underline{l}_{ij} + (\bar{l}_{ij} - \underline{l}_{ij}) y_{ij})$$

subject to

$$x_l \geq x_k + \underline{l}_{kl} + (\bar{l}_{kl} - \underline{l}_{kl})y_{kl} \quad \forall (k,l) \in A \; with \; l \leq i$$
$$x_k^{'} \geq x_l^{'} + \underline{l}_{kl} + (\bar{l}_{kl} - \underline{l}_{kl})y_{kl} \quad \forall (k,l) \in A \; with \; k \geq j$$
$$-\sum_{k \in \Gamma^-(l)} y_{kl} + \sum_{h \in \Gamma^+(l)} y_{lh} = b_l \quad l = 1,2,..,n$$
$$y_{kl} \in \{0,1\} \quad \forall (k,l) \in A \qquad x_l, x_l^{'} \geq 0 \quad l = 1,2,..,n$$

Different from the weak arc problem, we have variable $x_k^{'}$ which corresponds to the the length of the longest path from node k to node t and the constraints which specify these distances:

$$x_k^{'} \geq x_l^{'} + \underline{l}_{kl} + (\bar{l}_{kl} - \underline{l}_{kl})y_{kl} \quad \forall (k,l) \in A \; with \; k \geq j$$

Next, we give a characterization of strong arcs using the formulation SA.

Theorem 2.4 *Arc (i,j) is a strong arc if and only if SA has an optimal objective value of 0.*

Proof

Simply follows from Corollary 2.5. □

The above formulation searches among all paths. In fact we can restrict it to the paths that do not use arc (i,j).

Proposition 2.13 *Arc (i,j) is a strong arc if and only if* $\max_{p \in P \setminus P_{(i,j)}} \{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} \leq 0$.

Proof

An arc is a strong arc if and only if $\max_{p \in P}\{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} = 0$, so we need to show that $\max_{p \in P}\{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} = 0$ if and only if $\max_{p \in P \setminus P_{(i,j)}}\{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} \leq 0$.

If $\max_{p \in P}\{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} = 0$, then $\max_{p \in P \setminus P_{(i,j)}}\{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} \leq 0$.

Assume $\max_{p \in P \setminus P_{(i,j)}} \{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} \leq 0$. We can easily show that $\max_{p \in P_{(i,j)}} \{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} = 0$. Then $\max_{p \in P} \{\bar{l}_p - l^{s_p}_{p^*_{(i,j)}}\} = 0$. □

Let SA' be the formulation formed by adding the constraint $y_{ij} = 0$ to formulation SA. Using Proposition 2.13, we have:

Proposition 2.14 *Arc (i,j) is a strong arc if and only if SA' has an optimal objective value ≤ 0.*

Characterization of Strong Arcs Using Unionwise Permanent Sets

In case all arcs have non degenerate lengths, we give a necessary and sufficient condition for an arc to be strong using a special subset of weak paths. This subset contains all weak paths each of which is a unique longest path for some scenario. We show that a given arc is strong if and only if it lies on all of these paths. Moreover, we see that there exists a robust path using all of the strong arcs.

Definition 2.9 *A set of paths is a* **unionwise permanent set** *if for each realization there exists a longest path in this set.*

Definition 2.10 *A unionwise permanent set is a* **minimum unionwise permanent set** *if it is no longer a unionwise permanent set when a path is removed.*

Lemma 2.3 *There exists no scenario for which a given weak path is the unique longest path if and only if this path is not the unique longest path when the lengths of all arcs on it are at their upper bounds and the lengths of the remaining arcs are at their lower bounds.*

Proof

Since the path is not the unique longest path for any scenario, it won't be the

unique longest path for the stated scenario.

If a weak path p, is not the unique longest path when the lengths of all arcs on p are at upper bounds and the lengths of the remaining arcs are at lower bounds, there exists another path p' such that

$$\sum_{(i,j)\in p\setminus p'} \bar{l}_{ij} + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij} = \sum_{(i,j)\in p'\setminus p} \underline{l}_{ij} + \sum_{(i,j)\in p\cap p'} \bar{l}_{ij}$$

For arbitrary lengths of arcs in $p \cap p'$, we have:

$$\sum_{(i,j)\in p\setminus p'} \bar{l}_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij} = \sum_{(i,j)\in p'\setminus p} \underline{l}_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij}$$

Since $\sum_{(i,j)\in p\setminus p'} \bar{l}_{ij} \geq \sum_{(i,j)\in p\setminus p'} l_{ij}$ and $\sum_{(i,j)\in p'\setminus p} \underline{l}_{ij} \leq \sum_{(i,j)\in p'\setminus p} l_{ij}$, we have:

$$\sum_{(i,j)\in p\setminus p'} l_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij} \leq \sum_{(i,j)\in p'\setminus p} l_{ij} + \sum_{(i,j)\in p\cap p'} l_{ij}$$

that is $l_p \leq l_{p'}$ for all scenarios, so p can never be the unique longest path. □

From now on, we assume that all arcs have non degenerate lengths.

Lemma 2.4 *In any graph, there exists at least one path which is the unique longest path for some scenario.*

Proof

Assume no such path exists. Pick an arbitrary path p_1. From the proof of the previous lemma, it follows that there exists a path $p_2 \neq p_1$ such that $l_{p_1} \leq l_{p_2}$ for all scenarios. Since p_2 is not the unique longest path for any scenario, there exists another path $p_3 \neq p_2, p_1$ such that $l_{p_2} \leq l_{p_3}$ for all scenarios. Repeating this argument for all paths in the graph, since no path can be repeated (it is not possible for two paths to have the same lengths for all scenarios by perturbing the length of an arc which is on one of the paths, but not both, we can have different lengths), and the number of paths is finite, we end up with a sequence of paths:

$$l_{p_1} \leq l_{p_2} \leq l_{p_3} \ldots \leq l_{p_k}$$

where k is the number of paths in the graph. Since the above inequalities hold for all realizations, p_k is a permanent path. But a permanent path is the unique longest path when the lengths of all arcs are at lower bounds. This contradicts that no path is the unique longest path for any scenario. □

Lemma 2.5 *If a path p is never the unique longest path, there exists a path p' such that $l_p \leq l_{p'}$ for all scenarios and p' is the unique longest path for some scenario.*

Proof

If p is not the unique longest path for any scenario, there exists another path p' such that $l_p \leq l_{p'}$ for all scenarios. If $p' \neq p$ is the unique longest path for some scenario, we are done. Assume not, then there exists another path $p'' \neq p', p$ such that $l_{p'} \leq l_{p''}$ for all scenarios. If we repeat this argument, we will end up with a sequence of paths that are not the unique longest paths for any scenario. Again, since no path can be repeated , and the number of paths is finite, we will either find a unique longest path and stop or we will enumerate all paths. Since there exists at least one path which is the unique longest path for some scenario, we will stop with a unique longest path. □

Now we give a characterization for minimum unionwise permanent set and show that it is unique.

Theorem 2.5 *If P' is the set of paths each of which is the unique longest path when the lengths of all arcs on this path are at their upper bounds and the lengths of the remaining arcs are at their lower bounds, then P' is the minimum unionwise permanent set.*

Proof

Assume P' is not a unionwise permanent set. Then there exists a scenario s, for which no path in P' is longest. Then there exists a path $p \in P \backslash P'$ such that $l_p^s > l_{p_i}^s \quad \forall p_i \in P'$. But since $p \in P \backslash P'$, there exists a path $p_j \in P'$ such that $l_p \leq l_{p_j}$ for all scenarios. This contradicts that p is longer than all paths

in P'. So P' is a unionwise permanent set.

P' is minimum since any path in P' is the unique longest path for some scenario. □

Corollary 2.6 *P^* is a unionwise permanent set if and only if $P' \subseteq P^*$.*

Corollary 2.7 *The minimum unionwise permanent set is unique.*

The following proposition gives a characterization of strong arcs using the minimum unionwise permanent set.

Proposition 2.15 *An arc is strong if and only if all paths in the minimum unionwise permanent set share that arc.*

Proof

If an arc (i, j) is strong it is on a longest path for all scenarios. Since each path in the minimum unionwise permanent set is the unique longest path for some scenario, (i, j) should lie on all of them.

If an arc (i, j) is shared by all paths in the minimum unionwise permanent set, then it is on a longest path for all scenarios, thus it is strong. □

It follows that a given arc is strong if and only if the set of all paths that share that arc is a unionwise permanent set, since a set of paths is a unionwise permanent set if and only if it contains the minimum unionwise permanent set.

Proposition 2.16 *There exists a relative robust path in the minimum unionwise permanent set P'.*

Proof

Assume none exists. Then there is a relative robust path $p \in P \backslash P'$. Since p is

not the unique longest path for any scenario there exists a path $p' \in P'$ such that $l_p \leq l_{p'}$ for all scenarios. Consider the relative worst case scenario for path p'. we have:

$$l^{s^r_{p'}}_{p^*(s^r_{p'})} - l^{s^r_{p'}}_{p'} \leq l^{s^r_{p'}}_{p^*(s^r_{p'})} - l^{s^r_{p'}}_{p}$$

The left hand side of the inequality is equal to the worst case deviation for path p', and the right hand side of the inequality is less than or equal to the worst case deviation for path p. That is, $d_{p'} \leq d_p$. Since path p is a robust path, this inequality holds as an equality and p' is also a robust path. □

Corollary 2.8 *There exists a relative robust path which uses all strong arcs.*

Proof

Follows from Proposition 2.15 and Proposition 2.16. □

Chapter 3

Arc Problems On Layered Graphs

In this chapter, we consider the longest path problem with interval arc lengths on a special class of graphs, called layered graphs. We would like to investigate the arc problems on these graphs.

A layered graph with m layers is defined as $G = (V, A)$ where $V = \{s\} \cup \{(k, l) : k = 1, 2, ..., m; l = 1, 2, .., w_k\} \cup \{t\}$. Node s is the root and node t is the sink, nodes (k, l) for $l = 1, 2, .., w_k$ constitute layer k and w_k is the width of layer k. Arcs exist only from node s to nodes in layer 1, from nodes in layer m to node t and from nodes in layer k to nodes in layer $k+1$ for all $k = 1, 2, .., m-1$. The width of the graph w is computed as $w = \max_{k=1,2,..,m} w_k$. Figure 3.1 shows an example of a layered graph with width 2.

In fact, this class is not restrictive since any directed acyclic graph can be turned into a layered graph if we add dummy nodes and arcs. In Figure 3.2, there is a complete graph with 5 nodes represented as a layered graph. Nodes 1,2,..,5 stand for the original nodes. The remaining nodes are dummy nodes. To represent arc (1,3), we add node 13 and arcs (1,13) and (13,3). We set the length of arc (1,13) to be 0, and the length of arc (13,3) to be the length of

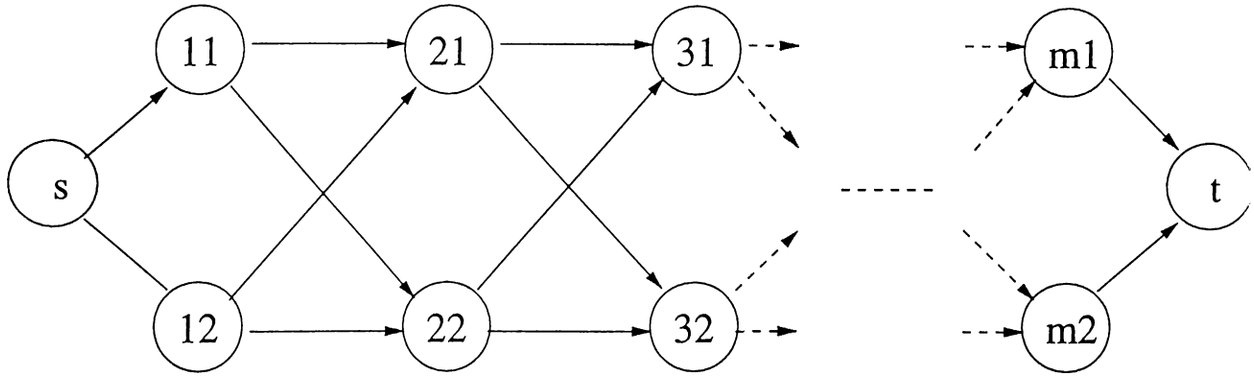

Figure 3.1: An m layered graph with width 2

arc (1,3) in the original graph. Since we can reach node 13 only from node 1 and we can use arc (13,3) only to reach node 3, the partial path $1 \rightarrow 13 \rightarrow 3$ stands for arc (1,3), and its length is the same as the length of arc (1,3). The other arcs are represented in a similar way by adding dummy nodes and arcs. Let $l_{i,j}$ denote the length of arc (i,j) in the complete directed acyclic graph G and $l'_{i,j}$ denote the length of arc (i,j) in the corresponding layered graph G'. The lengths of the arcs are as follows: $l'_{i,i+1} = l_{i,i+1}$, $l'_{ij,j} = l_{i,j}$ and the lengths of the remaining arcs are 0.

