Algorithms for some Graph-Theoretical Optimization Problems

Linda Moonen

Committee

Prof. dr. Frits C.R. Spieksma (Advisor)	*Katholieke Universiteit Leuven*
Prof. dr. Yves Crama	*Université de Liège*
Prof. dr. Erik Demeulemeester	*Katholieke Universiteit Leuven*
Prof. dr. Thomas Erlebach	*University of Leicester*
Prof. dr. Willy Gochet	*Katholieke Universiteit Leuven*
Prof. dr. ir. drs. Koos Vrieze	*Universiteit Maastricht*

Acknowledgements

No one who achieves success does so
without acknowledging the help of others.
The wise and confident
acknowledge this help with gratitude.
Alfred North Whitehead (1861-1947)

When I was working on my Master's thesis at Maastricht University a little over four years ago, the opportunity to start a PhD at the university of Leuven presented itself. At first I was very hesitant about taking this chance for two reasons. First of all, being at the end of my studies, I wasn't very enthusiastic about the thought of spending another four years studying, and second, it would mean I have to leave Maastricht and move to Belgium. Now, four years later, I am very glad that I eventually decided to take the chance that was given to me, and I even hope to be able to prolong my stay in Leuven. I want to take this opportunity to thank a number of people who helped and supported me during the last years.

First and foremost I wish to thank my advisor Frits Spieksma, who gave me the opportunity to come to Leuven in order to start a PhD under his supervision. I greatly appreciate all the time and effort he has put into this work, which probably would not have existed without him. William Arthur Ward, an American scholar, once said: *"The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great*

teacher inspires." Frits has been a great teacher to me, and the enthusiasm with which he tries to solve any problem related to operations research is inspiring. I owe him many thanks for passing on this enthusiasm to me.

During the third year of my PhD I visited the Eidgenössische Technische Hochschule in Zürich where I joined the research group of Thomas Erlebach. Thomas introduced me to the topic of computer network modelling, which has become a substantial part of this thesis. I am very grateful to him for giving me the opportunity to join his research group for a few weeks. In addition, I want to thank him for his useful remarks and suggestions for improving this thesis.

Many thanks also to the other members of my committee: Erik Demeulemeester, Yves Crama, Willy Gochet, and Koos Vrieze gave me a lot of comments and suggestions, which improved both the quality and the readability of this work.

Next, I would like to thank Lieve Janssens, Julie Callaert, Rosanne Vanpée, and Caroline van Eeckhout, for keeping my mind off work every now and then, for listening to all my complaints when I was having an off-day, but most of all for all the fun we had in the past years. I hope that we will continue our shopping days and our girls nights, even if we don't see each other at work every day anymore.

Furthermore, I want to thank everyone from the research groups ORSTAT and Operations Management for being such great colleagues. Special thanks to everyone who has accompanied me to the conferences I attended, to Roselinde Kessels for being a great office mate for all these years, and to Jan Adem and Roel Leus for their patience in explaining all the rules and regulations concerning the finalization of my PhD.

I also want to thank the members of the Computer Engineering and Networks Laboratory of the ETH in Zürich for making my stay there such a pleasant one. Special thanks go to Danica Vukadinović for her time and effort in trying to familiarize me with the topic of computer network modelling and Internet topologies.

Finally, I wish to thank all my friends and family, especially my parents, for their continuous support and encouragement. And last, but always first, Stef – thank you for everything, but most of all simply for being there. It means the world to me.

Thank you all.

Linda Moonen
September 2003

Abstract

This thesis is situated in the field of combinatorial optimization. More specifically, we focus on solution methods for solving a number of graph-theoretical optimization problems. First we give a short introduction to the field of integer programming and state of the art solution techniques. The remainder of this thesis can be roughly split into two parts. The first part is dedicated to the problem of partitioning partially ordered sets. The second part deals with the structure and the connectivity of the Internet.

The problem of partitioning a partially ordered set into a minimal number of chains, such that each element belongs to at least one chain, is a basic and fundamental problem in operations research. Dilworth (1950) showed that it is solvable in polynomial time, and that the minimum number of chains needed to cover all elements of X is equal to the value of a maximum antichain. We generalize this problem by assuming that the chains must have bounded size, and we propose a number of exact algorithms for solving this problem. We apply these algorithms to a real-world application of this problem encountered at a manufacturing company in the Netherlands. One of the interesting outcomes of this work is that, in this real-world setting, we were able to identify a special structure in the problem instances. It turns out that these problem instances have a property called bounded clique-width, which allows us to design a polynomial time algorithm for these special instances that works extremely well.

Next we further generalize the problem by assuming that a weight is given for each element in the partial order. The problem is now to partition the partial order into a minimum-weight set of chains (where the weight of a chain is defined as the largest weight of all elements in this chain), such that the size of each chain is bounded by a given parameter. We give a number of lower and upper bounds on the value of an optimal solution, and we propose a 2-approximation algorithm for solving this problem.

In the second part of this thesis we study the connectivity of the Internet. The Internet has become very popular over the past decades. Of course it is very important for Internet-based communication to be efficient, secure, and reliable, especially in a time of viruses that can take down entire computer networks within a few hours. In order to study the structure and the connectivity of the Internet, we model it as a graph. A natural means for analyzing the connectivity of a graph, is to determine the maximum number of vertex-disjoint paths and the size of a minimum vertex-cut for any pair of nodes. It is well known that these problems are solvable in polynomial time for ordinary graphs, but for the Internet-graph, this is not the case. Since the notion of a valid path is somewhat different for the Internet-graph, both problems become $\mathcal{NP}$-hard. We propose a number of exact algorithms for solving both problems, and compare their results with the results from two 2-approximation algorithms proposed by Erlebach et al. (2005).

Samenvatting

Deze thesis situeert zich in het onderzoeksgebied van operationeel onderzoek. We richten ons op methoden om een aantal graaf-theoretische optimalisatie problemen op te lossen. Allereerst geven we een korte introductie in lineair en integer programmeren en bespreken we enkele oplossingsmethoden die in deze thesis worden gebruikt. Het vervolg van deze thesis kan grofweg in twee delen worden opgesplitst. In het eerste deel komt het opdelen van een *partial order* aan bod. In het tweede deel bestuderen we de structuur en de connectiviteit van het Internet.

Het opsplitsen van een partial order in een zo klein mogelijk aantal chains is een welbekend en fundamenteel probleem in het vakgebied van operationeel onderzoek. Dilworth (1950) toonde aan dat het probleem polynomiaal oplosbaar is en dat het minimum benodigde aantal chains gelijk is aan het aantal elementen in een maximale antichain. We generaliseren dit probleem door te stellen dat een chain niet meer dan een gegeven aantal elementen mag bevatten. We stellen een aantal exacte algoritmen voor om dit probleem op te lossen en passen deze toe op een specifiek probleem bij een productiebedrijf in Nederland. Een interessant resultaat van dit onderzoek is dat we bij de probleem instanties van dit productiebedrijf een speciale structuur konden vaststellen, gerelateerd aan het concept van de *clique width* van een graaf. Door deze structuur kunnen we aantonen dat het probleem, voor deze speciale instanties, polynomiaal oplosbaar is.

Vervolgens behandelen we een tweede generalisatie van het probleem, waarbij we aan elk element van de partial order een gewicht toekennen. Het probleem wordt dan om alle elementen op te delen in chains zodanig dat de som van de gewichten van de chains minimaal is. Hierbij wordt het gewicht van een chain gedefinieerd als het gewicht van het zwaarste element in de chain. Ook hier geldt de capaciteitsbeperking dat elke chain ten hoogste een gegeven aantal elementen mag bevatten. We geven een aantal ondergrenzen voor de waarde van de optimale oplossing en we stellen een 2-approximatie algoritme voor.

In het tweede deel van deze thesis bestuderen we de structuur en de connectiviteit van het Internet. Het Internet is de laatste decennia zeer populair geworden en de hoeveelheid data die via het Internet wordt verstuurd is enorm gegroeid. Het is zeer belangrijk dat communicatie die via Internet verloopt efficiënt, veilig en betrouwbaar is, zeker in een tijd waarin virussen binnen enkele uren enorme computer netwerken kunnen stilleggen. Om de structuur en de connectiviteit van het Internet te bestuderen, modelleren we het Internet als een graaf. Een veel gebruikte manier om de connectiviteit van een graaf te analyseren is door het maximale aantal paden en de minimale sneden de bepalen. Het is welbekend dat deze twee problemen polynomiaal oplosbaar zijn voor gewone grafen, maar voor een Internetgraaf is dat niet het geval. Aangezien de definitie van een pad in de graaf in deze context anders is dan bij normale grafen, zijn beide problemen voor Internet-grafen $\mathcal{NP}$-compleet. We stellen een aantal exacte algoritmen voor om deze problemen op te lossen en vergelijken de resultaten met de resultaten van twee 2-approximatie algoritmes voorgesteld door Erlebach et al. (2005).

Contents

"There are three men on a train. One of them is an economist and one of them is a logician and one of them is a mathematician. And they have just crossed the border into Scotland (I don't know why they are going to Scotland) and they see a brown cow standing in a field from the window of the train (and the cow is standing parallel to the train). And the economist says, 'Look, the cows in Scotland are brown.' And the logician says, 'No. There are cows in Scotland of which one, at least, is brown.' And the mathematician says, 'No. There is at least one cow in Scotland, of which one side appears to be brown.' And it is funny because economists are not really scientists, and because logicians think more clearly, but mathematicians are best."

From **The curious incident of the dog in the night-time** by Mark Haddon.

Chapter 1

Introduction

This thesis deals with exact and approximation algorithms for some specific combinatorial optimization problems. Some of these algorithms are based on integer linear programming formulations of these combinatorial problems. Therefore, we discuss in this first introductory chapter several techniques that we use in this thesis to solve integer programming problems. Section 1.1 gives a short introduction to linear and integer programming. In Section 1.2 we describe a number of solution methods, both exact and heuristic approaches, for solving integer programs. For a more complete overview of integer programming and IP solution techniques, we refer to Nemhauser and Wolsey (1988), or Wolsey (1998). Section 1.3 deals with the approximability of optimization problems, and in Section 1.4 we give an outline of the remainder of the thesis.

1.1 Linear and Integer Programming

During the last 50 years, linear programming has become a well-established tool for solving a wide range of optimization problems. Linear programming can be described as a process in which we transform a real-life problem into a mathematical model, and try to find methods for solving this model (Sierksma, 1996). In general, a linear program (LP) is written in matrix

notation as follows:

$$\begin{aligned} &\max\ cx \\ \text{s.t. }\ &Ax \leq b \\ &x \geq 0 \end{aligned}$$

Here, c is an n-dimensional row vector, A is an $(m \times n)$-matrix, b is an m-dimensional column vector and x is an n-dimensional column vector of *decision variables*.

In many practical situations (see the upcoming chapters), the decision variables are required to have integer values. For example, when we consider an application in which we want to minimize the number of vehicles needed to transport goods from one point to another, we would like to find an integral solution. An LP in which all variables must have integer values is called an integer (linear) program (IP), as shown in model (1.1). Furthermore, if all variables are required to equal 0 or 1, we have a binary integer (linear) program (BIP).

$$\begin{aligned} &\max\ cx \\ \text{s.t. }\ &Ax \leq b \\ &x \in \mathbb{N} \end{aligned} \tag{1.1}$$

Many problems have only a finite number of alternative choices and consequently can be stated as combinatorial optimization problems. These problems can often be formulated as integer programming models where some or all of the variables can take on only a finite number of alternative possibilities (the word *combinatorial* referring to the fact that only a finite number of alternative feasible solutions exists). We describe the knapsack problem and the assignment problem, two well-known examples of combinatorial optimization problems. For a detailed overview of combinatorial optimization

problems and techniques, we refer to Schrijver (2003), or Papadimitriou and Steiglitz (1998).