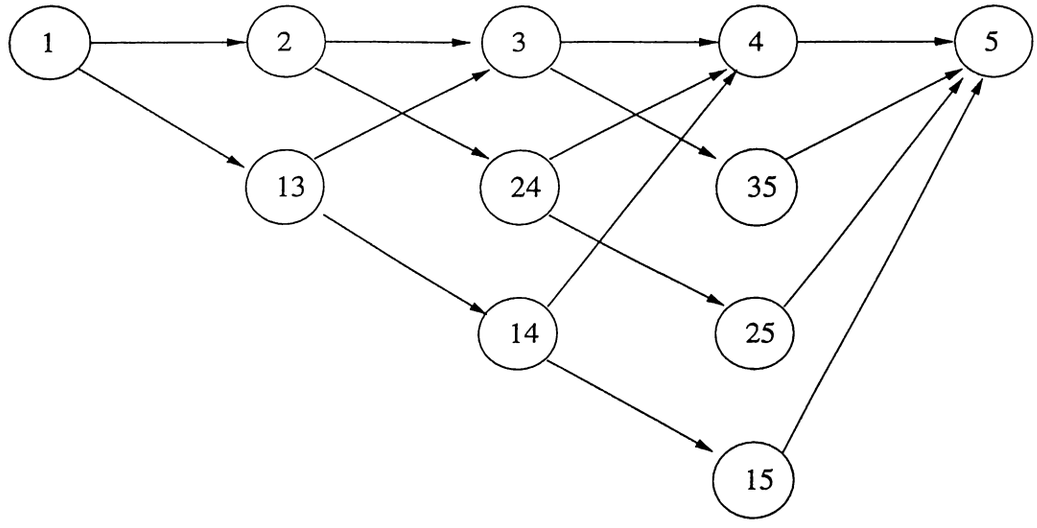

Figure 3.2: Representation of a complete directed acyclic graph with 5 nodes as a layered graph

We developed linear time algorithms for the weak arc and strong arc problems on layered graphs with width two. For layered graphs with arbitrary width, we also have algorithms to check whether arcs that are incident at the beginning node and the ending node are weak and strong in polynomial time. However we do not have a procedure to check the remaining arcs when the width is larger than two. Still, we have a necessary condition for these arcs to be weak and strong and it can be checked in polynomial time.

From now on, we assume that $w_k = w \ \forall k$ and all arcs exist, but unless otherwise stated, all the propositions we present below are valid when these assumptions are relaxed.

3.1 Weak Arcs

We now consider the weak arc problem on layered graphs. First we investigate a trivial case of the problem. Let $l_{(i,j),(k,l)}$ denote the length of arc $((i,j),(k,l))$.

Proposition 3.1 *In a layered graph, if for each layer $k < m$, there exists c_k such that $c_k \in [\underline{l}_{(k,l),(k+1,j)}, \overline{l}_{(k,l),(k+1,j)}]$ for all $l, j = 1, 2, ..., w$, $c_s \in [\underline{l}_{s,1j}, \overline{l}_{s,1j}]$ for all $j = 1, 2, ..., w$ and $c_t \in [\underline{l}_{ml,t}, \overline{l}_{ml,t}]$ for all $l = 1, 2, ..., w$, then all arcs are weak.*

Proof
Set the lengths of arcs $((k,l),(k+1,j))$ for all $l, j = 1, 2, .., w$ at c_k for all $k = 1, 2, .., m-1$, the lengths of arcs $(s, 1j)$ for all $j = 1, 2, .., w$ at c_s and the lengths of arcs (ml, t) for all $l = 1, 2, .., w$ at c_t. Then all paths in the graph have the same length. □

Proposition 3.2 *Consider arc $((i,1),(i+1,1))$. Assume all nodes in layer i are reachable from all nodes in layer $i-1$ and all nodes in layer $i+2$ are reachable from all nodes in layer $i+1$. If there exists c_{i-1} such that $c_{i-1} \in [\underline{l}_{(i-1,l),(i,j)}, \overline{l}_{(i-1,l),(i,j)}]$ for all $l, j = 1, 2, ..., w$ and there exists c_{i+1} such*

that $c_{i+1} \in [\underline{l}_{(i+1,l),(i+2,j)}, \overline{l}_{(i+1,l),(i+2,j)}]$ *for all* $l, j = 1, 2, ..., w$, *and* $\overline{l}_{(i,1),(i+1,1)} \geq \max_{j,k=2,3,..,w} \underline{l}_{(i,j),(i+1,k)}$, *then* $((i,1),(i+1,1))$ *is a weak arc.*

Proof

Consider a scenario in which the lengths of arcs $((i-1,l),(i,j))$ for all $l, j = 1, 2, .., w$ are at c_{i-1}, the lengths of arcs $((i+1,l),(i+2,j))$ for all $l, j = 1, 2, .., w$ are at c_{i+1}, the length of arc $((i,1),(i+1,1))$ is at its upper bound, and the lengths of arcs $((i,l),(i+1,j))$ for all $l, j = 2, 3, .., w$ are at lower bounds. Then the lengths of longest paths from node s to all nodes in layer i are the same since all nodes in layer i are reachable from all nodes in layer $i-1$. Similarly, the lengths of the longest paths from all nodes in layer $i+1$ to node t are the same. So if $\overline{l}_{(i,1),(i+1,1)} \geq \max_{j,k=2,3,..,w} \underline{l}_{(i,j),(i+1,k)}$, then there exists a longest path using arc $((i,1),(i+1,1))$ for this scenario. □

3.1.1 Arcs Incident at Nodes s and t

We next present an algorithm which can decide whether arcs $(s, 1k)$ for $k = 1, 2, .., w$ and (mk, t) for $k = 1, 2, .., w$ are weak or not. The procedure is given for arc $(s, 11)$. But we can also decide for arcs emanating from nodes in layer m by using the same procedure for the mirror version of the graph.

We will start with the scenario in which the lengths of all arcs that can possibly be on a path with $(s, 11)$ are set at their upper bounds and the lengths of the remaining arcs are set at their lower bounds. Our algorithm will construct a weak path using arc $(s, 11)$, if such a path exists, by eliminating at each iteration arcs that cannot possibly be on a weak path with arc $(s, 11)$.

In the first iteration, we consider the subgraph generated by node s and the nodes in layers 1 and 2. We find the longest paths from node s to all nodes in layer 2. We favor paths that use arc $(s, 11)$ in finding the longest paths. We consider three cases:

Case 1: If all the longest paths from node s to nodes in layer 2 use arc $(s, 11)$, then this arc is a weak arc. With the current scenario, one of the longest paths will definitely use arc $(s, 11)$.

Case 2: If none of the longest paths from node s to nodes in layer 2 uses arc $(s, 11)$, then $\bar{l}_{s,11} + \bar{l}_{11,2k} < \max_{j=1,2,..,w} \underline{l}_{s,1j} + \underline{l}_{1j,2k}$ for all $k = 1, 2, .., w$. This proves that there cannot be a longest path using arc $(s, 11)$ for any scenario. Thus, this arc is not weak.

Case 3: If some of the longest paths from node s to nodes in layer 2 use arc $(s, 11)$, then arc $(s, 11)$ is a weak arc if and only if there exists a scenario in which there exists a longest path using one of these longest partial paths. We shrink the graph between node s and layer 2. We set the length of arcs $(s, 2j)$ to be the length of the longest paths form node s to node $2j$ for all $j = 1, 2, ..w$. For a fixed j, if the longest path from node s to node $2j$ uses arc $(s, 11)$, we call such a partial path "candidate partial path" and set the lengths of arcs $(2j, 3l)$ at their upper bounds for all $l = 1, 2, ..w$ since these are arcs that can possibly be on a weak path with arc $(s, 11)$. Arc $(s, 11)$ is a weak arc if and only if one of the candidate partial paths is weak in the shrunk graph. On the other hand, if the longest path from node s to node $2j$ does not use arc $(s, 11)$, we call such a partial path "non-candidate partial path" and set the lengths of arcs $(2j, 3l)$ at their lower bounds for all $l = 1, 2, ..w$ since these arcs cannot be on a weak path with arc $(s, 11)$.

Now the problem is to check whether any of the candidate partial paths is weak in the shrunk graph. We continue by finding the longest paths from node s to all nodes in layer 3. If all of these paths use candidate partial paths, then arc $(s, 11)$ is a weak arc. If none of these paths uses a candidate partial path, then arc $(s, 11)$ is not weak. If some of them use candidate partial paths, we shrink the graph between node s and layer 3, and determine the new candidate partial paths and continue with the shrunk graph.

So at each iteration of the algorithm, there are 3 possible conclusions. One

possibility is that arc $(s, 11)$ may turn out to be weak. By enlarging our current graph to its original version keeping the arc lengths as they are set by the procedure, we can find a longest path using arc $(s, 11)$. This corresponds to Case 1. Another possibility is that we might conclude that arc $(s, 11)$ cannot be weak, which is Case 2. Other than these two cases, we may be in Case 3, in which we need to go through the same steps at least for one more iteration. In the worst case, we can shrink the graph till layer m, and conclude that arc $(s, 11)$ is weak or not in that graph.

Proposition 3.3 *The above procedure can decide whether arc* $(s, 11)$ *is weak in* $O(mw^2)$ *time in an* m *layered graph with width* w.

Proof At each iteration $k-1$, finding the longest paths from node s to all nodes in layer k takes $O(w^2)$ time, since for each node in layer k, we compare the lengths of w paths and we have w nodes in layer k. In the worst case we go through m iterations. □

Example 1

Below is a 3 layered graph with width 3 in Figure 3.3. We want to know whether arc $(s, 11)$ is weak or not.

We first consider the subgraph generated by node s and layers 1 and 2, which is given in Figure 3.4. We set the lengths of arcs $(s, 11)$, (11,21), (11,22) and (11,23) at upper bounds and the lengths of the remaining arcs in this subgraph at lower bounds. The dashed arcs refer to arcs whose lengths are at lower bounds. Next we find the longest paths from node s to all nodes in layer 2.

The longest paths form node s to nodes 21 and 22 use arc $(s, 11)$ and the longest path from node s to node 23 uses arc $(s, 13)$. We shrink the graph and add layer 3 to obtain the graph in Figure 3.5. We set the lengths of arcs (23,31), (23,32) and (23,33) at lower bounds and the remaining arcs in the

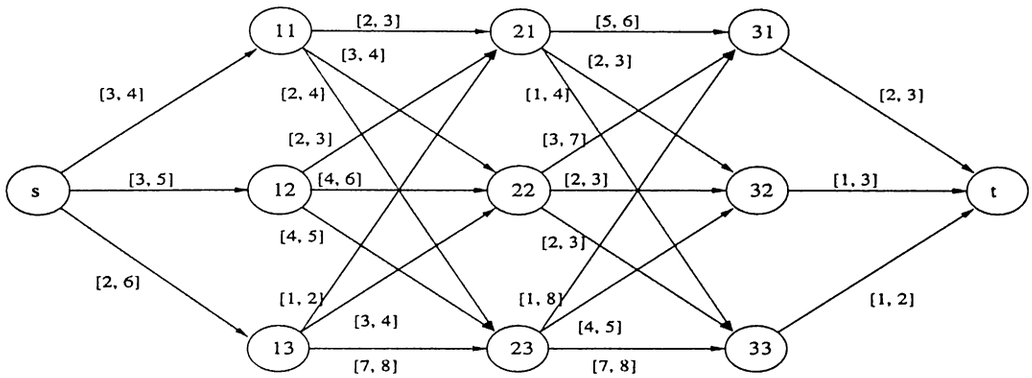

Figure 3.3: 3 layered graph with width 3 on which we check whether arc $(s, 11)$ is weak or not

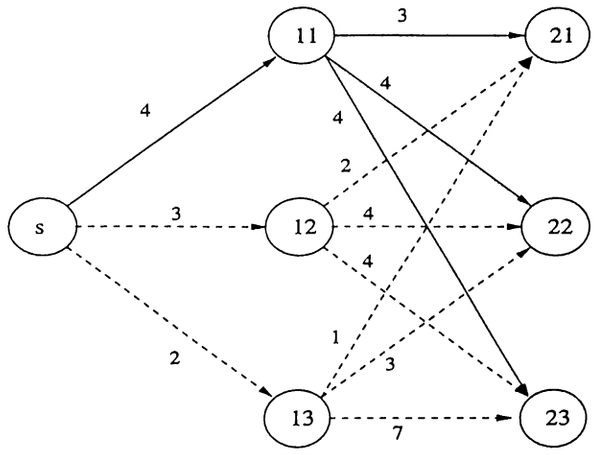

Figure 3.4: Subgraph generated by node s and layers 1 and 2

subgraph at upper bounds.

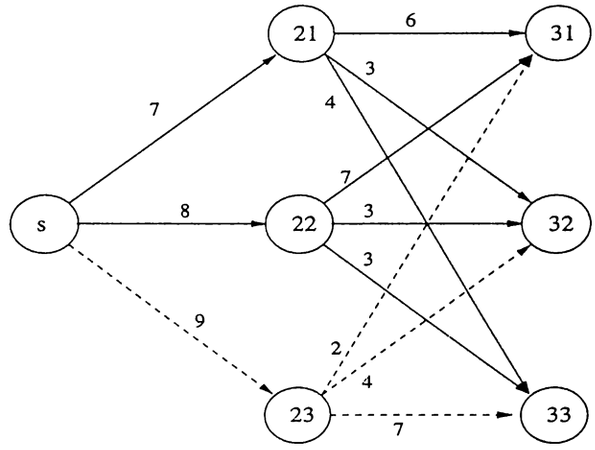

Figure 3.5: Subgraph shrunk between node s and layer 2

We find the longest paths from node s to all nodes in layer 3. Only, the longest path from node s to node 31 uses arc $(s, 11)$. We shrink the graph again, add node t and set the lengths of arcs $(32, t)$ and $(33, t)$ at lower bounds. The resulting graph is given in Figure 3.6.