The Knapsack Problem

In the Knapsack Problem, we are given n items and a knapsack with capacity b. Each item i has a value v_i and a weight w_i, $i = 1, \ldots, n$. The goal is to find a subset of items with maximal value, such that the total weight of all items in the subset does not exceed the knapsack capacity b. We define a decision variable x_i for each item i ($i = 1, \ldots, n$) such that $x_i = 1$ if item i is selected, and $x_i = 0$ otherwise. The IP formulation can be given as follows.

$$\max \sum_{i=1}^{n} v_i x_i \tag{1.2}$$

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \leq b \tag{1.3}$$

$$x_i \in \{0, 1\} \quad (\forall i) \tag{1.4}$$

We maximize the total value of the selected items in the objective function (1.2). Constraint (1.3) states that the capacity of the knapsack can not be exceeded, and constraints (1.4) are the integrality constraints for the decision variables.

The Assignment Problem

In the Assignment Problem, there are n people available to perform n jobs. Any person can be assigned to perform any job, incurring a cost that may vary according to the assignment. So the cost of assigning person i to perform job j is given by c_{ij}. The goal is to find a minimum cost assignment. If we define decision variables x_{ij} equal to 1 if person i performs job j, and 0 otherwise, we can formulate the problem as follows.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1.5}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad (\forall i) \tag{1.6}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (\forall j) \tag{1.7}$$

$$x_{ij} \in \{0, 1\} \quad (\forall i, j) \tag{1.8}$$

The objective (1.5) is the minimization of the total cost of the assignment. The first set of constraints (1.6) state that each person performs exactly one job, and the second set of constraints (1.7) state that each job is performed by one person. Finally, constraints (1.8) are the 0-1 constraints on the decision variables.

1.2 Solving IP Problems

A natural idea for solving IPs is called *rounding*, i.e., solving the IP as if it were an LP, and rounding the solution to integer values. However, this method of solving IPs is often inadequate, as shown in the following example.

Example – Rounding an LP

Consider the following IP:

$$\begin{aligned} \max\ & 11x_1 + 10x_2 \\ \text{s.t.}\ & 7x_1 + 5x_2 \leq 35 \\ & -x_1 + 2x_2 \leq 2 \\ & x_1, x_2 \in I\!N \end{aligned} \tag{1.9}$$

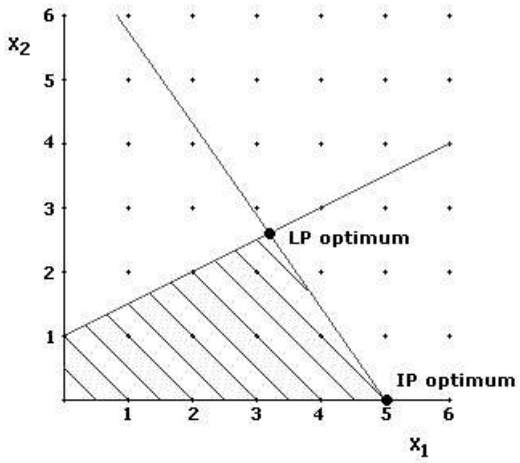

Figure 1.1: Rounding an LP

In Figure 1.1 we see that the LP-solution to problem (1.9) is $(\frac{60}{19}, \frac{49}{19})$, which is quite different from the IP-solution $(5, 0)$. ■

So in order to solve integer programming problems, we need other methods than just simply rounding the linear programming solution. When trying to find methods to optimize complex IP problems, often one has to make a trade-off between quality and efficiency. One can try to find the optimal solution to an IP using an *exact algorithm*. The downside of using exact solution methods is that, in general, the running times can be very high. On the other hand, one can settle for a *heuristic algorithm*. The running time for heuristic methods is usually small, but there is no guarantee about the quality of the solution. *Approximation algorithms* are heuristic approaches, with the difference that for these methods, a performance guarantee can be given. In this section we discuss a number of different solution methods for solving IPs, that are used in this thesis. In Section 1.2.1, we discuss a number of exact solution methods, and Section 1.2.2 deals with heuristics.

1.2.1 Exact Solution Methods

In this section we describe a number of exact solution methods for solving IPs. For a more detailed overview we refer to Wolsey (1998).

Branch-and-Bound

Branch-and-bound is probably the most widely used approach for solving IP-models to optimality. Basically, the branch-and-bound method solves an optimization problem by partitioning its solution space into smaller subsets (*branching*). Each of these subsets is further analyzed and partitioned, until a (better) feasible solution is found or it is determined that the subset does not contain a better solution. Lower and upper bounds can be used either to prove optimality of the current solution, or to discard certain submodels from further consideration (*bounding*). The branching procedure can be nicely visualized by means of a *branching tree*. In this branching tree, the root node corresponds to the original problem, and each child node represents a submodel created by partitioning the solution space of its parent (see Johnson et al. (2000)).

Example – Branch-and-Bound

In this example we will solve the IP-model given by (1.9) by branch-and-bound. First, we have to calculate an upper bound, and we do this by solving the LP-relaxation of (1.9). The solution to the LP-relaxation is an upper bound on the integer optimum. We find that the LP-solution is equal to $(\frac{60}{19}, \frac{49}{19})$ with value $\frac{1150}{19} \approx 60.5$, which we use as an upper bound during the branch-and-bound process. Since no feasible solution has been found yet, we initialize the lower bound at $-\infty$.

For the branching part, we have to select a fractional variable. Both x_1 and x_2 have fractional value in the LP-solution, so we arbitrarily choose x_1 for the branching procedure. We have to divide the solution space of the origi-

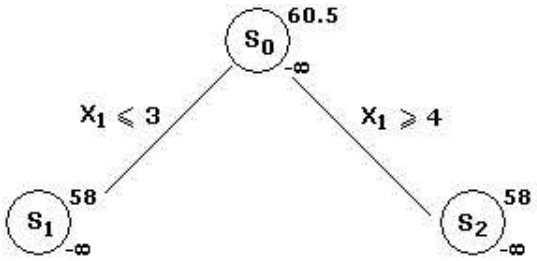

Figure 1.2: The first branching step

nal problem, S_0, in such a way that we cut off the fractional LP-solution, but don't cut off any integral solutions. We can do this by creating two submodels S_1 and S_2 as follows: $S_1 = \{x \in S_0 : x_1 \leq 3\}$ and $S_2 = \{x \in S_0 : x_1 \geq 4\}$. So we add two children to the root node of the branching tree, and we solve the LP-relaxation of the two subproblems. In both nodes we find a fractional solution with value 58, as shown in Figure 1.2.

If we continue the branch-and-bound process, we eventually arrive at the branching tree shown in Figure 1.3. From this figure, we can see that in node S_3, an integer solution is found with value 53, which means that we don't have to investigate this node any further, and that we can change the value of the lower bound to 53. In node S_4, the LP is infeasible, so we can discard this node. If, in some node, we would find a solution which has a value that is smaller than the lower bound, we can discard this node, since we can't find a better solution in this node than the currently best solution. Discarding such a node is called *fathoming* a node. After finishing the tree, we see that the best solution is found in node S_8, with value 55. ■

In order to be able to use branch-and-bound efficiently to solve an IP, we need two things. For the branching step, we need a set of solutions that can be partitioned into mutually exclusive sets. For the bounding step, we need an algorithm for calculating an upper bound (or a lower bound in case of a minimization problem) on the cost of any solution in a subset

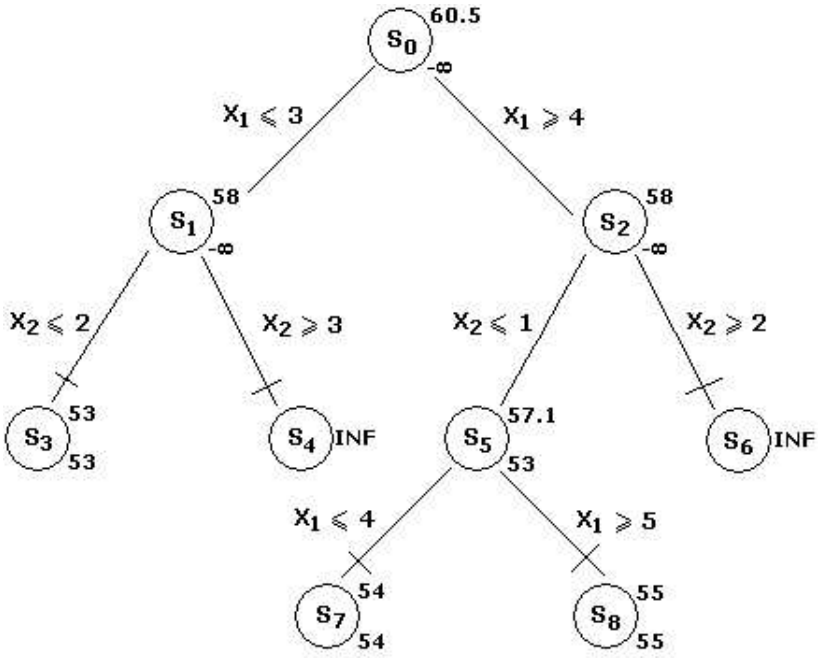

Figure 1.3: The entire branching tree

(Papadimitriou and Steiglitz, 1998).

Branch-and-Price

In the branch-and-bound approach, we usually solve a relaxation (the LP-relaxation for instance) of the original IP in order to obtain a bound on the integer optimum. However, it is not always easy to solve a relaxation optimally. For instance, if the number of variables is very large, common LP-solvers are no longer able to solve the LP-models. Branch-and-price is a technique for solving integer programs with a huge number of variables. We refer to Barnhart et al. (1998) or Vanderbeck and Wolsey (1996) for a thorough description of this technique. Basically, the branch-and-price process can be split into two phases. First, we solve the LP-relaxation of the problem by using *column generation*. Then, after having found the optimal LP-solution, we branch in order to find the integer optimum.

So, first we have to solve the LP-relaxation of the problem. Since the number of variables is very large, we start with a small subset S of all variables and consider the restriction of the LP to the variables in S. We call this restricted version of the LP the *restricted master problem* (RMP). We solve RMP using an LP-solver and obtain a solution to RMP and its corresponding dual solution. Now, we need to check whether this dual solution is also feasible in the dual program that includes constraints for all variables in order to test whether the primal solution is optimal. In other words, we need to check whether there exists a violated constraint in the dual. In the literature, this is called the *pricing problem* (Vanderbeck and Wolsey, 1996). If we find a violated constraint in the dual, we add the corresponding primal variable to RMP and solve it again. We repeat this procedure until we can no longer find a violated constraint in the dual, which means we have solved the LP-relaxation optimally. In general, this LP-solution is fractional, so we have to branch in order to find the integer optimum.

In the branching procedure, we partition the solution space in order to create a number of smaller subproblems. For each of these subproblems, we can solve the LP-relaxation again. In order to be able to use column generation throughout the branching tree, we need to find an appropriate way to partition the solution space. The way in which the solution space is divided usually differs from problem to problem. The complete branch-and-price procedure is depicted in Figure 1.4.