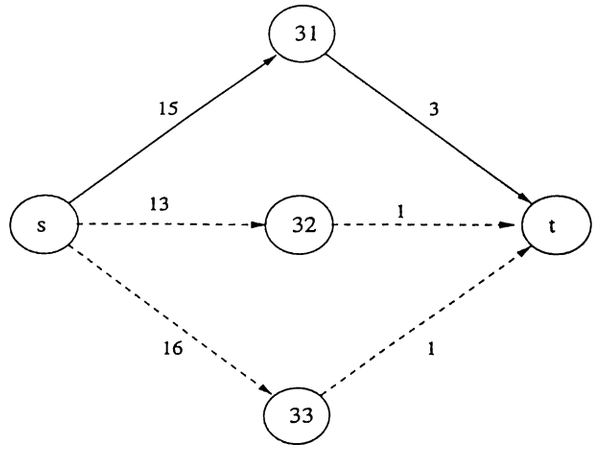

Figure 3.6: The final graph shrunk between node s and layer 3

There exists a longest path from node s to node t using arc $(s, 11)$ in the final graph. So we conclude that arc $(s, 11)$ is a weak arc. The path consisting of arcs $(s, 11)$, (11,22), (22,31) and $(31, t)$ is a weak path.

3.1.2 Intermediate Arcs

Now, we give a trivial necessary condition for an intermediate arc to be weak in a layered graph, which can be checked using the above procedure.

Proposition 3.4 *If arc* $(i1, j1)$ *is weak in a layered graph, then it is weak in the subgraph generated by node* s*, layers 1 up to* i *and node* $j1$*, and it is weak in the subgraph generated by node* $i1$*, layers* j *up to* m *and node* t*.*

We next consider the weak arc problem on an m layered graph with width 2, which is defined as $G = (V, A)$ where $V = \{s\} \cup \{(k, i) : i = 1, 2; k = 1, 2, ..., m\} \cup \{t\}$ and $A = \{(s, 11), (s, 12), (m1, t), (m2, t)\} \cup \{((k, i), (k+1, j)) : k = 1, 2, ..., m-1; i = 1, 2; j = 1, 2\}$.

First we give a necessary and sufficient condition for an arc $(i1, j1)$ to be weak in layered graphs with width 2. Let a_i^r be the length of the longest path from node s to node i in scenario r, and b_i^r be the length of the longest path from node i to node t in scenario r.

Proposition 3.5 *Arc* $(i1, j1)$ *is a weak arc if and only if there exists a longest path using this arc for the scenario* r^**, in which* $a_{i1}^{r^*} - a_{i2}^{r^*} = \max_{r \in S}\{a_{i1}^r - a_{i2}^r\}$ *and* $b_{j1}^{r^*} - b_{j2}^{r^*} = \max_{r \in S}\{b_{j1}^r - b_{j2}^r\}$ *and the length of arc* $(i1, j1)$ *is at its upper bound and the lengths of arcs* $(i1, j2), (i2, j1)$ *and* $(i2, j2)$ *are at lower bounds.*

Proof

If arc $(i1, j1)$ is on a longest path for scenario r^*, then by definition it is a weak arc.

If arc $(i1, j1)$ is a weak arc, then there exists a scenario r, in which there

exists a longest path using this arc. Moreover, if we set the length of arc $(i1, j1)$ at its upper bound and the lengths of arcs $(i1, j2), (i2, j1)$ and $(i2, j2)$ at lower bounds, there still exists a longest path using arc $(i1, j1)$. So we have the following inequalities:

$$a^r_{i1} + \bar{l}_{i1,j1} + b^r_{j1} - a^r_{i1} - \underline{l}_{i1,j2} - b^r_{j2} = \bar{l}_{i1,j1} + b^r_{j1} - \underline{l}_{i1,j2} - b^r_{j2} \geq 0 \quad (1)$$

$$a^r_{i1} + \bar{l}_{i1,j1} + b^r_{j1} - a^r_{i2} - \underline{l}_{i2,j1} - b^r_{j1} = \bar{l}_{i1,j1} + a^r_{i1} - \underline{l}_{i2,j1} - a^r_{i2} \geq 0 \quad (2)$$

$$a^r_{i1} + \bar{l}_{i1,j1} + b^r_{j1} - a^r_{i2} - \underline{l}_{i2,j2} - b^r_{j2} \geq 0 \quad (3)$$

For scenario r^*, we know:

$$a^{r^*}_{i1} - a^{r^*}_{i2} \geq a^r_{i1} - a^r_{i2} \quad (4)$$

$$b^{r^*}_{j1} - b^{r^*}_{j2} \geq b^r_{j1} - b^r_{j2} \quad (5)$$

Inequalities (1) and (5) imply that:

$$\bar{l}_{i1,j1} + b^{r^*}_{j1} - \underline{l}_{i1,j2} - b^{r^*}_{j2} \geq 0$$

inequalities (2) and (4) imply:

$$\bar{l}_{i1,j1} + a^{r^*}_{i1} - \underline{l}_{i2,j1} - a^{r^*}_{i2} \geq 0$$

and finally inequalities (3), (4) and (5) imply that:

$$a^{r^*}_{i1} + \bar{l}_{i1,j1} + b^{r^*}_{j1} - a^{r^*}_{i2} - \underline{l}_{i2,j2} - b^{r^*}_{j2} \geq 0$$

So, for scenario r^*, there exists a longest path using arc $(i1, j1)$. □

To check whether arc $(i1, j1)$ is a weak arc or not, we need to find the scenario r^*. From this point on, we will only look for a scenario r_{i1} in which $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2} = \max_{r \in S}\{a^r_{i1} - a^r_{i2}\}$. To do this, it is enough to consider the subgraph generated by layers 1 up to i and node s. Once we can develop an algorithm to find scenario r_{i1}, we can use it to find scenario r_{j1}, for which $b^{r_{j1}}_{j1} - b^{r_{j1}}_{j2} = \max_{r \in S}\{b^r_{j1} - b^r_{j2}\}$ using the mirror version of the subgraph generated by layers j up to m and node t. Since scenario r_{i1} sets the lengths of arcs among node s and layers 1 up to i, and scenario r_{j1} sets the lengths of

arcs among layers j up to m and node t, these two scenarios will not conflict with each other, and we can construct r^* using these scenarios.

Now, we will show that finding the scenario r_{i1} is equivalent to finding a path from node s to node $i1$ such that, the maximum difference $a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$ is attained when the lengths of all arcs on this path are at upper bounds and the lengths of the remaining arcs are at lower bounds.

Proposition 3.6 *Let p be the longest path from node s to node $i1$ for scenario r_{i1}. Consider the scenario r', in which the lengths of all arcs on p are at upper bounds and the remaining arcs are at lower bounds. Then, $a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}} = a_{i1}^{r'} - a_{i2}^{r'}$.*

Proof
When we move from scenario r_{i1} to scenario r', it is easy to show that path p has the higher increase in length, since no other path can have all arcs of p. This proves that $a_{i1}^{r'} - a_{i2}^{r'} \geq a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$. So $a_{i1}^{r'} - a_{i2}^{r'} = a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$. □

The above proposition suggests that we can restrict the set of scenarios to search to a smaller set of scenarios in each of which the lengths of all arcs on a path from node s to node $i1$ are at upper bounds and the lengths of the remaining arcs are at lower bounds. We can take the length of this path to be the length of the longest path from node s to node $i1$, since if it is not the longest path for this scenario, it is clear that we cannot attain $a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$ for this path. We developed a procedure to compute $a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$.

We start with node 21. We first consider path $s \rightarrow 11 \rightarrow 21$ and set the lengths of arcs on this path at upper bounds and the lengths of the other arcs at lower bounds. We find the difference between the length of this path and the length of the longest path to node 22, which is d_{21}^1. We repeat the same calculations for path $s \rightarrow 12 \rightarrow 21$ and find d_{21}^2. We pick the maximum difference d_{21}. We do the same computations for node 22 and find d_{22}. Then we shrink the graph among node s and layer 2. We set the upper length of

arc $(s, 21)$ to be d_{21} and the upper length of arc $(s, 22)$ to be d_{22}. The lower lengths for both arcs are 0. Next, we repeat the same steps for nodes in layer 3, and continue till we reach layer i.

Procedure ComputeMaxDifference

for k=2 to i do

1. $d^1_{k1} = \overline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,1)} - \max\{\overline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,2)}, \underline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,2)}\}$
   $d^2_{k1} = \overline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,1)} - \max\{\overline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,2)}, \underline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,2)}\}$
   $d_{k1} = \max\{d^1_{k1}, d^2_{k1}\}$

2. $\overline{l}_{s,(k,1)} = d_{k1}$
   $\underline{l}_{s,(k,2)} = 0$

3. $d^1_{k2} = \overline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,2)} - \max\{\overline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,1)}, \underline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,1)}\}$
   $d^2_{k2} = \overline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,2)} - \max\{\overline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,1)}, \underline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,1)}\}$
   $d_{k2} = \max\{d^1_{k2}, d^2_{k2}\}$

4. $\overline{l}_{s,(k,2)} = d_{k2}$
   $\underline{l}_{s,(k,1)} = 0$

In the procedure, $d_{k1} = a^{r_{k1}}_{k1} - a^{r_{k1}}_{k2}$ and $d_{k2} = a^{r_{k2}}_{k2} - a^{r_{k2}}_{k1}$. So at iteration $k-1$, d_{k1} and d_{k2} are computed using $d_{k-1,1}$ and $d_{k-1,2}$. In fact, at the $k-1$st iteration, we work with the shrunk graph in Figure 3.7.

The lengths of the arcs are as follows: $\underline{l}_{s,(k-1,1)} = 0, \overline{l}_{s,(k-1,1)} = d_{k-1,1}, \underline{l}_{s,(k-1,2)} = 0$ and $\overline{l}_{s,(k-1,2)} = d_{k-1,2}$.

Proposition 3.7 *The above procedure computes* $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2}$.

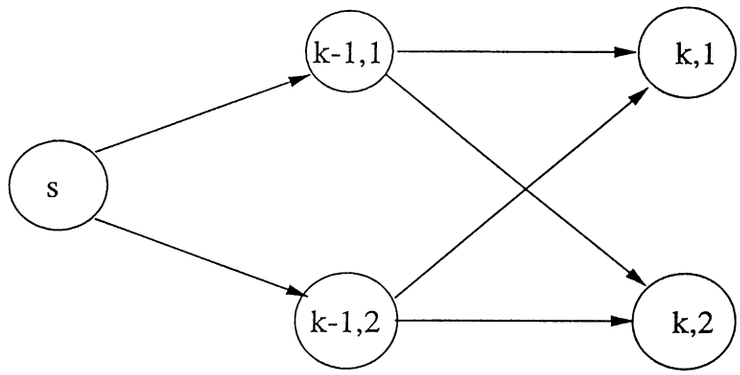

Figure 3.7: Graph shrunk between node s and layer $k-1$

Proof

The proof is done by induction on k.

Let $k = 2$ and $l = 1$. We have two paths from node s to node 21. Consider scenario r_{21}. Proposition 3.6 implies that in this scenario, the arcs on one of the paths from s to 21 are set at upper bounds and the remaining arcs are at lower bounds. Consider the equalities below:

$$d^1_{21} = \bar{l}_{s,11} + \bar{l}_{11,21} - \max\{\bar{l}_{s,11} + \underline{l}_{11,22}, \underline{l}_{s,12} + \underline{l}_{12,22}\}$$

$$d^2_{21} = \bar{l}_{s,12} + \bar{l}_{12,21} - \max\{\bar{l}_{s,12} + \underline{l}_{12,22}, \underline{l}_{s,11} + \underline{l}_{11,22}\}$$

$$d_{21} = \max\{d^1_{21}, d^2_{21}\}$$

The first inequality computes the difference between the length of path $s \rightarrow 11 \rightarrow 21$ and the length of the longest path from s to 22 when the arcs on path $s \rightarrow 11 \rightarrow 21$ are at upper bounds and the remaining arcs are at lower bounds. The second inequality does the same for the second path which is $s \rightarrow 12 \rightarrow 21$. We pick the one with the maximum difference in the third inequality. So $d_{21} = a^{r_{21}}_{21} - a^{r_{21}}_{22}$.

Assume that the algorithm computes differences $d_{k-1,1} = a^{r_{k-1,1}}_{k-1,1} - a^{r_{k-1,1}}_{k-1,2}$ and $d_{k-1,2} = a^{r_{k-1,2}}_{k-1,2} - a^{r_{k-1,2}}_{k-1,1}$ correctly. We will show that it can compute difference $a^{r_{k1}}_{k1} - a^{r_{k1}}_{k2}$. Let r_{k1} be the scenario for which $a^{r_{k1}}_{k1} - a^{r_{k1}}_{k2} = \max_{r \in S}\{a^r_{k1} - a^r_{k2}\}$. Then

$$a^{r_{k1}}_{k1} - a^{r_{k1}}_{k2} = \max\{a^{r_{k1}}_{k-1,1} + \bar{l}_{(k-1,1),(k,1)}, a^{r_{k1}}_{k-1,2} + \bar{l}_{(k-1,2),(k,1)}\}$$

$$- \max\{a^{r_{k1}}_{k-1,1} + \underline{l}_{(k-1,1),(k,2)}, a^{r_{k1}}_{k-1,2} + \underline{l}_{(k-1,2),(k,2)}\}$$

$$= \max\{\min\{\overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,1),(k,2)}, a^{r_{k1}}_{k-1,1} + \overline{l}_{(k-1,1),(k,1)} - a^{r_{k1}}_{k-1,2} - \underline{l}_{(k-1,2),(k,2)}\},$$

$$\min\{a^{r_{k1}}_{k-1,2} + \overline{l}_{(k-1,2),(k,1)} - a^{r_{k1}}_{k-1,1} - \underline{l}_{(k-1,1),(k,2)}, \overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,2),(k,2)}\}\}$$

Consider d^1_{k1} and d^2_{k1} computed by the procedure.

$$d^1_{k1} = \overline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,1)} - \max\{\overline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,2)}, \underline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,2)}\}$$

$$= \min\{\overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,1),(k,2)}, \overline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,1)} - \underline{l}_{s,(k-1,2)} - \underline{l}_{(k-1,2),(k,2)}\}$$

$$= \min\{\overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,1),(k,2)}, d_{k-1,1} + \overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,2),(k,2)}\}$$

$$= \min\{\overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,1),(k,2)}, a^{r_{k-1,1}}_{k-1,1} - a^{r_{k-1,1}}_{k-1,2} + \overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,2),(k,2)}\}$$

$$\geq \min\{\overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,1),(k,2)}, a^{r_{k1}}_{k-1,1} - a^{r_{k1}}_{k-1,2} + \overline{l}_{(k-1,1),(k,1)} - \underline{l}_{(k-1,2),(k,2)}\}$$

$$d^2_{k1} = \overline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,1)} - \max\{\overline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,2)}, \underline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,2)}\}$$

$$= \min\{\overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,2),(k,2)}, \overline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,1)} - \underline{l}_{s,(k-1,1)} - \underline{l}_{(k-1,1),(k,2)}\}$$

$$= \min\{\overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,2),(k,2)}, d_{k-1,2} + \overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,1),(k,2)}\}$$

$$= \min\{\overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,2),(k,2)}, a^{r_{k-1,2}}_{k-1,2} - a^{r_{k-1,2}}_{k-1,1} + \overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,1),(k,2)}\}$$

$$\geq \min\{\overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,2),(k,2)}, a^{r_{k1}}_{k-1,2} - a^{r_{k1}}_{k-1,1} + \overline{l}_{(k-1,2),(k,1)} - \underline{l}_{(k-1,1),(k,2)}\}$$

$$d_{k1} = \max\{d^1_{k1}, d^2_{k1}\}$$

So $d_{k1} \geq a^{r_{k1}}_{k1} - a^{r_{k1}}_{k2}$. Since $a^{r_{k1}}_{k1} - a^{r_{k1}}_{k2}$ is the maximum difference, we have $d_{k1} = a^{r_{k1}}_{k1} - a^{r_{k1}}_{k2}$. □

Some of the d_{kl}'s may turn out to be negative. Though the interval $[0, d_{kl}]$ does not make much sense in this case, this does not cause any problem as far as the computations are concerned.

Proposition 3.8 *We can compute $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2}$ and $b^{r_{j1}}_{j1} - b^{r_{j1}}_{j2}$ in $O(m)$ time using the above procedure.*

Proof

For each node, we make two comparisons, which takes constant time. After we do the mirror version, total number of steps repeated is in $O(m)$. □

After we compute $a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$ and $b_{j1}^{r_{j1}} - b_{j2}^{r_{j1}}$, we construct the following graph in Figure 3.8, in which we check whether arc $(i1, j1)$ is weak or not.

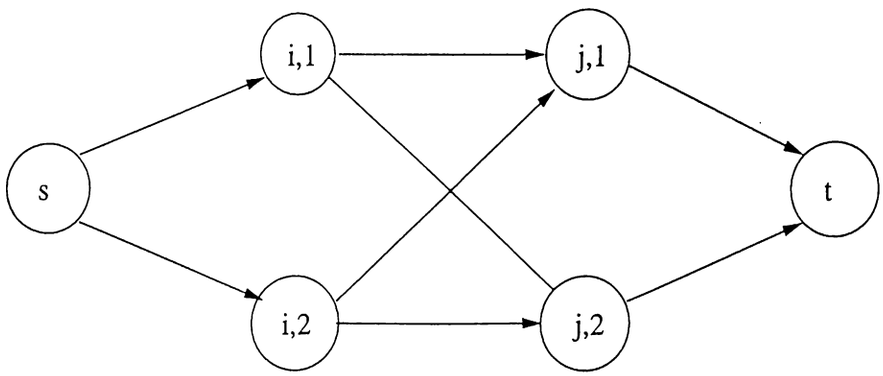

Figure 3.8: Graph in which we check if arc $(i1, j1)$ is weak

Scenario r^* is constructed by setting the following arc lengths: $l_{s,i1} = a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}, l_{s,i2} = 0, l_{j1,t} = b_{j1}^{r_{j1}} - b_{j2}^{r_{j1}}, l_{j2,t} = 0, l_{i1,j1} = \bar{l}_{i1,j1}, l_{i1,j2} = \underline{l}_{i1,j2}, l_{i2,j1} = \underline{l}_{i2,j1}, l_{i2,j2} = \underline{l}_{i2,j2}$.

Example 2

As an example, we check whether arc (22,31) is a weak arc or not in the graph in Figure 3.9.

We first would like to compute $a_{22}^{r_{22}} - a_{21}^{r_{22}}$. We consider the subgraph in Figure 3.10 generated by node s and layers 1 and 2.

We set the length of arcs $(s, 11)$ and (11,22) at upper bounds and the lengths of the remaining arcs at lower bounds and compute:

$$d_{22}^1 = \bar{l}_{s,11} + \bar{l}_{11,22} - \max\{\bar{l}_{s,11} + \underline{l}_{11,21}, \underline{l}_{s,12} + \underline{l}_{12,21}\} = 26 - 21 = 5$$

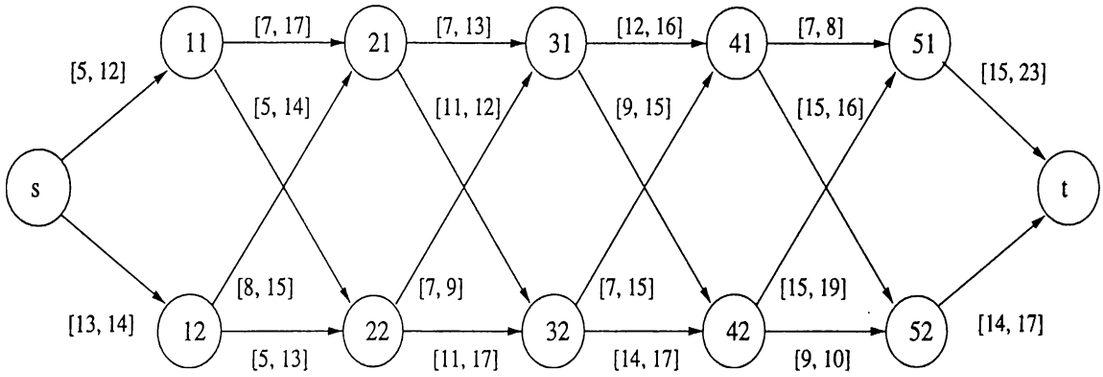

Figure 3.9: 5 layered graph with width 2 in which we check if arc (22,31) is weak

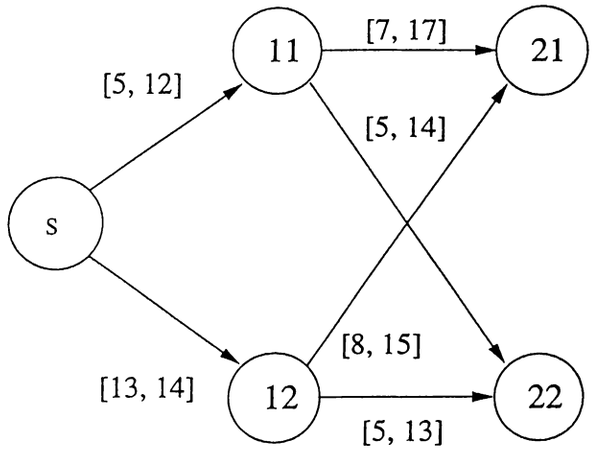

Figure 3.10: Subgraph generated by node s and layers 1 and 2

Next we set the length of arcs $(s,12)$ and (12,22) at upper bounds and the lengths of the remaining arcs at lower bounds and compute:

$$d_{22}^2 = \bar{l}_{s,12} + \bar{l}_{12,22} - \max\{\bar{l}_{s,12} + \underline{l}_{12,21}, \underline{l}_{s,11} + \underline{l}_{11,21}\} = 27 - 22 = 5$$

We pick the maximum difference:

$$d_{22} = \max\{d_{22}^1, d_{22}^2\} = 5$$

and set $\bar{l}_{s,22} = 5$ and $\underline{l}_{s,21} = 0$.

Next we would like to find $b_{31}^{r31} - a_{32}^{r31}$. We first consider the subgraph in Figure 3.11 generated by node t and layers 4 and 5.

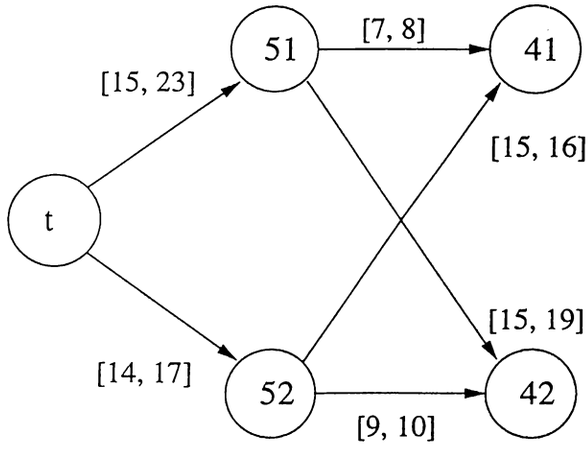

Figure 3.11: Mirror version of the subgraph generated by node t and layers 4 and 5

We do the same computations and find that $d_{41} = 3$ and $d_{42} = 12$. We set $\bar{l}_{t,41} = 3$ and $\underline{l}_{t,41} = 0$ and $\bar{l}_{t,42} = 12$ and $\underline{l}_{t,42} = 0$. We shrink the graph, add layer 3 and obtain the graph in Figure 3.12.

This time we find that $d_{31} = 5$. We set $\bar{l}_{31,t} = 5$ and $\underline{l}_{32,t} = 0$. We consider the small graph in Figure 3.13 and decide if there exists a longest path using arc (22,31) in this graph.

Since the longest path uses arc (22,31), we conclude that this arc is weak. In

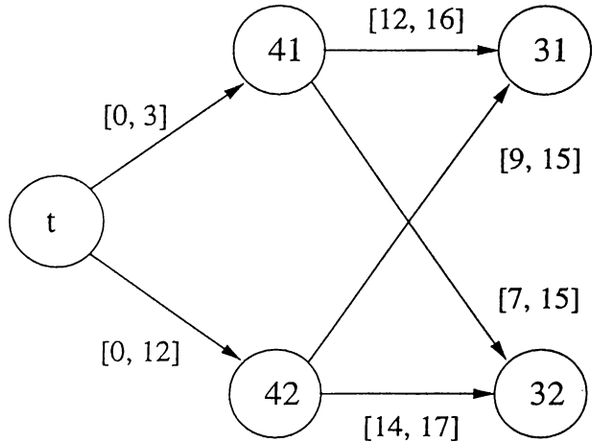

Figure 3.12: Mirror version of the subgraph shrunk between node t and layer 4

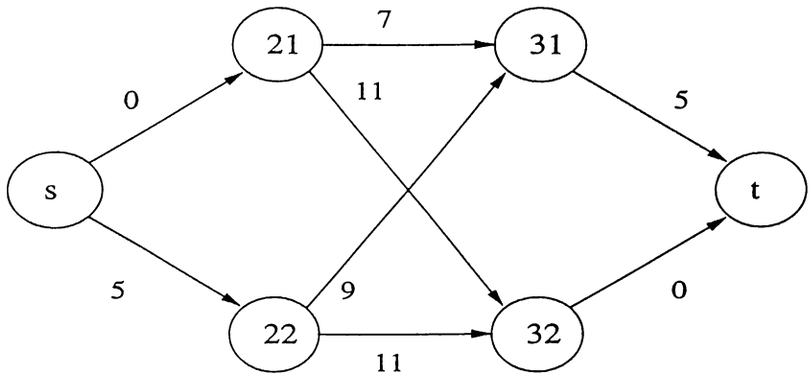

Figure 3.13: Final graph in which we check whether arc (22,31) is weak or not

fact, there are two weak paths using this arc. One is $s \rightarrow 11 \rightarrow 22 \rightarrow 31 \rightarrow 41 \rightarrow 52 \rightarrow t$ and the other one is $s \rightarrow 12 \rightarrow 22 \rightarrow 31 \rightarrow 41 \rightarrow 52 \rightarrow t$.

3.2 Strong Arcs

We shall now investigate the strong arc problem on layered graphs, and modify the procedures we have presented for the weak arc problem to solve the strong arc problem.