Branch-and-Cut

Branch-and-cut is a solution approach similar to branch-and-price. As mentioned above, in the branch-and-price procedure, we usually have a huge number of variables. In contrast, the branch-and-cut method is commonly used when trying to solve IPs with a huge number of constraints (see Johnson et al. (2001) or Caprara and Fischetti (1997)).

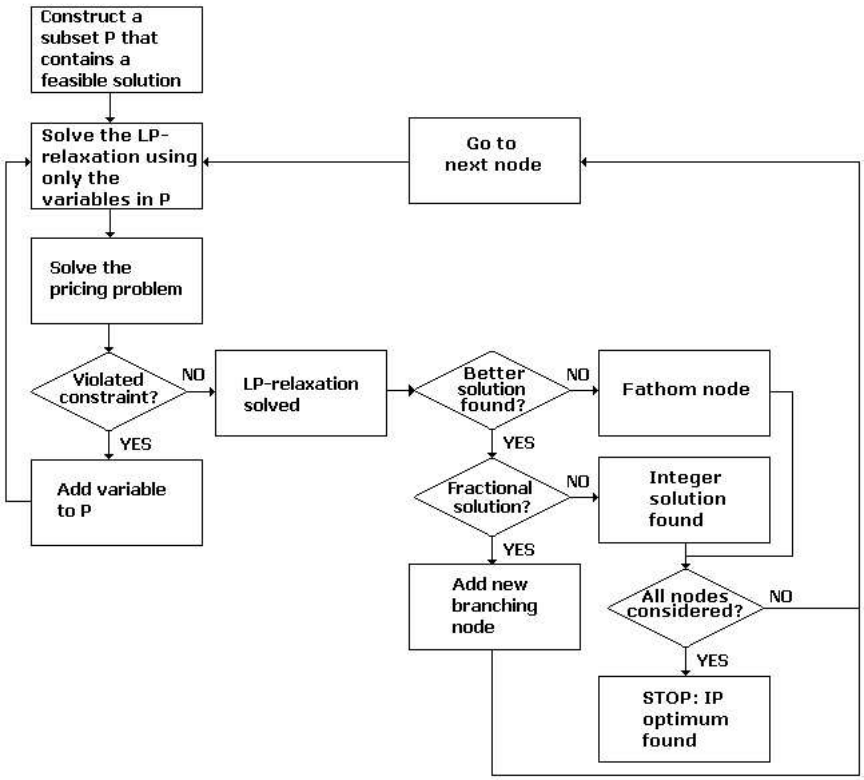

Figure 1.4: Branch-and-Price procedure

Again, we start by solving a restricted version of the LP-relaxation. We take a small subset of all constraints, and we solve the LP-relaxation restricted to these constraints. In order to determine whether the solution found is optimal, we try to find a valid inequality that is violated. This problem is known as the *separation problem* (see for example Wolsey (1998) or Ladányi et al. (2001)). If we can identify such an inequality, we add it to the restricted master problem (RMP). We continue this process until no violated inequality can be found, which means that the solution to the RMP is also optimal to the LP-relaxation. In general, this solution is fractional, and we need to branch in order to find the integer optimum.

Dynamic Programming

Dynamic programming is an approach developed by Bellman (1957) to solve sequential, or multi-stage, decision problems, but the approach is also applicable for decision problems where this sequential property is induced solely for computational convenience. It is, like branch-and-bound, a way of decomposing problems that are hard to solve into smaller subproblems that are easier to solve. The solution to the original problem is obtained recursively, either by working backward from the end of the problem to the beginning, or forward from the beginning to the end (see Denardo (1982)).

We explain the dynamic programming approach by applying it to the shortest path problem. Suppose we are given a directed graph $G = (V, E)$ with nodes $V = \{1, 2, \ldots, n\}$ and edges E, each edge (i, j) having a non-negative length $\ell(i, j)$ associated to it. At the end of the dynamic programming algorithm, we have an $(n \times n)$-matrix D in which an entry $D_{i,j}$ is the length of a shortest path from node i to node j in G.

The algorithm constructs a sequence of matrices $D^0, D^1, \ldots, D^n$ (with $D^n = D$). For each k, $1 \leq k \leq n$, $D^k_{i,j}$ is the length of a shortest i-j path when

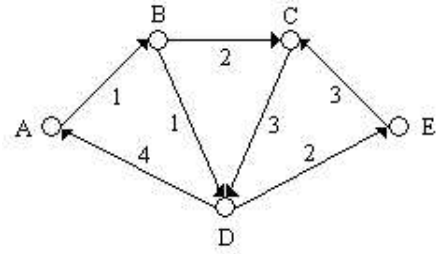

Figure 1.5: An instance of the shortest path problem

only the nodes $\{1, 2, \ldots, k\}$ can be used as interior nodes on the path. We initialize D^0 as follows.

$$D^0_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } (i,j) \notin E \\ \ell(i,j) & \text{if } (i,j) \in E. \end{cases}$$

We can then formulate a dynamic programming recursion for calculating the values of D^{k+1}, using the values from D^k.

$$D^{k+1}_{i,j} = min(D^k_{i,j}, D^k_{i,k+1} + D^k_{k+1,j}) \tag{1.10}$$

We illustrate this approach with an example.

Example – Dynamic Programming

Suppose we are given the directed graph G shown in Figure 1.5, with $V = \{A, B, C, D, E\}$, and edges with edge length as shown in the picture.

After initialization, the matrix D^0 looks like this:

$$D^0 = \begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ \infty & 0 & 2 & 1 & \infty \\ \infty & \infty & 0 & 3 & \infty \\ 4 & \infty & \infty & 0 & 2 \\ \infty & \infty & 3 & \infty & 0 \end{pmatrix}$$

Using the dynamic programming recursion (1.10), we ultimately get the following result for D^5:

$$D^5 = \begin{pmatrix} 0 & 1 & 3 & 2 & 4 \\ 5 & 0 & 2 & 1 & 3 \\ 7 & 8 & 0 & 3 & 5 \\ 4 & 5 & 7 & 0 & 2 \\ 10 & 11 & 3 & 6 & 0 \end{pmatrix}$$

The matrix D^5 represents the solution to our problem. For example, we see from entry $D^5_{A,E}$ that the shortest path from A to E has length 4. ■

1.2.2 Heuristic Solution Methods

So far we discussed a number of exact solution approaches for solving integer programs. In this section we present a special type of heuristic approach, being *approximation algorithms*. Since we do not use any other types of heuristics in this thesis, we will not discuss them here. For an overview of heuristic methods for solving IPs, we refer to Aarts and Lenstra (1998) or Silver (2004).

Approximation Algorithms

Approximation algorithms are an approach for solving $\mathcal{NP}$-hard optimization problems. Since it is unlikely that there can ever be efficient exact algorithms solving $\mathcal{NP}$-hard problems (unless $\mathcal{P} = \mathcal{NP}$), one can settle for non-optimal solutions, but require them to be found in polynomial time.

Unlike heuristics, which usually find reasonable good solutions reasonably fast, one wants provable solution quality and provable run time bounds.

Assume we have a polynomial time algorithm $\mathcal{A}$ for solving a maximization problem. We say $\mathcal{A}$ is an ϵ-approximation algorithm if, for every problem instance I,

$$\mathcal{A}(I) \geq \epsilon \cdot OPT(I).$$

Here, $\mathcal{A}(I)$ denotes the value of the solution found by algorithm $\mathcal{A}$, $OPT(I)$ is the value of the optimal solution for instance I, and $\epsilon \leq 1$ is the approximation ratio or performance guarantee. In case of a minimization problem, $\mathcal{A}$ is an ϵ-approximation algorithm if, for every problem instance I,

$$\mathcal{A}(I) \leq \epsilon \cdot OPT(I),$$

where $\epsilon \geq 1$.

In the next example we give a well-known example of an approximation algorithm, for the *node cover* problem: given a graph $G = (V, E)$, find the smallest possible set $C \subseteq V$ such that $(u, v) \in E \Rightarrow u \in C$ or $v \in C$.

Example – Approximation algorithm

Consider the following algorithm for the node cover problem. Given is a graph $G = (V, E)$.

ALGORITHM NODE COVER

---

1. Compute a maximum matching M^* in G.
2. For each edge $(u, v) \in M^*$, add both u and v to the node cover C.

The set C computed by the algorithm is definitely a node cover, since any edge not in M^* shares an endpoint with some edge in M^* (otherwise this

edge could have been added to the matching, and hence M^* is not a maximum matching). Can we say anything about the performance of the algorithm? Let $|M^*|$ be the size of a maximum matching, and $|C^*|$ the size of a minimum node cover. We know that $|M^*| \leq |C^*|$. (If this was not the case, some vertex $v \in C^*$ must touch two edges in M^*, and this is in contradiction with the construction of M^*.) Therefore, the node cover produced by the algorithm, which has size $2 \cdot |M^*|$, is within a factor 2 of the optimum. So the algorithm is a 2-approximation algorithm for the node cover problem. ■

For a detailed overview of approximation algorithms, we refer to Vazirani (2001) or Ausiello et al. (1999).

1.3 Approximability and Proving $\mathcal{APX}$-hardness

As discussed earlier, it is very unlikely to find efficient algorithms for solving optimization problems that are $\mathcal{NP}$-hard to optimality (since this would imply that $\mathcal{P} = \mathcal{NP}$). If we want to solve these problems efficiently, we have to accept that the solution we find is not guaranteed to be an optimal solution. Approximation algorithms, as described in Section 1.2.2, provide us with an efficient way to solve $\mathcal{NP}$-hard problems, while giving a provable performance guarantee. In this section we describe, informally, the complexity class $\mathcal{APX}$ in more detail, and we describe how to prove that a problem is $\mathcal{APX}$-hard. For a detailed description of approximation algorithms and their complexity, we refer to Ausiello et al. (1999).

For $\mathcal{NP}$-hard optimization problems, we are interested in finding approximation algorithms with an approximation ratio as close to one as possible. We are mainly interested in algorithms that give a *constant* approximation ratio. The complexity class $\mathcal{APX}$ contains all $\mathcal{NP}$-hard optimization problems that admit an efficient (i.e., polynomial) algorithm with a constant approximation ratio, and a problem is called *approximable* if it belongs to the class $\mathcal{APX}$ (Ausiello et al., 1999). For some optimization problems, we can do

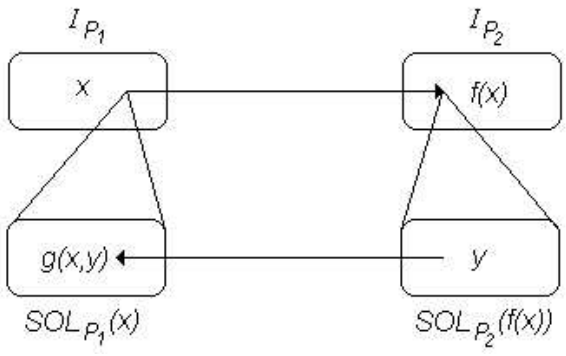

Figure 1.6: Reduction between two optimization problems (Ausiello et al., 1999)

better than a constant-factor approximation algorithm. A *polynomial-time approximation scheme*, or PTAS, is a family of algorithms such that, for any $\epsilon > 0$, there exists an algorithm in the family that produces a solution within a factor ϵ of the optimum and that runs in polynomial time (Moret, 1998). Optimization problems that belong to the class $\mathcal{APX}$ and that don't permit a PTAS are called $\mathcal{APX}$-hard.

In order to prove that a problem $\mathcal{P}$ is $\mathcal{APX}$-hard, we have to show an approximation preserving reduction from a known $\mathcal{APX}$-hard problem to $\mathcal{P}$. Usually, a so-called AP-reduction (Ausiello et al., 1999) or an L-reduction (Papadimitriou and Yannakakis, 1991) is used. In such a reduction we need a function f mapping instances of problem $\mathcal{P}_1$ to instances of problem $\mathcal{P}_2$. Next, we also need a function g to map a solution to problem $\mathcal{P}_2$ back to a solution to problem $\mathcal{P}_1$. Figure 1.6 from Ausiello et al. (1999) gives a schematic view of such a reduction. Here, I_{P_k} denotes the set of all instances of problem P_k, and $SOL_{P_k}(x)$ denotes the set of all feasible solutions to a problem instance x of problem P_k.

1.4 Outline of the Thesis

The remainder of this thesis consists of two parts. The first part (Chapters 2 and 3) is dedicated to the problem of partitioning a partial order. The second part (Chapter 4) deals with the connectivity of the Internet. All chapters can be read independently of the others.

In Chapter 2 we discuss the problem of partitioning a permutation graph into cliques of bounded size. We present two exact algorithms for solving this problem. The first algorithm is a branch-and-price algorithm based on an IP formulation. The second algorithm is a branch-and-bound algorithm based on the concept of the clique width of a graph, and was motivated by a special structure present in the real-world problem instances. This algorithm is, as far as we are aware, the first implementation of an algorithm based on bounded clique width. The performance of both algorithms is tested using a number of real-world and randomly generated instances. Chapter 2 is joint work with Frits Spieksma.

Chapter 3 deals with the problem of partitioning a weighted partially ordered set into chains of bounded size. We show that this problem is $\mathcal{APX}$-hard, and derive lower bounds on the value of the optimum. Based on these lower bounds, we exhibit a 2-approximation algorithm for solving the problem, and we show that it is tight. Chapter 3 is joint work with Frits Spieksma.

In Chapter 4 we study the structure and the connectivity of the Internet. First we explain how the Internet can be modelled as a graph. Next, we analyze the connectivity of the Internet-graph by computing the maximum number of vertex-disjoint paths and the size of a minimum cut for any pair of nodes. Although these problems are solvable in polynomial time for regular graphs, this is not the case for Internet-graphs. Since the notion of a valid path is somewhat different for the Internet-graph, both problems become $\mathcal{NP}$-hard. We present exact and approximation algorithms for solving both

problems, and we give computational results for all algorithms. Chapter 4 is joint work with Thomas Erlebach, Frits Spieksma, and Danica Vukadinovič.

Finally, in Chapter 5 we describe a number of topics for future research.

Chapter 2

Exact Algorithms for a Loading Problem with Bounded Clique Width

In this chapter we discuss a special pallet-loading problem, which we encountered at a manufacturing company in the Netherlands. In graph-theoretical terms, the problem is equivalent to partitioning a permutation graph into bounded-size cliques. We formulate the problem as an integer program, and present two exact algorithms for solving it. The first algorithm is a branch-and-price algorithm based on the integer-programming formulation; the second one is an algorithm based on the concept of bounded clique width. The latter algorithm was motivated by the structure present in the real-world instances. Test results are given, both for real-world instances and randomly generated instances. As far as we are aware, this is the first implementation of an algorithm based on bounded clique width.

This chapter is organized as follows. Section 2.1 introduces the problem, and describes the application of the problem encountered at Bruynzeel Storage Systems. Section 2.2 proposes a branch-and-price approach based on a set-partitioning formulation of the loading problem (see Barnhart et al. (1998)

for a description of branch-and-price algorithms). We show that the pricing problem is solvable in polynomial time, and that we can generalize this approach to partial orders. Section 2.3 is devoted to an exact enumeration algorithm for a special case of the loading problem. This algorithm is based on the concept of bounded clique width. In Section 2.5, we show computational results from the branch-and-price algorithm and from the algorithm based on bounded clique width. Section 2.6 contains the conclusions.

2.1 Introduction

Consider the following situation. Given is a set S of distinct points (or items) in the plane, $S = \{1, 2, ..., n\}$. For any pair of points $i, j \in S$, we say that i is *smaller* than j ($i \prec j$) if and only if

$$(x_i \leq x_j) \wedge (y_i \leq y_j),$$

where x_i and y_i denote the x- and y-coordinate of i, $i \in S$. Furthermore, we call $R \subseteq S$ a *feasible subset* (or a *stable pallet*, see Section 2.1.2) if and only if for any two points $i, j \in R$ either $i \prec j$ or $j \prec i$.

The problem we consider is as follows: given the set S and an integer $B \leq n$, partition S into as few feasible subsets as possible such that each subset contains no more than B points. For reasons to become clear in Section 2.1.2, we refer to this problem as the *loading problem*.

Of course, if B is not present in the input of our problem, the resulting problem is solvable in polynomial time since it is a special case of Dilworth's chain decomposition theorem (Dilworth, 1950). However, Jansen (2003) proves that for each fixed $B \geq 6$, our loading problem is $\mathcal{NP}$-hard. We now proceed by describing the relation to graph theory.

2.1.1 Relation to Graph Theory

The loading problem is intimately related to problems in graph theory. We first discuss the concept of a *permutation graph* before we explain this relation in more detail.

Definition 1 (Golumbic, 1980). A graph $G = (V, E)$ is a permutation graph if $(i, j) \in E \Leftrightarrow (i - j) \cdot (\pi_i^{-1} - \pi_j^{-1}) < 0$, where $\pi = [\pi_1, \pi_2, \ldots, \pi_{|V|}]$ is a permutation of the vertices of G, and π_i^{-1} is the position of i in π.

In other words, a graph is a permutation graph when one can exhibit a sequence of the vertices such that there is an edge (i, j) between two vertices (with $i > j$) if and only if i precedes j in the sequence.

Example – Permutation graphs

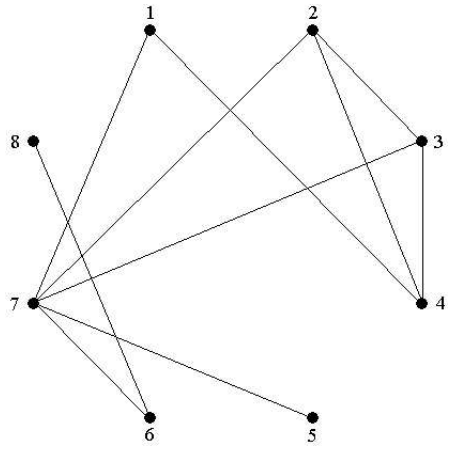

Figure 2.1: Example of a permutation graph.

The graph shown in Figure 2.1 is the permutation graph corresponding to the permutation $\pi = [4, 7, 1, 3, 2, 5, 8, 6]$. ■

Now, suppose we are given a set of points $S = \{1, 2, \ldots, n\}$. We can build a graph $G = (V, E)$ as follows: for each point $i \in S$ there is a node in V, and two nodes are adjacent if and only if $i \prec j$ or $j \prec i$.

Claim 1. *The resulting graph G is a permutation graph.*

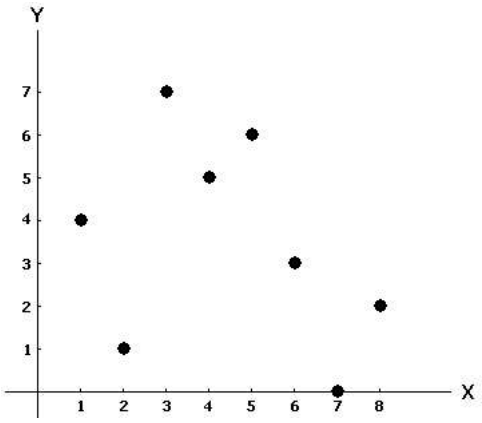

Figure 2.2: Point set S.

Proof: Suppose we are given the point set as shown in Figure 2.2. First, project all points to the y-axis, and label them $1, \ldots, n$ in such a way that the point with the largest y-coordinate gets the lowest label. Next, project all original points to the x-axis. Now, write the numbers 1 to n from left to right. Under this sequence, write the numbers again, in the order in which they appear in the projection on the x-axis. Finally, connect the points in the upper sequence with the points in the lower sequence that have the same number. This results in the *matching diagram* of S (see Figure 2.3). Notice that line segments between the points i and j intersect if and only if i and j appear in reversed order in the lower sequence (Golumbic, 1980). This corresponds exactly to the definition of a permutation graph, so the permutation π is equal to the lower sequence. So, for the example in Figure 2.3 the permutation π is equal to $[4, 7, 1, 3, 2, 5, 8, 6]$, corresponding to

the permutation graph depicted in Figure 2.1. □

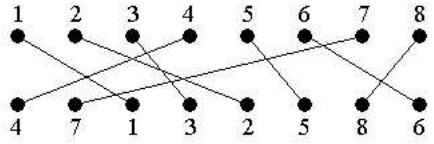

Figure 2.3: Matching diagram corresponding S.

Observe that a feasible subset in S corresponds to a clique in G. Thus, the loading problem is equivalent to the problem of partitioning a permutation graph into cliques of bounded size.

2.1.2 Motivation

Our motivation comes from a real-world setting encountered at Bruynzeel Storage Systems (BSS), a manufacturing company in the Netherlands. BSS produces mobile storage systems that are delivered worldwide. To construct such a system, BSS produces many rectangular shaped boxes, each with a specific length and a specific width. We refer to such a rectangular shaped box as an *item*. A single storage system may consist of up to 200 items. Furthermore, there are no standard sizes, so each customer specifies his or her own requirements. The height of an item, however, is identical for all items. The items have to be loaded onto pallets for transportation to the clients. It is allowed to place items on top of each other in layers; however, the number of items per layer is restricted to one. Since the items all have identical heights, it follows that the height of the trucks that transport the pallets determines the maximum number of layers of each pallet. We denote this number by B (in the case of BSS, $B = 12$). A crucial feature involves the *stability* of the pallets (see for example Bischoff (1991)). BSS stipulated that no larger item could be placed on top of a smaller item. More precisely, both the length and the width of an item placed in some layer must

be smaller than or equal to the length and the width of the item placed in the layer directly under it. This restriction ensures that pallets arrive in good shape at their final destination (Moonen, 2001). In order to achieve an efficient usage of the trucks it is important to minimize the total number of pallets used.

> **Remark:** Notice that the application described here allows for identical items, whereas we assume in the loading problem that all items are pairwise distinct. It is not difficult to see, however, that all results presented later are valid for the case where identical items are allowed.

2.1.3 Related Work

Thus, our problem can be seen as a type of pallet-loading problem (PLP). Pallet- or container-loading problems concern the optimal packing of small items into large containers or pallets. The terms *pallets* and *containers* are used interchangeably in most studies, although there is an important difference between them. When loading goods onto a pallet, the notion of the *stability* of the loading schemes is far more important than when the goods are to be loaded into a container, since we cannot make use of the upstanding walls that we have when loading items into a container, so the stability must be guaranteed (Bischoff, 1991).

Although the problem discussed in this paper is a type of pallet-loading problem, it is quite different from customary PLPs. Usually, PLPs are three-dimensional packing problems that concern the optimal packing of a number of small items into a large container, with the objective to minimize the volume of product packed on the pallet. The problem we discuss is a two-dimensional problem. Also, restricting the pallets such that there can only be one item on each layer is unusual for general PLPs. Indeed, when it is possible to store multiple items on a layer, the resulting packings may

have a very complex structure. Thus, our loading problem is a very specific pallet-loading problem, and different from ordinary PLPs.