3.2.1 Arcs Incident at Nodes s and t

We first modify the algorithm we have given to check whether arcs $(s, 1k)$ for $k = 1, 2, .., w$ and (mk, t) for $k = 1, 2, .., w$ are weak or not to check whether the same arcs are strong. The procedure is changed to answer the question whether arc $(s, 11)$ is strong or not.

We will start with the scenario in which the lengths of all arcs that can possibly be on a path with $(s, 11)$ are set at their lower bounds and the lengths of the remaining arcs are set at their upper bounds. We try to obtain a scenario in which no longest path uses arc $(s, 11)$. So at each iteration, we set the lengths of arcs that can possibly be on a path that is longer than all paths using arc $(s, 11)$ to their upper bounds. Since this is equivalent to checking whether any of the remaining arcs is weak or not, favoring paths that use arc $(s, 11)$ in case of equalities, we do not give a separate procedure here, the procedure we have given for weak arc problem can be used.

3.2.2 Intermediate Arcs

Next, we give a trivial necessary condition for an intermediate arc to be strong in a layered graph, which can be checked using the above procedure.

Proposition 3.9 *If arc $(i1, j1)$ is strong in a layered graph, then it is strong in the subgraph generated by node s, layers 1 up to i and node $j1$, and it is strong in the subgraph generated by node $i1$, layers j up to m and node t.*

Now, we consider the strong arc problem on an m layered network with width 2. All results we obtained for the weak arc problem are modified for the strong arc problem.

Proposition 3.10 *Arc $(i1, j1)$ is a strong arc if and only if there exists a longest path using this arc for the scenario r^*, in which $a^{r^*}_{i1} - a^{r^*}_{i2} = \min_{r \in S}\{a^r_{i1} - a^r_{i2}\}$ and $b^{r^*}_{j1} - b^{r^*}_{j2} = \min_{r \in S}\{b^r_{j1} - b^r_{j2}\}$ and the length of arc $(i1, j1)$ is at its lower bound and the lengths of arcs $(i1, j2), (i2, j1)$ and $(i2, j2)$ are at upper bounds.*

Proof

If arc $(i1, j1)$ is a strong arc, then it is on a longest path for scenario r^*.

If arc $(i1, j1)$ is on a longest path for scenario r^* then the following inequalities hold:

$$a^{r^*}_{i1} + \underline{l}_{i1,j1} + b^{r^*}_{j1} - a^{r^*}_{i1} - \bar{l}_{i1,j2} - b^{r^*}_{j2} = \underline{l}_{i1,j1} + b^{r^*}_{j1} - \bar{l}_{i1,j2} - b^{r^*}_{j2} \geq 0 \quad (1)$$

$$a^{r^*}_{i1} + \underline{l}_{i1,j1} + b^{r^*}_{j1} - a^{r^*}_{i2} - \bar{l}_{i2,j1} - b^{r^*}_{j1} = \underline{l}_{i1,j1} + a^{r^*}_{i1} - \bar{l}_{i1,j2} - a^{r^*}_{i2} \geq 0 \quad (2)$$

$$a^{r^*}_{i1} + \underline{l}_{i1,j1} + b^{r^*}_{j1} - a^{r^*}_{i2} - \bar{l}_{i2,j2} - b^{r^*}_{j2} \geq 0 \quad (3)$$

For an arbitrary scenario r, we know:

$$a^{r^*}_{i1} - a^{r^*}_{i2} \leq a^r_{i1} - a^r_{i2} \quad (4)$$

$$b^{r^*}_{j1} - b^{r^*}_{j2} \leq b^r_{j1} - b^r_{j2} \quad (5)$$

$$l^r_{i1,j1} \geq \underline{l}_{i1,j1} \quad (6)$$

$$l^r_{i1,j2} \leq \bar{l}_{i1,j2} \quad (7)$$

$$l^r_{i2,j1} \leq \bar{l}_{i2,j1} \quad (8)$$

$$l^r_{i2,j2} \leq \bar{l}_{i2,j2} \quad (9)$$

Inequalities (1), (5), (6) and (7) imply that:

$$l^r_{i1,j1} + b^r_{j1} - l^r_{i1,j2} - b^r_{j2} \geq 0$$

inequalities (2), (4), (6) and (8) imply:

$$l^r_{i1,j1} + a^r_{i1} - l^r_{i2,j1} - a^r_{i2} \geq 0$$

and finally inequalities (3), (4), (5), (6) and (9) imply that:

$$a^r_{i1} + l^r_{i1,j1} + b^r_{j1} - a^r_{i2} - l^r_{i2,j2} - b^r_{j2} \geq 0$$

So, for any scenario r, there exists a longest path using arc $(i1, j1)$. So $(i1, j1)$ is a strong arc. □

To check whether arc $(i1, j1)$ is a strong arc or not, we need to find the scenario r^*. Similar to the work we have done for weak arcs, from this point on, we will only look for a scenario r_{i1}in which $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2} = \min_{r \in S}\{a^r_{i1} - a^r_{i2}\}$. To do this, it is enough to consider the subgraph generated by layers 1 up to i and node s.

Now, we will show that finding the scenario r_{i1} is equivalent to finding a path from node s to node $i1$ such that, the minimum difference $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2}$ is attained when the lengths of all arcs on this path are at lower bounds and the lengths of the remaining arcs are at upper bounds.

Proposition 3.11 *Let p be the longest path from node s to node $i1$ for scenario r_{i1}. Consider the scenario r', in which the lengths of all arcs on p are at lower bounds and the remaining arcs are at upper bounds. $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2} = a^{r'}_{i1} - a^{r'}_{i2}$.*

Proof

When we move from scenario r_{i1} to scenario r', it is easy to show that path p has the higher decrease in length, since no other path can have all arcs of p. This proves that $a^{r'}_{i1} - a^{r'}_{i2} \leq a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2}$. So $a^{r'}_{i1} - a^{r'}_{i2} = a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2}$. □

We developed a procedure similar to ComputeMaxDifference to find $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2}$.

Procedure ComputeMinDifference

for k=2 to i do

1. $d^1_{k1} = \underline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,1)} - \max\{\underline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,2)}, \overline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,2)}\}$
   $d^2_{k1} = \underline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,1)} - \max\{\underline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,2)}, \overline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,2)}\}$
   $d_{k1} = \min\{d^1_{k1}, d^2_{k1}\}$

2. $\underline{l}_{s,k1} = d_{k1}$
   $\overline{l}_{s,k2} = 0$

3. $d^1_{k2} = \underline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,2)} - \max\{\underline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,1)}, \overline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,1)}\}$
   $d^2_{k2} = \underline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,2)} - \max\{\underline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,1)}, \overline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,1)}\}$
   $d_{k2} = \min\{d^1_{k2}, d^2_{k2}\}$

4. $\underline{l}_{s,k2} = d_{k2}$
   $\overline{l}_{s,k1} = 0$

Proposition 3.12 *The above procedure computes* $a^{r_{i1}}_{i1} - a^{r_{i1}}_{i2}$.

Proof

The proof is done by induction on k.

Let $k = 2$ and $l = 1$. We have two paths from node s to node (21). Consider scenario r_{21}. In this scenario, the arcs on one of the paths from s to 21 are set at lower bounds and the remaining arcs are at upper bounds. Consider the below equalities:

$$d^1_{21} = \underline{l}_{s,11} + \underline{l}_{11,21} - \max\{\underline{l}_{s,11} + \overline{l}_{11,22}, \overline{l}_{s,12} + \overline{l}_{12,22}\}$$

$$d^2_{21} = \underline{l}_{s,12} + \underline{l}_{12,21} - \max\{\underline{l}_{s,12} + \overline{l}_{12,22}, \overline{l}_{s,11} + \overline{l}_{11,22}\}$$

$$d_{21} = \min\{d_{21}^1, d_{21}^2\}$$

The first inequality computes the difference between the length of path $s \rightarrow 11 \rightarrow 21$ and the length of the longest path from s to 22 when the arcs on path $s \rightarrow 11 \rightarrow 21$ are at lower bounds and the remaining arcs are at upper bounds. The second inequality does the same for the second path which is $s \rightarrow 12 \rightarrow 21$. We pick the one with the minimum difference in the third inequality. So $d_{21} = a_{21}^{r_{21}} - a_{22}^{r_{21}}$.

Assume that the algorithm computes differences $d_{k-1,1} = a_{k-1,1}^{r_{k-1,1}} - a_{k-1,2}^{r_{k-1,1}}$ and $d_{k-1,2} = a_{k-1,2}^{r_{k-1,2}} - a_{k-1,1}^{r_{k-1,2}}$ correctly. We will show that it can compute difference $a_{k1}^{r_{k1}} - a_{k2}^{r_{k1}}$. Let r_{k1} be the scenario for which $a_{k1}^{r_{k1}} - a_{k2}^{r_{k1}} = \min_{r \in S}\{a_{i1}^r - a_{i2}^r\}$. Then $a_{k1}^{r_{k1}} - a_{k2}^{r_{k1}} = \min\{a_{k-1,1}^{r_{k1}} + \underline{l}_{(k-1,1),(k,1)}, a_{k-1,2}^{r_{k1}} + \underline{l}_{(k-1,2),(k,1)}\} - \max\{a_{k-1,1}^{r_{k1}} + \overline{l}_{(k-1,1),(k,2)}, a_{k-1,2}^{r_{k1}} + \overline{l}_{(k-1,2),(k,2)}\}$. This is equivalent to $a_{k1}^{r_{k1}} - a_{k2}^{r_{k1}} = \min\{\underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,1),(k,2)}, a_{k-1,1}^{r_{k1}} + \underline{l}_{(k-1,1),(k,1)} - a_{k-1,2}^{r_{k1}} - \overline{l}_{(k-1,2),(k,2)}, a_{k-1,2}^{r_{k1}} + \underline{l}_{(k-1,2),(k,1)} - a_{k-1,1}^{r_{k1}} - \overline{l}_{(k-1,1),(k,2)}, \underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,2),(k,2)}\}$.

Consider d_{k1}^1 and d_{k1}^2 computed by the procedure.

$$d_{k1}^1 = \underline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,1)} - \max\{\underline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,2)}, \overline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,2)}\}$$

$$= \min\{\underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,1),(k,2)}, \underline{l}_{s,(k-1,1)} + \underline{l}_{(k-1,1),(k,1)} - \overline{l}_{s,(k-1,2)} - \overline{l}_{(k-1,2),(k,2)}\}$$

$$= \min\{\underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,1),(k,2)}, d_{k-1,1} + \underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,2),(k,2)}\}$$

$$= \min\{\underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,1),(k,2)}, a_{k-1,1}^{r_{k-1,1}} - a_{k-1,2}^{r_{k-1,1}} + \underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,2),(k,2)}\}$$

$$\leq \min\{\underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,1),(k,2)}, a_{k-1,1}^{r_{k1}} - a_{k-1,2}^{r_{k1}} + \underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,2),(k,2)}\}$$

$$d_{k1}^2 = \underline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,1)} - \max\{\underline{l}_{s,(k-1,2)} + \overline{l}_{(k-1,2),(k,2)}, \overline{l}_{s,(k-1,1)} + \overline{l}_{(k-1,1),(k,2)}\}$$

$$= \min\{\underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,2),(k,2)}, \underline{l}_{s,(k-1,2)} + \underline{l}_{(k-1,2),(k,1)} - \overline{l}_{s,(k-1,1)} - \overline{l}_{(k-1,1),(k,2)}\}$$

$$= \min\{\underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,2),(k,2)}, d_{k-1,2} + \underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,1),(k,2)}\}$$

$$= \min\{\underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,2),(k,2)}, a_{k-1,2}^{r_{k-1,2}} - a_{k-1,1}^{r_{k-1,2}} + \underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,1),(k,2)}\}$$

$$\leq \min\{\underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,2),(k,2)}, a_{k-1,2}^{r_{k1}} - a_{k-1,1}^{r_{k1}} + \underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,1),(k,2)}\}$$

$$d_{k1} = \min\{d_{k1}^1, d_{k1}^2\}$$

$$= \min\{\underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,1),(k,2)}, d_{k-1,1} + \underline{l}_{(k-1,1),(k,1)} - \overline{l}_{(k-1,2),(k,2)},$$

$$d_{k-1,2} + \underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,1),(k,2)}, \underline{l}_{(k-1,2),(k,1)} - \overline{l}_{(k-1,2),(k,2)}\}$$

So $d_{k1} \leq a_{k1}^{r_{k1}} - a_{k2}^{r_{k1}}$. Since $a_{k1}^{r_{k1}} - a_{k2}^{r_{k1}}$ is the minimum difference, $d_{k1} = a_{k1}^{r_{k1}} - a_{k2}^{r_{k1}}$. □

Proposition 3.13 *We can compute* $a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$ *and* $b_{j1}^{r_{j1}} - b_{j2}^{r_{j1}}$ *in* $O(m)$ *time using the above procedure.*

We can compute $a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}$ and $b_{j1}^{r_{j1}} - b_{j2}^{r_{j1}}$ using the above procedure. Scenario r^* is constructed by setting the following arc lengths: $l_{s,i1} = a_{i1}^{r_{i1}} - a_{i2}^{r_{i1}}, l_{s,i2} = 0, l_{j1,t} = b_{j1}^{r_{j1}} - b_{j2}^{r_{j1}}, l_{j2,t} = 0, l_{i1,j1} = \underline{l}_{i1,j1}, l_{i1,j2} = \overline{l}_{i1,j2}, l_{i2,j1} = \overline{l}_{i2,j1}, l_{i2,j2} = \overline{l}_{i2,j2}$.