Most of the research on PLPs has concentrated on the case where a set of identical items has to be loaded onto a single pallet. Dyckhoff (1990) gives a detailed overview of the different types of PLPs and proposes a number of solution approaches for solving them. In more recent work, Morabito and Morales (1998) developed a heuristic based on a recursive procedure to solve the problem, and G and Kang (2001) propose a heuristic that can be applied to relatively large instances (more than 5000 items). Letchford and Amaral (2001) give a detailed analysis of upper bounds for the PLP. Also, some heuristics have been suggested for solving the PLP with non-identical items. Scheithauer and Terno (1996b) developed a heuristic combining a general branch-and-bound framework with optimal two-dimensional loading patterns. More recently, Terno et al. (2000) proposed an algorithm that uses the G4-heuristic introduced in Scheithauer and Terno (1996a), and combine this with a branch-and-bound procedure.

Our loading problem also occurs in the field of mutual exclusion scheduling problems (Baker and Coffman, 1996; Jansen, 2003). In such a scheduling problem a graph is given such that each vertex corresponds to a job, and an edge between two vertices indicates that the two corresponding jobs are incompatible, i.e., cannot be processed at the same time. Assuming that we have B processors available, and that each job needs a single time unit, computing a schedule such that the latest job finishes as soon as possible is an instance of the loading problem (provided that the conflict graph is a permutation graph).

This type of problem is also known under the name of *batch scheduling with job compatibilities*. Batch scheduling involves a machine that can process multiple jobs simultaneously. Often, jobs within a batch need to be pairwise compatible, and these compatibilities can be expressed using a compatibility

graph. Boudhar (2003) and Finke et al. (2004) study different variants of these batch scheduling problems when the compatibility graph is bipartite or an interval graph.

Lee et al. (2004) describe an application from the steel-industry where molten steel is turned into solid steel. For this application, the problem is equivalent to partitioning an interval graph into bounded size cliques.

Another related problem, described in Felsner and Wernisch (1998), involves covering as many points in a planar point set as possible, using a given number of chains.

2.2 A Branch-and-Price Algorithm

In this section we formulate the loading problem as an integer program and we describe a branch-and-price algorithm for solving it (see eg. Barnhart et al. (1998)). We use terminology corresponding to the application, i.e., we use "items" (instead of points), and "stable pallets" or simple "pallets" (instead of feasible subsets).

2.2.1 Problem Formulation

We introduce a decision variable x_k for every possible stable pallet k, such that:

$$x_k = \begin{cases} 1 & \text{if stable pallet } k \text{ is in the solution} \\ 0 & \text{otherwise.} \end{cases}$$

Next, we define $\mathcal{I}_k$ as the set of all items contained in pallet k. Using a set-partitioning formulation, we get the following model:

$$\min \sum_k x_k \tag{2.1}$$

$$\text{s.t.} \sum_{k:i\in\mathcal{I}_k} x_k = 1 \qquad \forall i \tag{2.2}$$

$$x_k \in \{0,1\} \qquad \forall k \tag{2.3}$$

The objective (2.1) is to minimize the total number of pallets needed to pack all items. Constraints (2.2) state that each item has to be in exactly one pallet, and constraints (2.3) are the zero-one constraints on the x_k variables.

2.2.2 Column Generation

Since the number of variables in formulation (2.1)-(2.3) is exponentially large, we employ column generation to solve its LP-relaxation without having to enumerate all variables. In the column-generation process we start with a small subset of the variables that contains a feasible solution. All other variables are implicitly assigned the value zero. The subproblem constructed in this way is called the *restricted master problem* (RMP). We solve the LP-relaxation of RMP, and then we have to determine whether the solution found is optimal for the master problem. To do this, we have to try to identify a variable with negative reduced cost, or, a violated constraint in the dual. The dual of the LP-relaxation of formulation (2.1)-(2.3) is given below.

$$\max \sum_i u_i \tag{2.4}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}_k} u_i \leq 1 \qquad \forall k \tag{2.5}$$

We can now formulate an expression for the reduced costs of a variable x_k:

$$1 - \sum_{i\in\mathcal{I}_k} u_i$$

Thus, given a feasible solution to the LP-relaxation and the corresponding dual variables, the pricing problem boils down to the following question:

$$\exists k \quad \text{such that} \sum_{i \in \mathcal{I}_k} u_i > 1?$$

Lemma 1. *The pricing problem can be solved in polynomial time.*

Proof: We construct a directed graph $D = (V, A)$ as follows: There is a node in V for each item, and there is a source s and a sink t in V. We draw an arc from node i to node j if for the corresponding items $i \prec j$ holds; this arc has length u_j. Also, there is an arc from s to each node $i \in V$ with length u_i, and an arc from each node i to t with length 0. Observe that the constructed graph is acyclic. We now define $d^p(j)$ to be the length of a longest path from s to j using at most p arcs ($j = 1, ..., n$). These longest paths can be calculated in polynomial time using the following dynamic-programming recursion:

$$\begin{aligned} d^p(j) &= \max(\max_{i:(i,j)\in A} d^{p-1}(i) + u_j, d^{p-1}(j)) \\ \text{with } d^1(s) = 0 \text{ and } d^1(j) &= u_j \quad \forall j \neq s \quad (p = 2, ..., B) \end{aligned} \tag{2.6}$$

Let us show by induction that the values $d^p(j)$ computed by the dynamic programming recursion (2.6) satisfy their definition. The case $p = 1$ is trivial, so let us assume that it holds for $p = \ell - 1$. Consider now a longest path from s to j using at most ℓ arcs. If this path contains exactly ℓ arcs, there is a predecessor of j in this path, say j', such that the longest path from s to j' using at most $\ell - 1$ arcs consists of the first $\ell - 1$ arcs in the longest path from s to j. By induction the latter value (i.e., the length of a longest path from s to j' using at most $\ell - 1$ arcs) is recorded in $d^{\ell-1}(j')$. If this path contains less than ℓ arcs, it follows that $d^\ell(j) = d^{\ell-1}(j)$. It follows that (2.6) computes $d^\ell(j)$ correctly. Thus, testing whether $d^{B+1}(t)$, which is the length of a longest path from s to t containing at most $B+1$ edges (and therefore at most B interior nodes) is larger than 1 amounts to answering

the pricing problem. □

> **Remark.** One could consider a situation where a weight p_k is given for each possible pallet k, and next minimize total weight. For instance, in terms of the application, it would be quite natural to define the weight of pallet k as the area of its largest item. Indeed, it is easy to exhibit examples where minimizing total area is not equivalent to minimizing the number of pallets needed. Notice that in this case the efficient solvability of the pricing problem is preserved since by computing $d^B(j)$ using (2.6), and next comparing each value with the corresponding area of item j determines whether a variable with negative reduced costs exists.

The solution found by applying the column-generation procedure will in general be a fractional solution. We now sketch a branching structure in order to find the integer optimum.

2.2.3 Branching Procedure

Ryan and Foster (1981) proposed a general branching rule for set-partitioning problems, where two rows of the constraint matrix have to be covered by the same column in one branch, and by different columns in the other branch. For our problem, this would mean that two items have to be packed onto the same pallet in one branch, and on different pallets in the other branch. Since it is difficult to force two items to appear on the same pallet, or to appear in different pallets, we use a slightly modified version of the Ryan-Foster branching rule (see also Vanderbeck (1994)). We partition the solution space based on *the order* in which items are packed onto a pallet. Two items are called *successors* if they are packed on the same pallet such that one item lies directly above the other. The branching rule we use is then as follows: in one branch two items i and j are forced to appear as successors on a

pallet, and in the other branch items i and j cannot be successors on a pallet. Similar modifications of the Ryan-Foster rule have also been used in the field of airline crew scheduling (see for example Desrosiers et al. (1991) or Vance et al. (1997)).

Lemma 2. *If a given LP-solution x is fractional, there exists a pair of items i and j which are successors on a certain pallet k (denoted by $suc(i,j)_k$) with $1 > x_k > 0$, such that*

$$0 < \sum_{k:i,j \in \mathcal{I}_k \wedge suc(i,j)_k} x_k < 1$$

Proof: Suppose that the lemma is false. Consider a fractional pallet k (i.e., a pallet whose corresponding variable has a fractional value) and suppose that it contains m items, $\{1, 2, ..., m\}, m \geq 2$, such that no other fractional pallet contains more than m items (notice that such a pallet always exists). For the lemma to be false, it must hold that

$$\sum_{k:\ell,\ell+1 \in \mathcal{I}_k \wedge suc(\ell,\ell+1)_k} x_k = 1, \text{for } \ell = 1, ..., m-1.$$

Thus all fractional pallets that contain item ℓ must also contain item $\ell + 1$ as its successor ($\ell = 1, ..., m-1$). Further, since the LP-solution x satisfies constraints (2.2) for each item $\ell = 1, ..., m$, it follows that $x_k = 1$. Thus, the LP-solution is integral, which is in contradiction with our assumption of a fractional solution, and proves the correctness of the lemma. □

When an optimal, fractional LP-solution has been found, we identify two items i and j for which the sum of all pallets where i and j are successors lies between 0 and 1. In the integer optimum, these two items will either be successors on a pallet, or they will not. So, given two items i and j, we branch as follows. In one branch we modify the directed graph D in such a

way that items i and j have to be successors. We do this by deleting all arcs (i, p) for $p \neq j$ and all arcs (p, j) for $p \neq i$. In the second branch, we make sure that items i and j can never be successors in a solution by deleting arc (i, j) from D. In our algorithm we employ this branching step repeatedly to find an integer solution to our problem. Notice that this branching scheme keeps the problem structure intact, which allows us to use column generation throughout the branch-and-bound tree.

The entire branch-and-price algorithm can be summarized as follows.

Branch-and-Price Algorithm PartitionPermutationGraph

---

1. Calculate an initial solution consisting of a set $\mathcal{S}$ of stable pallets with value $V_{\mathcal{S}}$, and let $V^* = V_{\mathcal{S}}$. Create a list L and add to L a branching node corresponding to the input graph D and the set $\mathcal{S}$ of pallets.

2. $L = \varnothing$?

   YES: STOP. An optimal solution is found with value V^*.

   NO: Select the next branching node from L (i.e., the branching node that was added most recently to L), remove it from L, and go to step 3.

3. Solve the LP-relaxation using only those variables that correspond to a stable pallet in $\mathcal{S}$.

4. Solve the pricing problem. Is there a variable with negative reduced costs?

   YES: Add this variable to $\mathcal{S}$ and go to step 3.

   NO: An optimal solution to the LP-relaxation is found with value V_{LP}. Continue with step 5.

5. $V_{LP} < V^*$?

YES: Continue with step 6.

NO: Go to step 2.

6. Is the solution to the LP-relaxation integral?

YES: $V^* = V_{LP}$. Go to step 2.

NO: Select two items i and j for which the sum of all pallets where i and j are successors is fractional. Create two new branching nodes, with their corresponding input graph D (i.e., the graph obtained by deleting the respective edges) and set of pallets $\mathcal{S}$, corresponding to the assumption that either i and j are successors, or they are not. Add these nodes to L, and go to step 2.