Chapter 4

Minimum Spanning Tree Problem with Interval Data

In this chapter, we consider the minimum spanning tree problem with interval edge costs. We figure out if there exists a spanning tree which is minimum for all realizations. We also determine if there exists a realization for which a given spanning tree is minimum. We define two robustness measures. Then we investigate which edges are on minimum spanning trees for all realizations, and which edges are on a minimum spanning tree for no realization.

Let $G = (V, E)$ be an undirected graph with n nodes. Each edge (i, j) has cost $c_{ij} \in [\underline{c}_{ij}, \overline{c}_{ij}]$. No probability distribution is assumed for edge costs. c^s_{ij} denotes the cost of edge (i, j) in scenario s. We denote by c_{ij} an arbitrary cost for edge (i, j) in $[\underline{c}_{ij}, \overline{c}_{ij}]$.

Γ denotes the set of all spanning trees. We denote the cost of spanning tree T in scenario s by c^s_T. $\underline{c}_T$ denotes the cost of spanning tree T when the costs of all edges on this tree are at lower bounds and $\overline{c}_T$ denotes the cost of spanning tree T when the costs of all edges on this tree are at upper bounds.

4.1 Spanning Trees

First we analyze spanning trees. We define permanent, weak and robust trees and derive some results for them.

4.1.1 Permanent Trees

We first investigate whether there exists a spanning tree which has a minimum cost for all realizations of edge costs. We present a procedure to find such a spanning tree if there exists one, when all edges have non degenerate costs.

Definition 4.1 *A spanning tree is a* **permanent tree** *if it is a minimum spanning tree for all realizations of edge costs.*

The following theorem characterizes permanent trees.

Theorem 4.1 *A spanning tree is a permanent tree if and only if it is a minimum spanning tree when the costs of all edges on this tree are at their upper bounds and the costs of all other edges are at their lower bounds.*

Proof

If a spanning tree is a permanent tree then it is one of the minimum spanning trees for the stated realization by definition.

If a spanning tree T is minimum when the costs of all edges on T are at their upper bounds and the costs of all other edges are at their lower bounds, for an arbitrary spanning tree $T' \in \Gamma$, we have:

$$\sum_{(i,j)\in T\setminus T'} \overline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \overline{c}_{ij} \leq \sum_{(i,j)\in T'\setminus T} \underline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \overline{c}_{ij}$$

If we consider arbitrary values in the given intervals for the costs of edges $(i,j) \in T \cap T'$, we obtain:

$$\sum_{(i,j)\in T\setminus p'} \overline{c}_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij} \leq \sum_{(i,j)\in T'\setminus T} \underline{c}_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij}$$

Since $\sum_{(i,j)\in T\setminus T'} c_{ij} \leq \sum_{(i,j)\in T\setminus T'} \bar{c}_{ij}$ and $\sum_{(i,j)\in T'\setminus T} \underline{c}_{ij} \leq \sum_{(i,j)\in T'\setminus T} c_{ij}$, we have:

$$\sum_{(i,j)\in T\setminus T'} c_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij} \leq \sum_{(i,j)\in T'\setminus T} c_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij}$$

So $c_T \leq c_{T'}$ for all realizations. Since T' is an arbitrary spanning tree, this is true for all spanning trees in Γ. Thus, T is a permanent tree. □

Proposition 4.1 *Suppose there exists a permanent tree T. Consider the scenario in which all edge costs are at upper bounds. If there exists a tree T' such that $\bar{c}_T = \bar{c}_{T'}$, then $\underline{c}_{ij} = \bar{c}_{ij}$ for all $(i,j) \in T'\setminus T$. Moreover, a spanning tree T' with $\bar{c}_T = \bar{c}_{T'}$ is a permanent tree if and only if $\underline{c}_{ij} = \bar{c}_{ij}$ for all $(i,j) \in T\setminus T'$.*

Proof

Suppose there exists a permanent tree T. Assume there exists a spanning tree T' such that $\bar{c}_T = \bar{c}_{T'}$ and there exists an edge $(i,j) \in T'\setminus T$ whose cost is non degenerate, that is $\underline{c}_{ij} < \bar{c}_{ij}$. By setting the cost of edge (i,j) to its lower bound, we obtain a scenario in which cost of T' is less than the cost of T. This contradicts that T is a permanent tree.

A spanning tree T' with $\bar{c}_T = \bar{c}_{T'}$ is also a permanent tree if and only if for all realizations $c_T = c_{T'}$, which is equivalent to:

$$\sum_{(i,j)\in T\setminus T'} c_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij} = \sum_{(i,j)\in T'\setminus p} c_{ij} + \sum_{(i,j)\in p\cap T'} c_{ij}$$

for all realizations. This holds if and only if:

$$\sum_{(i,j)\in T\setminus T'} c_{ij} = \sum_{(i,j)\in T'\setminus T} c_{ij}$$

Since $\bar{c}_T = \bar{c}_{T'}$, we should have $\underline{c}_{ij} = \bar{c}_{ij}$ for all $(i,j) \in T'\setminus T$. Then, the right hand side of the inequality is a constant for all realizations. So the above equality holds if and only if $\underline{c}_{ij} = \bar{c}_{ij}$ for all $(i,j) \in T\setminus T'$. □

Corollary 4.1 *If all edge costs are non degenerate, that is $\underline{c}_{ij} < \bar{c}_{ij}$ for all $(i,j) \in E$ and there exists a permanent tree, it is the unique minimum spanning tree when the costs of all edges are at their upper bounds.*

Corollary 4.2 *If all edge costs are non degenerate and there exists a permanent tree, it is unique.*

Corollary 4.3 *If there exists a permanent tree such that all edges on this tree have non degenerate costs, then this is the unique permanent tree.*

We have a procedure to find the permanent tree if it exists when all edge costs are non degenerate.

Procedure FindPermanentTree

1. Set all edge costs at their upper bounds and find a minimum spanning tree T.
2. Check whether T is a permanent tree or not using Theorem 4.1. If T is a permanent tree, it is the unique permanent tree. If it is not permanent, then there exists no permanent tree.

4.1.2 Weak Trees

Next, we analyze spanning trees that have minimum costs for some realization of edge costs. We give a characterization for such spanning trees and investigate the case where there exists a permanent tree.

Definition 4.2 *A spanning tree is a* **weak tree** *if it is a minimum spanning tree for some realization of edge costs.*

The following theorem characterizes weak trees.

Theorem 4.2 *A spanning tree is a weak tree if and only if it is a minimum spanning tree when the costs of all edges on this tree are at their lower bounds and the costs of the other edges are at their upper bounds.*

Proof

If a spanning tree is minimum for the stated realization of edge costs, it is a weak tree by definition.

If a spanning tree, T is a weak tree then there exists a scenario s for which

$$\sum_{(i,j)\in T} c_{ij}^s \leq \sum_{(i,j)\in T'} c_{ij}^s \quad \forall T' \in \Gamma$$

Let T' be an arbitrary spanning tree in Γ. We have:

$$\sum_{(i,j)\in T\setminus T'} c_{ij}^s + \sum_{(i,j)\in T\cap T'} c_{ij}^s \leq \sum_{(i,j)\in T'\setminus T} c_{ij}^s + \sum_{(i,j)\in T\cap T'} c_{ij}^s$$

When we set the costs of all edges in $T \cap T'$ at their lower bounds, we still have:

$$\sum_{(i,j)\in T\setminus T'} c_{ij}^s + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij} \leq \sum_{(i,j)\in T'\setminus T} c_{ij}^s + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij}$$

When we consider the scenario in which the costs of all edges on T are at lower bounds and the costs of the remaining edges are at upper bounds, we obtain:

$$\sum_{(i,j)\in T\setminus T'} \underline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij} \leq \sum_{(i,j)\in T'\setminus T} \overline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij}$$

Since T' is picked arbitrarily, this is true for all spanning trees in Γ. Thus T is also a minimum spanning tree, when the costs of edges on T are at their lower bounds and the costs of all other edges are at their upper bounds. □

We consider the case where there exists at least one permanent tree in the graph.

Proposition 4.2 *If there exists a permanent tree T, then a spanning tree T' is a weak tree if and only if $\overline{c}_T = c_{T'}^s$ in scenario s in which the costs of all edges on T are at their upper bounds and the costs of all other edges are at their lower bounds.*

Proof

If $\overline{c}_T = c_{T'}^s$ for scenario s, then T' is a minimum spanning tree for this scenario.

Thus T' is a weak tree.

If T' is a weak tree, then T' is a minimum spanning tree when the costs of all edges on T' are at lower bounds and the costs of the remaining edges are at upper bounds. So we have:

$$\sum_{(i,j)\in T\setminus T'} \overline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij} = \sum_{(i,j)\in T'\setminus T} \underline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij}$$

When we set the costs of edges in $T \cap T'$ at upper bounds, we still have:

$$\sum_{(i,j)\in T\setminus T'} \overline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \overline{c}_{ij} = \sum_{(i,j)\in T'\setminus T} \underline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \overline{c}_{ij}$$

So $\overline{c}_T = c^s_{T'}$. □

Corollary 4.4 *If there exists a permanent tree T, and if $\overline{c}_T < c^s_{T'}$ $\forall T' \in \Gamma$, in scenario s in which the costs of all edges on T are at upper bounds and the costs of all other edges are at lower bounds, then no other tree can be weak.*

4.1.3 Robust Trees

We define two robustness measures, absolute robustness and relative robustness for the minimum spanning tree problem with interval edge costs, and characterize the worst case scenarios for a given spanning tree for both measures.

Let S denote the set of all possible scenarios.

Definition 4.3 *Given a spanning tree T, the* **absolute worst case scenario** *s^a_T is the scenario in which the cost of this spanning tree is the maximum. That is, $s^a_T = \arg\max_{s\in S} c^s_T$.*

Similar to what we had for the absolute robust longest path problem, the absolute worst case scenario for a given spanning tree corresponds to the scenario in which all edges on this spanning tree are at upper bounds.

Definition 4.4 *The spanning tree whose cost is the minimum for the absolute worst case scenario is called an* **absolute robust spanning tree**. *So absolute robust spanning tree* $T^a = \arg\min_{T\in\Gamma} \max_{s\in S} c_T^s = \arg\min_{T\in\Gamma} c_T^{s_T^a}$.

Kouvelis and Yu [4] have also studied the absolute robust spanning tree problem, where the scenario set is finite, and they have shown that the absolute robust spanning tree problem is NP-complete for bounded scenario set and strongly NP-hard when the scenario set is unbounded. In case of interval edge costs, the absolute worst case scenarios for all spanning trees correspond to the scenario in which all edge costs are at upper bounds. Thus, the absolute robust spanning tree problem with interval edge costs can be solved in polynomial time by finding a minimum spanning tree when all edge costs are at upper bounds.

Definition 4.5 *Given a spanning tree* T, *the* **relative worst case scenario** s_T^r *is the scenario in which the difference between the cost of the spanning tree* T *and the cost of the minimum spanning tree is the maximum. That is,* $s_T^r = \arg\max_{s\in S} c_T^s - c_{T^*(s)}^s$, *where* $T^*(s)$ *is the minimum spanning tree for scenario* s. *We call the difference* $d_T = c_T^{s_T^r} - c_{T^*(s)}^{s_T^r}$ *the* **robust deviation** *for spanning tree* T.

Definition 4.6 *The spanning tree whose robust deviation is the minimum is called a* **relative robust spanning tree**. *In other words, a relative robust spanning tree* $T^r = \arg\min_{T\in\Gamma} d_T = \arg\min_{T\in\Gamma} \max_{s\in S} c_T^s - c_{T^*(s)}^s$.

We know that an absolute robust spanning tree is a weak tree. The below proposition states that a relative robust spanning tree is also a weak tree.

Proposition 4.3 *A relative robust spanning tree is a weak tree.*

Proof

Let T be a spanning tree which is not weak. Consider the spanning tree T' which is a minimum spanning tree for the scenario in which the costs of all

edges on T are at lower bounds and the costs of the remaining edges are at upper bounds. Then $c_T > c_{T'}$ for all scenarios. Consider the scenario s' which is the relative worst case scenario for spanning tree T'. We have:

$$d_{T'} = c^{s'}_{T'} - c^{s'}_{T^*(s')} < c^{s'}_{T} - c^{s'}_{T^*(s')} \leq \max_{s \in S} c^{s}_{T} - c^{s}_{T^*(s)} = d_T$$

So T cannot be a robust spanning tree. □

The following proposition states a relative worst case scenario for a given spanning tree.