Remark. The branch-and-price approach sketched in Sections 2.2.2 and 2.2.3 remains valid for so-called partial orders. Consider the problem of decomposing a partial order into a minimum number of chains, such that each chain contains no more than B elements (see Shum and Trotter (1996), and Chapter 3). We will refer to such a chain as a *B-chain*. We claim that this problem can be tackled using the approach sketched here. First, one easily verifies that the formulation (2.1)-(2.3) goes through by substituting the word "B-chain" for "stable pallet" in the definition of the x_k-variables. Second, the efficient solvability of the pricing problem (Lemma 1) depends on the fact that the digraph contains no directed cycles. This property is preserved when we consider partial orders. Finally, notice that the branching rule also holds in this more general setting, and it follows that the branch-and-price approach remains valid.

2.3 An Algorithm Based on Bounded Clique Width

In this section we propose an enumeration algorithm that is based on a property of some of the instances encountered at BSS. It turns out that, in some

instances, many items have the same length. We exploit this property in this section by assuming that the number of different lengths in an instance is bounded by a given parameter K. In other words, we assume that in the input of the problem an additional parameter K is present; we refer to this variant of our loading problem as problem $P(K)$.

As a motivating example we first explore the case $K = 2$. We define n_j as the number of items of length j, and we assume that $L_1 < L_2$, where L_j is the jth length. Furthermore, define p and q as follows:

$$p = n_1 \bmod B$$

$$q = n_2 \bmod B$$

Consider now the items with length L_1, and let w_1 be the width that corresponds to the pth smallest item. Then consider the items of length 2, and let w_2 be the width that corresponds to the qth largest item. Notice that the optimal solution of problem $P(2)$ has value $\lceil \frac{n}{B} \rceil$ or $\lceil \frac{n}{B} \rceil + 1$ since $\lceil \frac{n_1}{B} \rceil + \lceil \frac{n_2}{B} \rceil \leq \lceil \frac{n}{B} \rceil + 1$. In fact, we characterize below in Proposition 1 when instances of problem $P(2)$ have value $\lceil \frac{n}{B} \rceil$ or $\lceil \frac{n}{B} \rceil + 1$.

Proposition 1. *The optimal solution of problem $P(2)$ has value $\lceil \frac{n}{B} \rceil$ if and only if $\lceil \frac{n}{B} \rceil = \lceil \frac{n_1}{B} \rceil + \lceil \frac{n_2}{B} \rceil$ or $w_1 \leq w_2$.*

We now consider problem $P(K)$ in case of a fixed value of K. We assume that the lengths are ordered such that $L_1 < L_2 < ... < L_K$. In Section 2.3.1 we focus on the concept of (bounded) clique width. Section 2.3.2 describes an exact algorithm for problem $P(K)$.

2.3.1 Clique Width

A property of graphs that has received wide attention recently is clique width. This property was first introduced by Courcelle et al. (1993); a related concept called NLC-width has been introduced by Wanke (1994). The

reason for this attention is the fact that important graph-theoretic problems (like maximum clique or independent set) can be solved in polynomial time for graphs with bounded clique width.

Before giving a definition of the clique width of a graph, we start with some notation. A labeled graph $G = (V_G, E_G, \ell_G)$ is a graph whose vertices are labeled by some mapping $\ell_G : V_G \to \mathbb{N}$. A labeled graph $H = (V_H, E_H, \ell_H)$ is a subgraph of G if $V_H \subseteq V_G, E_H \subseteq E_G$ and $\ell_H(v) = \ell_G(u)$ for all $v \in V_H$.

Informally, the notion of clique width of a graph G can be described as in Definition 2. For formal definitions, see Courcelle and Olariu(2000) or Brandstädt et al. (2004).

Definition 2. The concept of the clique width of a graph G can be described using the following four operations:

Operation 1. Creation of a vertex labeled with some integer i (the vertex is said to have label i); we denote the single vertex graph with label ℓ as $\bullet_\ell$.

Operation 2. Disjoint union of two vertex-labeled graphs: given $G_1 = (V_1, E_1, \ell_1)$ and $G_2 = (V_2, E_2, \ell_2)$, define the disjoint union $G = (V_G, E_G, \ell_G)$ as follows:

- $V_G = V_1 \cup V_2$
- $E_G = E_1 \cup E_2$
- $\ell_G = \begin{cases} \ell_1(v) & \text{if } v \in V_1 \\ \ell_2(v) & \text{if } v \in V_2 \end{cases}$

We denote the disjoint union of two graphs G_1 and G_2 as $G_1 \oplus G_2$.

Operation 3. Adding an edge between each vertex with label i and each vertex with label j, $i \neq j$; we denote this operation by $\eta_{i,j}$.

Operation 4. Relabel each vertex with label i by label j; we denote this by $\rho_{i\to j}$.

The minimum number of labels needed to construct a graph G using these operations is the clique width of G.

Example – Clique width

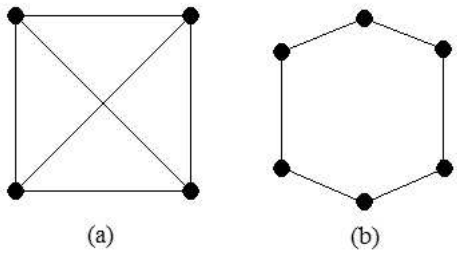

Figure 2.4: (a) K_4: the complete graph with 4 vertices. (b) C_6: the cycle with 6 vertices.

The clique width of K_4, the complete graph with 4 vertices (see Figure 2.4a), is 2. K_4 can be defined by the following operations:

$$\rho_{b\to a}(\eta_{a,b}(\bullet_b \oplus \rho_{b\to a}(\eta_{a,b}(\bullet_b \oplus \rho_{b\to a}(\eta_{a,b}(\bullet_a \oplus \bullet_b))))))$$

The clique width of C_6, the cycle with 6 vertices (see Figure 2.4b), is equal to 3, and we can describe it by the following operations (Courcelle and Olariu, 2000):

$$\rho_{c\to a}(\rho_{b\to a}(\eta_{b,c}(\bullet_c \oplus \rho_{c\to a}(\eta_{a,c}(\bullet_c \oplus \eta_{a,b}(\bullet_a \oplus \bullet_b) \oplus \eta_{a,b}(\bullet_a \oplus \bullet_b)))))) \blacksquare$$

Permutation graphs in general have unbounded clique width (Brandstädt and Lozin, 2003), however, in case of $P(K)$ we have the following:

Lemma 3. *A graph associated to an instance of $P(K)$ has clique width at most $K+1$.*

Proof: We prove the lemma by exhibiting a sequence of operations. First, we order the vertices according to the width of the associated item in decreasing order. In case of a tie, the vertex with the highest length goes first.

For each vertex $i = 1, \ldots, n$, letting the vertex correspond to an item with length L_j, $1 \leq j \leq K$, we perform the following operations:

- create vertex i and label it $K+1$, using operation 1.
- add vertex i to the graph, using operation 2, i.e., $G := (V \cup \{i\}, E)$.
- connect the vertex with label $K+1$ to all vertices with label $j, j+1, \ldots, K$, using operation 3 repeatedly.
- relabel the vertex with label $K+1$ by label j using operation 4.

Observe that this construction guarantees that each vertex that corresponds to an item with length L_j is connected to all vertices that correspond to items that have length L_j or larger. Thus, the resulting permutation graph corresponds to an instance of $P(K)$. □

> **Remark**: It is easy to verify that the graphs corresponding to instances of $P(K)$ do not have bounded tree width.

We can now state the following theorem:

Theorem 1. *Problem $P(K)$ is solvable in polynomial time.*

Proof: This result follows from Lemma 3 above and Theorem 2 in Espelage et al. (2001), which states that the problem of partitioning a graph into cliques of bounded size is solvable in polynomial time for graphs with

bounded clique width. □

Notice, however, that the approach in Espelage et al. (2001) is quite general, and does not lead to a ready-to-use algorithm. We propose such an algorithm in the next section.

2.3.2 An Algorithm for $P(K)$

We describe an exact algorithm for problem $P(K)$ that, for a fixed B and K, runs in polynomial time. We now state some preliminaries.

Definition 3. A pallet is called *pure* when it contains B items of the same length; otherwise a pallet is called *mixed* (notice that a pallet with fewer than B items of the same length is called a *mixed* pallet).

Definition 4. The length of an item i is denoted by ℓ_i; its width by w_i.

Property 1. A solution of problem $P(K)$ is said to have property 1 if it contains no more than 2^K mixed pallets.

Property 2. A solution of problem $P(K)$ is said to have property 2 if no item r in a mixed pallet can be replaced by an item s from a pure pallet, with $\ell_s = \ell_r$ and $w_s < w_r$, in such a way that both pallets remain feasible.

Definition 5. We call a solution to problem $P(K)$ *minimal* if it simultaneously satisfies properties 1 and 2.

Lemma 4. *There exists an optimal solution to problem $P(K)$ that is minimal.*

Proof: Consider some optimal solution to problem $P(K)$. By interchanging and transferring items, we show that there is an optimal solution that is minimal. If there exists an item r occurring in a mixed pallet that can

be replaced by an item s from a pure pallet with $\ell_s = \ell_r$ and $w_s < w_r$, we interchange these items so that property 2 is satisfied. To see that there exists an optimal solution that satisfies property 1, observe that an upper bound on the maximum number of mixed pallets with different length sets is equal to 2^K. Therefore, if we have found a solution containing more than 2^K mixed pallets, there exist at least two pallets with identical length sets. We now show that, by interchanging some items between these pallets, we can alter the solution such that no pallets with identical length sets are present in the solution.

If, in an optimal solution, the number of mixed pallets that contain items of one single length exceeds K, we can transfer items in a straightforward way, and reduce the number of mixed pallets that have items of a same length to K.

If, in an optimal solution, the number of mixed pallets that contain items of at least two different lengths exceeds $2^K - K$, we can reduce the number of mixed pallets that have items of at least two different lengths by transferring items. For this, we first define

$$p^A_{L_i} : \text{the smallest width of an item of length } L_i \text{ from pallet } A$$

$$q^A_{L_i} : \text{the largest width of an item of length } L_i \text{ from pallet } A$$

Observe that, when we discard the size requirement of a pallet, all items of length L_i can be transferred from a pallet A to a pallet C if the following two conditions hold:

$$q^A_{L_i} \leq p^C_{L_{i+1}} \tag{2.7}$$

$$p^A_{L_i} \geq q^C_{L_{i-1}} \tag{2.8}$$

Now, consider an optimal solution that contains more than $2^K - K$ mixed pallets with items of at least two different lengths. Then there exist two

pallets A and C with identical length sets, say $L_1, L_2, ..., L_m$ ($m \geq 2$). We claim that there exist two lengths L_i and L_j such that either all items of L_i can be transferred from pallet A to pallet C, or all items of L_j can be transferred from C to A. This implies that we can construct an alternative optimal solution by interchanging items between A and C such that these pallets no longer have identical length sets.

Without loss of generality we assume that $p^A_{L_m} \geq q^C_{L_{m-1}}$. (If this would not be the case, we have $p^A_{L_m} < q^C_{L_{m-1}}$. We know, by feasibility of pallets A and C, that $q^A_{L_{m-1}} \leq p^A_{L_m}$ and $q^C_{L_{m-1}} \leq p^C_{L_m}$. From this it follows that $p^C_{L_m} \geq q^A_{L_{m-1}}$, and we can simply change the order of the two pallets to arrive at our assumption that $p^A_{L_m} \geq q^C_{L_{m-1}}$.)