Proposition 4.4 *The relative worst case scenario for spanning tree T is the scenario in which the costs of all edges on T are at upper bounds and the costs of all other edges are at lower bounds.*

Proof

Let d_T be the robust deviation for spanning tree T. Then:

$$d_T = c^{s^r_T}_{T} - c^{s^r_T}_{T^*(s^r_T)} = \sum_{(i,j) \in T \setminus T^*(s^r_T)} c^{s^r_T}_{ij} - \sum_{(i,j) \in T^*(s^r_T) \setminus T} c^{s^r_T}_{ij}$$

Let s be the scenario in which the costs of all edges on T are at their upper bounds and the costs of the remaining edges are at their lower bounds. We have:

$$d_T \leq \sum_{(i,j) \in T \setminus T^*(s^r_T)} c^{s}_{ij} - \sum_{(i,j) \in T^*(s^r_T) \setminus T} c^{s}_{ij} = c^{s}_{T} - c^{s}_{T^*(s^r_T)}$$

Since $c^{s}_{T^*(s)} \leq c^{s}_{T^*(s^r_T)}$, we have:

$$d_T \leq c^{s}_{T} - c^{s}_{T^*(s^r_T)} \leq c^{s}_{T} - c^{s}_{T^*(s)}$$

Since $d_T = \max_{s' \in S} c^{s'}_{T} - c^{s'}_{T^*(s')}$, we have $d_T = c^{s}_{T} - c^{s}_{T^*(s)}$. So s is a relative worst case scenario for spanning tree T. □

Kouvelis and Yu [4] have also proved that relative robust spanning tree problem is NP-complete for bounded number of scenarios and is strongly NP-hard with unbounded number of scenarios. They conjecture that the problem with interval edge costs is also NP-complete.

4.2 Edges

We next analyze the problems of deciding whether a given edge is always on a minimum spanning tree, or whether a given edge is never on a minimum spanning tree and give characterizations to solve both problems in polynomial time.

4.2.1 Weak Edges

Definition 4.7 *An edge is a* **weak edge** *if it lies on some weak tree.*

The following theorem gives a characterization of weak edges.

Theorem 4.3 *Edge (i,j) is a weak edge if and only if there exists a minimum spanning tree using edge (i,j) when the cost of edge (i,j) is at its lower bound and the costs of the remaining edges are at upper bounds.*

Proof
If there exists a minimum spanning tree that uses edge (i,j) for the above scenario, then edge (i,j) is weak by definition.

Consider Kruskal's algorithm [1] which sorts all edges in nondecreasing order of their costs, and defines a set LIST, which is the set of edges chosen as part of the minimum spanning tree. Initially, the set LIST is empty. The algorithm examines the edges in the sorted order one by one and checks whether adding the edge currently examined to the set LIST creates a cycle with the edges already in LIST. If it doesn't, it is added to LIST, otherwise this edge is discarded. In case of ties, the algorithm favors edge (i,j). The algorithm stops when there are $n-1$ edges in LIST. Let E_{ij}^{s} be the set of edges preceding edge (i,j) in the sorted set of edges for scenario s in which the cost of edge (i,j) is at its lower bound and the costs of the other edges are at upper bounds. Let s' be another scenario and $E_{ij}^{s'}$ be the set of edges preceding edge (i,j) in the

sorted set for scenario s'. It is easy to show that $E_{ij}^{s} \subseteq E_{ij}^{s'}$ for any scenario s'. If (i,j) is not on a minimum spanning tree for scenario s, then either the size of LIST reaches $n-1$ before edge (i,j) is examined or adding edge (i,j) to the LIST results in a cycle. We consider the second case only, since in both cases, adding (i,j) into LIST results in a cycle. Let C denote this cycle. If there were a scenario in which it would be possible to add edge (i,j) to LIST, then there would be at least one edge $(k,l) \in C$, which wouldn't be added to LIST. This could be possible only if adding edge (k,l) to the LIST would result in another cycle C'. But this implies that when we add (i,j) to the list we get another cycle $C'' = (C \cup C') \backslash (k,l)$. So it is not possible to find a scenario in which there exists a minimum spanning tree that uses edge (i,j). Thus edge (i,j) is not weak. □

4.2.2 Strong Edges

Definition 4.8 *An edge is a* **strong edge**, *if it lies on a minimum spanning tree for all realizations of edge costs.*

Below, we give a characterization for strong edges.

Theorem 4.4 *Edge (i,j) is a strong edge if and only if there exists a minimum spanning tree using edge (i,j) when the cost of edge (i,j) is at its upper bound and the costs of the remaining edges are at lower bounds.*

Proof

If edge (i,j) is a strong edge, then it lies on a minimum spanning tree for the above scenario by definition.

Assume that edge (i,j) is not strong. Then there exists at least one scenario for which edge (i,j) does not lie on any minimum spanning tree. Thus in considering that scenario, adding (i,j) to LIST creates a cycle, C. Let s denote this scenario and E_{ij}^{s} denote the set of edges that precede edge (i,j)

in the sorted set for this scenario. If we set the cost of (i,j) to its upper bound and the costs of the remaining edges at their lower bounds and call this scenario s' and denote the edges that precede edge (i,j) in the sorted set for this scenario by $E_{ij}^{s'}$, we get $E_{ij}^{s} \subseteq E_{ij}^{s'}$. For scenario s', for (i,j) to be added to LIST, there should exist an edge $(k,l) \in C$ that could not be added to LIST, this implies that adding edge (k,l) to LIST creates another cycle C'. Using the above argument, we see that adding edge (i,j) to LIST will result in a cycle. So edge (i,j) cannot lie on a minimum spanning tree for scenario s'. □

The next proposition considers the case in which there exists at least one permanent tree.

Proposition 4.5 *If there exists a permanent tree then an edge is strong if and only if it lies on a permanent tree.*

Proof

If an edge lies on a permanent tree, then it is strong.

Suppose there exists a permanent tree T in the graph. Suppose edge (i,j) is not on a permanent tree and assume it is a strong edge. Consider the scenario in which the costs of all edges on spanning tree T are at lower bounds and the costs of the remaining edges are at upper bounds. Since (i,j) is a strong edge, there exists a spanning tree T' which uses edge (i,j) and which is a minimum spanning tree for this scenario. Then we have:

$$\sum_{(m,n)\in T\setminus T'} \underline{c}_{mn} + \sum_{(m,n)\in T\cap T'} \underline{c}_{mn} = \sum_{(m,n)\in T'\setminus T} \bar{c}_{mn} + \sum_{(m,n)\in T\cap T'} \underline{c}_{mn}$$

Since T is a permanent tree, we should have $\underline{c}_{mn} = \bar{c}_{mn}$ for all edges $(m,n) \in T'\setminus T$ and for all edges $(m,n) \in T\setminus T'$. Then for all realizations, we have $c_T = c_{T'}$. This implies that T' is also a permanent tree. But this contradicts that edge (i,j) does not lie on any permanent tree. So (i,j) is not a strong edge. □

We next give a characterization of strong edges using unionwise permanent

sets. We conclude that all spanning trees in a minimum unionwise permanent set share all strong edges, and there exists a relative robust spanning tree in the minimum unionwise permanent set. We assume that all edges have non degenerate costs.

Definition 4.9 *A set of spanning trees is a* **unionwise permanent set** *if for each realization there exists a minimum spanning tree in this set.*

Definition 4.10 *A unionwise permanent set is a* **minimum unionwise permanent set** *if it is no longer a unionwise permanent set when a spanning tree is removed.*

Lemma 4.1 *There exists no scenario for which a given weak tree is the unique minimum spanning tree if and only if this spanning tree is not the unique minimum spanning tree when the costs of all edges on it are at their lower bounds and the costs of the remaining edges are at their upper bounds.*

Proof

Since the spanning tree is not the unique minimum spanning tree for any scenario, it won't be the unique minimum spanning tree for the stated scenario.

If a spanning tree T, is not the unique minimum spanning tree when the costs of all edges on T are at lower bounds and the costs of the remaining edges are at upper bounds, there exists another spanning tree T' such that

$$\sum_{(i,j)\in T\setminus T'} \underline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij} = \sum_{(i,j)\in T'\setminus T} \overline{c}_{ij} + \sum_{(i,j)\in T\cap T'} \underline{c}_{ij}$$

For arbitrary costs of edges in $T \cap T'$, we have:

$$\sum_{(i,j)\in T\setminus T'} \underline{c}_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij} = \sum_{(i,j)\in T'\setminus T} \overline{c}_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij}$$

Then:

$$\sum_{(i,j)\in T\setminus T'} c_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij} \geq \sum_{(i,j)\in T'\setminus T} c_{ij} + \sum_{(i,j)\in T\cap T'} c_{ij}$$

that is $c_T \geq c_{T'}$ for all scenarios, so T can never be the unique minimum spanning tree. □

Lemma 4.2 *In any graph, there exists at least one spanning tree which is the unique minimum spanning tree for some scenario.*

Proof

Assume no such spanning tree exists. Pick an arbitrary spanning tree T_1. From the proof of the previous proposition, it follows that there exists a spanning tree $T_2 \neq T_1$ such that $c_{T_1} \geq c_{T_2}$ for all scenarios. Since T_2 is not the unique minimum spanning tree for any scenario, there exists another spanning tree $T_3 \neq T_2, T_1$ such that $c_{T_2} \geq c_{T_3}$ for all scenarios. Repeating this argument for all spanning trees in the graph, since no spanning tree can be repeated (it is not possible for two spanning trees to have the same costs for all scenarios, since by perturbing the cost of an edge which is on one of the spanning trees, but not both, we can have different costs), and the number of spanning trees is finite, we end up with a sequence of spanning trees:

$$c_{T_1} \geq c_{T_2} \geq c_{T_3} \ldots \geq c_{T_k}$$

where k is the number of spanning trees in the graph. Since the above inequalities hold for all realizations, T_k is a permanent tree. But a permanent tree is the unique minimum spanning tree when the costs of all edges are at upper bounds. This contradicts that no spanning tree is the unique minimum spanning tree for any scenario. □

Lemma 4.3 *If a spanning tree T is never the unique minimum spanning tree, there exists a spanning tree T' such that $c_T \geq c_{T'}$ for all scenarios and T' is the unique minimum spanning tree for some scenario.*

Proof

If T is not the unique minimum spanning tree for any scenario, there exists another spanning tree T' such that $c_T \geq c_{T'}$ for all scenarios. If $T' \neq T$ is the unique minimum spanning tree for some scenario, we are done. Assume not, then there exists another spanning tree $T'' \neq T', T$ such that $c_{T'} \geq c_{T''}$ for all scenarios. If we repeat this argument, we will end up with a sequence of spanning trees that are not the unique minimum spanning trees for any

scenario. Again, since no spanning tree can be repeated , and the number of spanning trees is finite, we will either find a unique minimum spanning tree and stop or we will enumerate all spanning trees. Since there exists at least one spanning tree which is the unique minimum spanning tree for some scenario, we will stop with a unique minimum spanning tree in any case. □

Now we give a characterization for minimum unionwise permanent set and show that it is unique.

Theorem 4.5 *Let Γ' be the set of spanning trees each of which is the unique minimum spanning tree when the costs of all edges on this spanning tree are at their lower bounds and the costs of the remaining edges are at their upper bounds. Then, Γ' is the minimum unionwise permanent set.*

Proof

Assume Γ' is not a unionwise permanent set. Then there exists a scenario s, for which no spanning tree in Γ' is minimum. Then there exists a spanning tree $T \in \Gamma \backslash \Gamma'$ such that $c_T^s < c_{T_i}^s \quad \forall T_i \in \Gamma'$. But since $T \in \Gamma \backslash \Gamma'$, there exists a spanning tree $T_j \in \Gamma'$ such that $c_T \geq c_{T_j}$ for all scenarios. This contradicts that cost of T is smaller than the costs of all spanning trees in Γ'. So Γ' is a unionwise permanent set.

Γ' is minimum since any spanning tree in Γ' is the unique minimum spanning tree for some scenario. □

Corollary 4.5 *Γ^* is a unionwise permanent set if and only if $\Gamma' \subseteq \Gamma^*$.*

Corollary 4.6 *The minimum unionwise permanent set is unique.*

Proposition 4.6 *An edge is strong if and only if all spanning trees in the minimum unionwise permanent set share that edge.*

Proof

If an edge (i, j) is strong it is on a minimum spanning tree for all scenarios.

Since each spanning tree in the minimum unionwise permanent set is the unique minimum spanning tree for some scenario, (i,j) should lie on all of them.

If an edge (i,j) is shared by all spanning trees in the minimum unionwise permanent set, then it is on a minimum spanning tree for all scenarios, thus it is strong. □

Proposition 4.7 *There exists a relative robust spanning tree in the minimum unionwise permanent set* Γ'.

Proof

Assume none exists. Then there is a relative robust spanning tree $T \in \Gamma \backslash \Gamma'$. Since T is not the unique minimum spanning tree for any scenario there exists a spanning tree $T' \in \Gamma'$ such that $c_T \geq c_{T'}$ for all scenarios. Consider the relative worst case scenario for spanning tree T'. we have:

$$c_{T'}^{s^r_{T'}} - c_{T^*(s^r_{T'})}^{s^r_{T'}} \leq c_T^{s^r_{T'}} - c_{T^*(s^r_{T'})}^{s^r_{T'}}$$

The left hand side of the inequality is equal to the worst case deviation for spanning tree T', and the right hand side of the inequality is less than or equal to the worst case deviation for spanning tree T. That is, $d_{T'} \leq d_T$. Since spanning tree T is a relative robust spanning tree, this inequality holds as an equality and T' is also a relative robust spanning tree. □

Corollary 4.7 *There exists a relative robust spanning tree which uses all strong edges.*

Chapter 5

Single Machine Scheduling with Interval Data

In this chapter, we consider the scheduling problem on a single machine with the total flow time criterion, where processing times are given as intervals. We investigate if there is a schedule which is optimal for all realizations. Then, we analyze which schedules can be optimal for some realizations. We also consider the positions of jobs in schedules. We find if a job is assigned the same position in optimal schedules for all realizations or if a job is assigned a given position in an optimal schedule for some realization. We use this information for preprocessing relative robust scheduling problem, which has been studied by Daniels and Kouvelis [3].

Let J denote the set of jobs to be scheduled. There are n jobs to be scheduled. Processing time of job i is $p_i \in [\underline{p}_i, \overline{p}_i]$. We do not assume any probability distribution for the processing times. We assume that all jobs are ready at time 0. Given a scenario of processing times, a schedule solves the problem if and only if it is a shortest processing time (SPT) schedule.

5.1 Schedules

We characterize permanent and weak schedules in this section. The relative robust scheduling problem has been studied by Daniels and Kouvelis [3].

5.1.1 Permanent Schedules

We first consider schedules that are optimal for all realizations of processing times.

Definition 5.1 *A schedule is a* **permanent schedule** *if it is an SPT schedule for all realizations.*

Proposition 5.1 *A schedule $\sigma = (\sigma_1, \sigma_2, .., \sigma_n)$ is a permanent schedule if and only if $\overline{p}_{\sigma_i} \leq \underline{p}_{\sigma_{i+1}}$ for $i = 1, 2.., n-1$, where σ_i is the index of the job which is assigned the ith position in the schedule σ.*

Proof
If $\overline{p}_{\sigma_i} \leq \underline{p}_{\sigma_{i+1}}$ for $i = 1, 2, .., n-1$, then schedule σ will be an SPT schedule for all realizations.

If there exists i such that $\overline{p}_{\sigma_i} > \underline{p}_{\sigma_{i+1}}$, by setting the processing time of job σ_i at its upper bound and the processing times of the remaining jobs at lower bounds, we obtain a scenario in which job σ_{i+1} precedes job σ_i in all SPT schedules. So σ cannot be a permanent schedule. □.

So we can check whether there exists a permanent schedule or not using the above proposition.

Corollary 5.1 *If all jobs have non degenerate processing times and there exists a permanent schedule, it is unique.*

5.1.2 Weak Schedules

We next investigate schedules that are optimal for some realizations of processing times.

Definition 5.2 *A schedule is a* **weak schedule** *if it is an SPT schedule for some realization.*

Below is a trivial proposition that characterizes weak schedules.

Proposition 5.2 *A schedule $\sigma = (\sigma_1, \sigma_2, .., \sigma_n)$ is a weak schedule if and only if for each job σ_i there exists $p_{\sigma_i} \in [\underline{p}_{\sigma_i}, \overline{p}_{\sigma_i}]$ such that $p_{\sigma_i} \leq p_{\sigma_{i+1}}$ for all $i = 1, 2, .., n-1$.*

Corollary 5.2 *If there exists $p \in [\underline{p}_i, \overline{p}_i]$ for all $i = 1, 2, .., n$, then all schedules are weak.*

5.1.3 Robust Schedules

We first give a definition for the relative robust schedule and then list some of the results by Daniels and Kouvelis [3].

Let S denote the set of all scenarios.

Definition 5.3 *The* **relative worst case scenario** *s_σ for a given schedule σ, is the scenario for which the difference between the total flow time for that schedule and the total flow time for an SPT schedule for that scenario is the maximum. That is, $s_\sigma = \arg\max_{s \in S} C_\sigma^s - C_{\sigma^*(s)}^s$ where $\sigma^*(s)$ denotes an SPT schedule for scenario s and C_σ^s denotes that total flow time of schedule σ in scenario s. This maximum difference is called the* **robust deviation** *for schedule σ.*

Definition 5.4 *A* relative robust schedule *is the schedule whose robust deviation is minimum.*

Daniels and Kouvelis [3] prove that a relative robust schedule is a weak schedule. They also show that in the relative worst case scenario for a schedule, all processing times are at bounds. Then they prove that the problem is NP-hard for the case of discrete scenarios, even when the number of scenarios is 2. But they do not provide any information for the case of interval processing times.

5.2 Assignments

We investigate if a given job is assigned the same position in an SPT schedule for all realizations. Then we decide whether a given job is assigned a certain position in an SPT schedule for some realization. We use this information to preprocess the schedules for the relative robust scheduling problem.

5.2.1 Weak Assignments

We investigate whether there exists a scenario in which job i assigned position j in an SPT schedule or not.

Definition 5.5 *Assignment of job i to position j is a* **weak assignment** *if there exists a weak schedule in which job i is assigned position j.*

Proposition 5.3 *Assignment of job i to position j is a weak assignment if and only if there exists $p_i \in [\underline{p}_i, \overline{p}_i]$ such that there are $j-1$ jobs with each job k having $\underline{p}_k \leq p_i$ and $n-j$ jobs with each job h having $p_i \leq \overline{p}_h$.*

Proof

If there exists such $p_i \in [\underline{p}_i, \overline{p}_i]$, then we partition the set of jobs except job i

into two sets, I^- and I^+ such that $j-1$ jobs with each job k having $\underline{p}_k \leq p_i$ are in I^- and $n-j$ jobs with each job h having $p_i \leq \overline{p}_h$ are in I^+. Then there exists an SPT schedule in which job i is assigned position j when processing times of all jobs in I^- are at lower bounds, processing times of jobs in I^+ are at upper bounds and processing time of job i is at p_i.

If there exists a weak schedule in which job i is assigned position j, then there exists $p_i \in [\underline{p}_i, \overline{p}_i]$ such that there are $j-1$ jobs with each job k having $p_k \leq p_i$ and $n-j$ jobs with each job h having $p_i \leq p_h$. For this p_i, we have $\underline{p}_k \leq p_i$ for all $j-1$ jobs preceding job i and $p_i \leq \overline{p}_h$ for all $n-j$ jobs succeeding job i. □

Since a relative robust schedule is a weak schedule, each assignment in this schedule is weak. So if an assignment is not weak, we can force the relative robust schedule not to do this assignment.

5.2.2 Strong Assignments

We next find out if for all realizations there exists an SPT schedule in which job i is assigned position j.

Definition 5.6 *Assignment of job i to position j is a* **strong assignment** *if for all realizations, there exists an SPT schedule in which job i is assigned position j.*

Below, we characterize strong assignments and then show that there exists a relative robust schedule in which all strong assignments are done.

Proposition 5.4 *Assume that job i has a non degenerate processing time, that is $\underline{p}_i < \overline{p}_i$. Then, assignment of job i to position j is a strong assignment if and only if there are $j-1$ jobs with each job k having $\overline{p}_k \leq \underline{p}_i$ and $n-j$ jobs with each job h having $\underline{p}_h \geq \overline{p}_i$.*

Proof

If there exists such a partition of jobs except job i into two sets, I^- and I^+ such that job $k \in I^-$ if and only if $\overline{p}_k \leq \underline{p}_i$ and job $h \in I^+$ if and only if $\underline{p}_k \geq \overline{p}_i$, then there exist SPT schedules in which all jobs in I^- will precede job i and all jobs in I^+ will succeed job i for all realizations.

If there does not exist such a partition, then either there are less than $j-1$ jobs with $\overline{p}_k \leq \underline{p}_i$, or there are at least $j-1$ jobs such that $\overline{p}_k \leq \underline{p}_i$, but for any choice of $j-1$ jobs from these jobs, there exists a job l such that $\underline{p}_l < \overline{p}_i$ among the remaining jobs. In the first case by setting the processing time of job i to lower bound and the processing times of remaining jobs at upper bounds, we obtain a scenario in which job i is assigned an earlier position than position j in all SPT schedules. So the assignment is not strong.

In the second case, by setting the processing times of $j-1$ jobs with $\overline{p}_k \leq \underline{p}_i$ and the processing time of job i at upper bounds and the processing times of the remaining jobs at lower bounds, we end up with a scenario in which in all SPT schedules job i is assigned a later position compared to j, since for all $j-1$ jobs with $\overline{p}_k \leq \underline{p}_i$, we have $\overline{p}_i > \underline{p}_i \geq \overline{p}_k$ and there exists a job l among the remaining jobs such that $\underline{p}_l < \overline{p}_i$. Thus the assignment is not strong. □

If assignment of job i to position j is strong then the relative robust scheduling problem can be solved for jobs in I^- and jobs in I^+ separately.

Theorem 5.1 *If assignment of job i to position j is strong, there exists a relative robust schedule in which job i is assigned position j, all jobs in I^- precede job i and all jobs in I^+ succeed job i.*

Proof

Let σ be a relative robust schedule. Suppose there exists a job l such that $\overline{p}_l \leq \underline{p}_i$ and job i precedes job l. Since σ is a weak schedule, we have $\overline{p}_l = \underline{p}_i$. Let σ' be the schedule in which positions of jobs i and l are reversed. Let π_i denote the position of job i in schedule σ and p_i^s denote the processing time of

job i in scenario s. Consider the worst case scenario s' for schedule σ'. Then:

$$C_{\sigma}^{s'} = C_{\sigma'}^{s'} + (\pi_l - \pi_i)(p_i^{s'} - p_l^{s'})$$

Since $\overline{p}_l = \underline{p}_i$, $p_i^{s'} - p_l^{s'} \geq 0$ and since job i precedes job l in schedule σ, $\pi_l - \pi_i \geq 0$. So we have $C_{\sigma}^{s'} \geq C_{\sigma'}^{s'}$.

$$C_{\sigma}^{s'} - C_{\sigma^*(s')}^{s'} \geq C_{\sigma'}^{s'} - C_{\sigma^*(s')}^{s'}$$

Since the right hand side of the inequality is the robust deviation for schedule σ' and the robust deviation for schedule σ is greater than or equal to the left hand side of the inequality, schedule σ' is also a relative robust schedule.

So there exists a robust schedule in which all jobs in I^- precede job i, since for all jobs $k \in I^-$ we have $\overline{p}_k \leq \underline{p}_i$, and all jobs in I^+ succeed job i, since for all jobs $k \in I^+$ we have $\underline{p}_k \geq \overline{p}_i$. In this schedule job i will be assigned position j.
□

Chapter 6

Conclusion

In this thesis, we investigated three problems, longest path problem on directed acyclic graphs, minimum spanning tree problem and single machine scheduling problem with total flow time criterion where the input data for all problems are given as intervals. We approached the problems in two ways. We first defined weak and permanent solutions and gave characterizations for these solutions. Then we defined two robustness measures and derived some results for them. In all problems, we saw that determining permanent and weak solutions help us in finding robust solutions, so that two approaches are closely related with each other.

We considered the longest path problem on directed acyclic graphs in connection with project management. We showed that finding weak and permanent paths help us in determining critical activities. We also analyzed arcs to distinguish activities that are critical for all realizations and activities that are critical for some realizations. We saw that this analysis is not only meaningful for project management, but also helpful to preprocess a given graph for finding a relative robust path.

We extended these results to minimum spanning tree problem with interval edge costs. We obtained stronger results in analyzing weak and strong edges

compared to what we had for weak and strong arcs. We again concluded that knowing weak and strong edges help us in finding a relative robust spanning tree.

Finally, we investigated the single machine scheduling problem with total flow time criterion where the processing times are interval numbers. We defined and characterized weak and permanent schedules. We also considered the positions of jobs in optimal schedules, and showed that using such an analysis, we can eliminate some of the schedules in searching for a relative robust schedule.

In all problems we have worked on, we saw that it is enough to consider extreme scenarios, that is the scenarios in which all data are at bounds, to find weak and permanent solutions. Moreover, the worst case scenarios for both measures of robustness correspond to extreme scenarios. Another similar result we obtained in all problems is that, if all intervals are non degenerate and there exists a permanent solution, it is unique. We also observed that, robust solutions are weak solutions for all problems. Also, we proved that there exists a relative robust path which uses all strong arcs, there exists a relative robust minimum spanning tree which uses all strong edges and there exists a relative robust schedule which obeys all strong assignments.

Bibliography

[1] Ahuja, R.K., T.L. Magnanti and J.B. Orlin, *Network Flows*, Prentice Hall, New Jersey, 1993.

[2] Chanas, S. and J. Kamburowski, "The Use of Fuzzy Variables in PERT", *Fuzzy Sets and Systems*, **5**, 11-19, 1981.

[3] Daniels, R.L. and P. Kouvelis, "Robust Scheduling to Hedge Against Processing Time Uncertainty in Single-stage Production", *Management Science*, **41** (2), 363-376, 1995.

[4] Kouvelis, P. and G. Yu, *Robust Discrete Optimization and Its Applications*, Kluwer Academic Publishers, The Netherlands, 1997.

[5] Kozina, G.L. and V.A. Perepelista, "Interval Spanning Trees Problem: Solvability and Computational Complexity", *Interval Computations*, (1), 42-50, 1994.

[6] Lootsma, F.A, "Stochastic and Fuzzy PERT", *European Journal of Operational Research*, **43**, 174-183, 1989.

[7] McCahon, C.S., "Using PERT as an Approximation of Fuzzy Project Network Analysis", *IEEE Transactions on Engineering Management*, **40** (2), 146-153, 1993.

[8] Nasution, S.H., "Fuzzy Critical Path Method", *IEEE Transactions on Systems, Man and Cybernetics*, **24** (1), 49-57, 1994.

[9] Rommelfanger, H.J., "Network Analysis and Information Flow in Fuzzy Environment", *Fuzzy Sets and Systems*, **67**, 119-128, 1994.

[10] Tansel, B.Ç., Personal Communication, 1998.

[11] Tansel, B.Ç. and G.F. Scheuenstuhl, "Facility Location on Tree Networks with Imprecise Data", Research Report:IEOR-8819. Department of Industrial Engineering, Bilkent University, 1988.