Since $p^A_{L_m} \geq q^C_{L_{m-1}}$, the items of length L_m from pallet A can be transferred to pallet C. Now, we have to find a length such that items from pallet C can be transferred to pallet A. In order to do so, we have to find a length for which conditions (2.7) and (2.8) hold. Assume that we cannot find such a length; we then show that we will ultimately arrive at a contradiction, proving that such a length does exist.

Claim 2. *If items of length $L_1, ..., L_j$ cannot be transferred from C to A, it follows that $q^C_{L_j} > p^A_{L_{j+1}}, j = 1, ..., m-1$.*

Proof: We use induction to prove this claim. Consider the case $j = 1$. We can transfer the items of L_1 from pallet C to pallet A if $q^C_{L_1} \leq p^A_{L_2}$. Since the items of L_1 are the smallest items, condition (2.8) does not apply, since there is no length smaller than L_1. We assumed that we could not transfer items from pallet C to pallet A, so it must hold that $q^C_{L_1} > p^A_{L_2}$. Next, suppose the claim is true for $j = \ell - 1$; is it then true for $j = \ell$? Since we are not able to transfer the items of L_ℓ from C to A, at least one of the inequalities $q^C_{L_\ell} \leq p^A_{L_{\ell+1}}$ and $p^C_{L_\ell} \geq q^A_{L_{\ell-1}}$ must be violated. But we know by induction that $q^C_{L_{\ell-1}} > p^A_{L_\ell}$, which, together with $p^C_{L_\ell} \geq q^C_{L_{\ell-1}}$ and $p^A_{L_\ell} \geq q^A_{L_{\ell-1}}$, implies

$p^C_{L_\ell} \geq q^A_{L_{\ell-1}}$. Hence, it follows that $q^C_{L_\ell} > p^A_{L_{\ell+1}}$, proving the claim. □

Claim 2 states that if we cannot transfer items of length $L_1, ..., L_j$ from C to A, it follows that $q^C_{L_j} > p^A_{L_{j+1}}, j = 1, ..., m-1$. However, this in in contradiction with our assumption that $p^A_{L_m} \geq q^C_{L_{m-1}}$, meaning that there does exist a length for which we can transfer items from C to A. This means that we can reduce the number of mixed pallets that contain items of at least two different lengths to at most $2^K - K$, implying that property 1 is satisfied. This proves that there indeed exists an optimal solution that is minimal. □

Lemma 4 implies that there exists an optimal solution such that for each $j = 1, ..., K$ the number of items of length L_j present in mixed pallets (denoted by s_j) equals $s_j = n_j - \alpha_j B$, for some $\alpha_j \in \{0, 1, \ldots, \lceil \frac{n_j}{B} \rceil\}$, and satisfies $\sum_{j=1}^K s_j \leq B2^K$. Now, given a set of possible s_j values with $\sum_{j=1}^K s_j \leq B2^K$, we enumerate all possible minimal solutions. We do this using the concept of a *partial* solution.

Definition 6. A *partial* solution is a family of 2^K sets of items such that each set corresponds to a stable pallet and such that each item occurs at most once in the family.

To each partial solution we associate a minimum length ℓ_{min}, that is the minimum length L_j for which fewer than s_j items are present in the current partial solution. Furthermore, we associate a *minimal item* to each stable pallet with fewer than B items in the partial solution. This minimal item is the item with length ℓ_{min} that has minimal width, and can be feasibly added to that pallet.

We now give an algorithm that finds an optimal minimal solution to problem $P(K)$, assuming that a set of s_j values, satisfying $\sum_{j=1}^K s_j \leq B2^K$, is given. First, we deal exclusively with constructing the mixed pallets. For this, we start with a partial solution that has 2^K empty pallets, and we gradually

fill - in many different ways - these pallets.

ALGORITHM ENUM

---

1. Start with the initial partial solution that consists of 2^K empty pallets. We associate length L_1 to this solution (assuming $s_1 > 0$; if $s_1 = 0$, we associate length L_j to the solution, with j minimal while satisfying $s_j > 0$), and set as minimal item for each pallet the smallest item of L_1. Go to step 2.

2. For each different pallet in the current partial solution, generate a new partial solution by adding to this pallet its minimal item. Notice that we get at most 2^K new partial solutions, since there are 2^K pallets in the old partial solution. Go to step 3.

3. Associate to each partial solution the new minimum length L_j for which fewer than s_j items are present, and associate to each pallet its new minimal item. If $\sum_{j=1}^{K} s_j$ items are present in the new partial solution, go to step 4. Otherwise, go to step 2.

4. For each final partial solution, i.e., for each partial solution where $\sum_{j=1}^{K} s_j$ items are assigned, verify whether each pallet in the solution is a mixed pallet. If not, simply discard the solution. Go to step 5.

5. Complete each final partial solution to a feasible solution by adding the remaining items in pure pallets in a straightforward way. Output the solution that contains the smallest number of pallets. STOP.

By associating a node to each partial solution and connecting two nodes if one partial solution is constructed by adding a single minimal item to the other, a tree results. We refer to this as the *tree of partial solutions*.

Example – Algorithm ENUM

Consider a problem instance containing 9 items, with lengths and widths as shown in Figure 2.5.

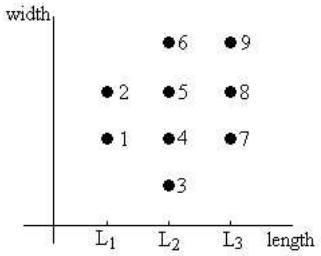

Figure 2.5: Problem instance for algorithm ENUM

Assume that $B = 3$, $K = 3$, and that the following s_j-values are given: $s_1 = 2$, $s_2 = 1$, and $s_3 = 0$. The tree of partial solutions corresponding to this problem instance is shown in Figure 2.6. ■

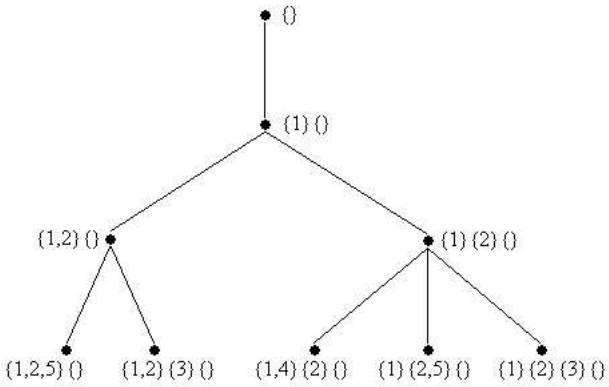

Figure 2.6: Tree of partial solutions

Lemma 5. *A solution produced by algorithm ENUM is minimal.*

Proof: We verify whether a solution found by ENUM satisfies properties

1 and 2. Obviously, it satisfies property 1. Now suppose that the solution found does not satisfy property 2; that is, there exists at least one item r that is present in a mixed pallet, that could be interchanged with an item s satisfying $\ell_s = \ell_r$ and $w_s < w_r$. Let r be the smallest interchangeable item and consider the step in the algorithm when we added item r to a pallet. Apparently, we could have added item s at that time. But that implies that item r was not a minimal item for that pallet. Hence, such a solution cannot have been generated by the algorithm. □

Lemma 6. *Any minimal solution is generated by algorithm ENUM.*

Proof: Consider a minimal solution that is not generated by ENUM. Remove from this solution all pure pallets, so that a final partial solution S remains. So each final partial solution generated by ENUM differs from S. Consider the tree of partial solutions. Let us find a set of paths in this tree: starting with the initial solution, follow a branch to a next partial solution if it puts an item in a pallet if in S the same item is in the same pallet. Notice that no path makes it until the end (since S was not generated by ENUM). So let us consider a partial solution for which we cannot follow a branch anymore and that is not final. To this partial solution a length is associated, say the current length.

Consider now the minimal item of the current length of that partial solution that is used in S, and that has not been considered when we followed branches. Say that this is item d and that it is in pallet j in solution S. This pallet j has another item, say item c, serving as minimal item when we look at the branch from our current partial solution to the partial solution where pallet j receives an item (if $c = d$, we would have followed that branch). Thus, $c \prec d$. Now, since S is minimal it must use item c somewhere else (if S did not use c at all, we could replace d by c in S, contradicting the minimality of S (property 2)). Say item c is used in pallet j' ($j' \neq j$). If we look at the branch from our current partial solution to the partial solution

that gives j' another item, we know there is a minimal item that cannot be item c (otherwise we would have followed that branch). Thus, there is another item present in that partial solution, say item b, $b \prec c$. Again, b must be somewhere in S, say in pallet j''. Notice that $j'' \neq j'$ (for obvious reasons) but also $j'' \neq j$ (since c is minimal for j and $b \prec c$). Let us look at the pallet j'' and its minimal item given our current partial solution. It cannot be b (else we would have followed this branch), so it must be less than b, say a. Thus, a must be in S (otherwise we can interchange contradicting the minimality of S), say in j'''. Again, this pallet j''' is different from the previously considered pallets j'', j', and j (otherwise each of the pallets wouldn't have the minimal item they have). Continuing in this way, it leads to the conclusion that S has more than 2^K pallets, contradicting property 1; hence S is not minimal. □

Lemma 7. *The number of nodes in all trees of partial solutions generated by algorithm ENUM is bounded by* $(2^K)^{B2^K} n^K$.

Proof: The number of nodes generated by ENUM depends on the number of solutions generated. This number depends on the number of items that are present in mixed pallets. Property 1 implies that $\sum_{j=1}^{K} s_j \leq B2^K$. Hence, ENUM cannot generate more than $(2^K)^{B2^K}$ different solutions. Furthermore, ENUM has to be executed for each possible set of s_j values. Observe that for each s_j there are $O(\frac{n_j}{B})$ possible values, $j = 1, \ldots, K$, leading to $O(n^K)$ possible sets of s_j values for a fixed B. The result follows. □

Theorem 2. *The running time of algorithm ENUM is polynomial in case of a fixed* B *and a fixed* K.

Proof: From Lemma 7 we know that the number of nodes generated by ENUM is bounded by $(2^K)^{B2^K} n^K$. Furthermore, the calculations in a single node can be done polynomially. This means that, for a fixed B and a fixed K, we have a polynomial-time algorithm. □

2.4 The Price of Stability

The stability restriction posed on the pallets is valid in the context of the application from the field of pallet loading discussed in this chapter. However, in many other applications this restriction is unnecessary. Therefore, we consider in this section the problem which results from disregarding this stability restriction. For the problem with unit weights (i.e., minimizing the number of pallets), as discussed in this chapter, the generalization without stability constraints is not very interesting, since the solution to this problem is trivial (an optimal solution contains exactly $\lceil \frac{n}{B} \rceil$ pallets). Therefore, we consider the weighted problem, where each item has a weight corresponding to its area, and we want to minimize the total area of all pallets. In this setting, the area of a pallet equals the area of its largest item. (See also the remark at the end of Section 2.2.2.) The complexity of this problem with a capacity constraint on the number of items on a pallet is, as far as we are aware, unknown. However, in case this capacity constraint is relaxed (i.e., $B = n$), the problem is solvable in polynomial time, as will be explained hereunder.

So, we consider problem with weights corresponding to the area of an item. We refer to this generalization of the loading problem as the *weighted loading problem*. For the weighted loading problem with $B = n$, we have the following result.

Theorem 3. *The weighted loading problem with $B = n$ is solvable in polynomial time.*

Proof. Let D be the set of all dominating items (an item i is dominating if there is no item j such that $\ell_j \geq \ell_i$ and $w_j \geq w_i$), and assume that the items in D are ordered such that $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_m$ (m denotes the number of items in D). The proof is now based on two observations. First, any item that is non-dominating can be ignored, since we can always add the dominated item to the pallet containing the dominating item without changing

the area of the pallet. This means that we only need to consider the items in D. Second, if an item i belongs to the same pallet as another item j ($j > i$), we can also add the items $i+1, i+2, \ldots, j-1$ to this pallet without changing its area. Now, in order to solve the problem, we create a directed graph as shown in Figure 2.7: we have a source s, a sink t, and m layers of vertices. In each layer there are a number of vertices corresponding to feasible pallets: in layer k we have a vertex for all feasible pallets consisting of consecutive items that contain item k as smallest item (with respect to the length of the item). We connect the vertices as follows: there is an arc from the source s to each vertex in the first layer. Then, between layer k and layer $k+1$ the vertices are connected such that their corresponding pallets do not contain an item more than once. All vertices corresponding to a pallet containing item m are connected to the sink. The length of an arc from i to j corresponds to the area of the pallet corresponding to vertex j, and arcs directed to the sink have length zero. Now we can solve the problem by calculating a shortest path from s to t in this network. The solution to the weighted loading problem that corresponds to such a shortest path contains a pallet for each interior vertex on the s-t path (that is, each vertex on the path except for s and t). □

In Figure 2.7 an example is given. In this example we have $D = \{a, b, c\}$, with $\ell_a \leq \ell_b \leq \ell_c$.

2.5 Computational Experiments

In this section we discuss some issues concerning the implementation of the algorithms described in this chapter, and we show the computational results.

2.5.1 Implementation Issues

Both algorithms described in this paper are implemented on a 733 MHz computer with 128MB of physical memory. The algorithms are coded in

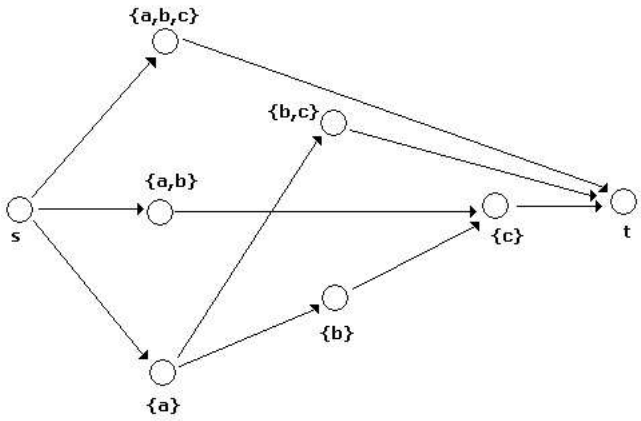

Figure 2.7: Directed graph corresponding to $D = \{a, b, c\}$.

C++, and in the branch-and-price algorithm, we use LINDO to solve the restricted master problems.

We use three data sets for the computational experiments. The first data set contains 50 real-world instances where the length and the width of each item was provided to us by BSS; the second data set contains 50 randomly generated instances where the length and the width of each item is uniformly distributed between 0 and 3000. The third data set also contains 50 randomly generated instances, but in this data set all instances have small clique width. More specifically, the number of different lengths is uniformly distributed between 5 and 15, whereas the width of the items is uniformly distributed between 0 and 3000. In all three data sets the number of items ranges from 0 to 200 (see Table 2.1). The number of items for an instance of the second and third data set follows a uniform distribution between 0 and 200 items per instance. We use different values for B, ranging from 3 to 15. In the real-world setting from BSS, $B = 12$.

In the branch-and-price algorithm, we use a heuristic to find a good starting solution, before starting the actual branch-and-price procedure. This

Table 2.1: Characteristics of the Data Sets

#Items	*#Instances (data set 1)*	*#Instances (data set 2)*	*#Instances (data set 3)*
0-40	10	9	4
40-80	15	13	13
80-120	10	7	15
120-160	7	12	8
160-200	8	9	10

starting solution is computed in a very straightforward way: all items are ordered, first according to their length (increasing), and second according to their width (also increasing). We start with the first item and put it in a pallet. Then we simply go down the list. If an item can be added to the current pallet, we add it; otherwise we continue with the next item. If a pallet contains B items, or if we are at the end of the list, we start a new pallet with the first available item and start this procedure over. To determine whether a solution generated by this heuristic is optimal, we use the lower bound $\lceil \frac{n}{B} \rceil$.

In the pricing problem, when trying to find new variables with negative reduced costs, we add one variable in each iteration of the longest-path procedure. This is the variable with reduced costs that are the most negative.

In the enumeration algorithm, we first compute a lower and an upper bound. The lower bound equals $\lceil \frac{n}{B} \rceil$, and the upper bound is equal to $\lceil \frac{n_1}{B} \rceil + \ldots + \lceil \frac{n_K}{B} \rceil$. If these bounds coincide, there exists an optimal solution with value $\lceil \frac{n}{B} \rceil$, and we do not need to run ENUM to find a solution.

Apart from computing the LP-relaxation and $\lceil \frac{n}{B} \rceil$, we compute a third lower bound, AC. AC stands for the size of a maximum antichain. In other words, AC is the optimal value of the loading problem if there is no restriction on B (i.e., $B = n$).

2.5.2 Results

For data set 1, the value of K ranges from 2 to 9, and in most instances (approximately 85%) K equals 2, 3, or 4. For the second data set, however, the value of K is very close to n. Thus, the clique width of the instances of data set 1 is small; this is not guaranteed for the instances of data set 2. Since the running time of the enumeration algorithm is exponential in K, the instances from data set 2 are very hard to solve for ENUM. In fact, none of the instances could be solved by ENUM in less than one hour of computation time, so for ENUM we present only the results of the first and third data set.

Table 2.2: Performance of lower bounds for data set 1

B	n	$\lceil \frac{n}{B} \rceil$	*Gap (%)*	*AC*	*Gap (%)*	*LP*	*Gap (%)*	*OPT*
3	≤ 40	11.90	0.00	2.50	77.52	11.47	3.69	11.90
	≤ 80	19.13	0.00	3.33	86.54	18.91	1.22	19.13
	≤ 120	36.90	0.00	2.90	94.54	36.40	1.35	36.90
	≤ 160	45.86	0.00	3.00	95.23	45.43	0.94	45.86
	≤ 200	60.50	0.00	2.38	96.68	60.08	0.68	60.50
6	≤ 40	6.20	0.00	2.50	56.60	5.72	7.54	6.20
	≤ 80	9.73	0.74	3.33	73.98	9.48	3.49	9.80
	≤ 120	18.70	0.00	2.90	89.21	18.20	2.66	18.70
	≤ 160	23.29	0.00	3.00	90.60	22.71	2.46	23.29
	≤ 200	30.38	0.00	2.38	93.38	30.04	1.09	30.38
9	≤ 40	4.40	0.00	2.50	39.00	3.97	8.77	4.40
	≤ 80	6.67	0.00	3.33	61.79	6.35	4.97	6.67
	≤ 120	12.80	0.00	2.90	84.26	12.18	4.83	12.80
	≤ 160	15.57	0.00	3.00	85.99	15.16	2.67	15.57
	≤ 200	20.38	0.00	2.38	90.13	20.04	1.61	20.38
12	≤ 40	3.50	5.83	2.50	30.83	3.23	12.02	3.70
	≤ 80	5.13	2.67	3.33	52.29	4.82	8.54	5.27
	≤ 120	9.60	0.00	2.90	79.00	9.10	5.22	9.60
	≤ 160	11.86	0.00	3.00	81.45	11.48	3.13	11.86
	≤ 200	15.50	0.00	2.38	87.03	15.02	3.07	15.50
15	≤ 40	2.70	11.67	2.50	20.00	2.81	9.57	3.10
	≤ 80	4.33	1.33	3.33	42.22	3.96	9.48	4.40
	≤ 120	7.60	0.00	2.90	73.39	7.29	3.98	7.60
	≤ 160	9.43	0.00	3.00	76.80	9.10	3.47	9.43
	≤ 200	12.63	0.00	2.38	84.06	12.19	3.52	12.96

Table 2.3: Performance of lower bounds for data set 2

B	n	$\lceil \frac{n}{B} \rceil$	*Gap (%)*	AC	*Gap (%)*	LP	*Gap (%)*	OPT
3	≤ 40	9.89	6.23	8.67	16.56	10.11	3.42	10.44
	≤ 80	18.77	0.40	11.69	36.85	18.54	1.67	18.85
	≤ 120	33.57	0.00	16.14	51.78	33.38	0.60	33.57
	≤ 160	47.92	0.00	20.83	56.41	47.47	0.95	47.92
	≤ 200	61.11	0.00	23.22	62.04	60.85	0.41	61.11
6	≤ 40	5.22	39.96	8.67	0.00	8.67	0.00	8.67
	≤ 80	9.62	16.82	11.69	0.64	11.72	0.43	11.77
	≤ 120	17.00	0.79	16.14	9.24	17.10	1.34	17.29
	≤ 160	24.17	0.38	20.83	15.13	23.90	1.35	24.25
	≤ 200	30.67	0.00	23.22	23.21	30.50	0.50	30.67
9	≤ 40	3.78	55.44	8.67	0.00	8.67	0.00	8.67
	≤ 80	6.62	42.06	11.69	0.00	11.69	0.00	11.69
	≤ 120	11.57	25.82	16.14	0.79	16.14	0.00	16.14
	≤ 160	16.33	23.43	20.83	4.03	21.42	0.00	21.42
	≤ 200	20.67	19.56	23.22	8.49	26.00	0.00	26.00
12	≤ 40	2.78	66.91	8.67	0.00	8.67	0.00	8.67
	≤ 80	5.08	55.62	11.69	0.00	11.69	0.00	11.69
	≤ 120	8.71	43.72	16.14	0.00	16.14	0.00	16.14
	≤ 160	12.42	38.88	20.83	0.00	20.83	0.00	20.83
	≤ 200	15.56	34.00	23.22	0.00	23.22	0.00	23.22
15	≤ 40	2.44	71.08	8.67	0.00	8.67	0.00	8.67
	≤ 80	4.23	63.02	11.69	0.00	11.69	0.00	11.69
	≤ 120	7.29	52.44	16.14	0.00	16.14	0.00	16.14
	≤ 160	10.00	50.90	20.83	0.00	20.83	0.00	20.83
	≤ 200	12.67	46.03	23.22	0.00	23.22	0.00	23.22

Tables 2.2, 2.3, and 2.4 give an overview of the performance of the three lower bounds: $\lceil \frac{n}{B} \rceil$, the size of a maximum antichain (AC), and the value of the LP-relaxation. In the first two columns we give the value of B and a range for the number of items. The following columns give the values of three lower bounds, and the gap (%) between the lower bound and the integer optimum, i.e., $\frac{OPT-LB}{OPT}$. The column labelled "OPT" denotes the value of the optimal integer solution.

From these tables we see that the LP-relaxation provides us with a good lower bound for all three data sets. However, the quality of the other lower bounds depends on the characteristics of the different data sets. For data

Table 2.4: Performance of lower bounds for data set 3

B	n	$\lceil \frac{n}{B} \rceil$	*Gap (%)*	*AC*	*Gap (%)*	*LP*	*Gap (%)*	*OPT*
3	≤ 40	10.25	0.00	6.00	36.31	9.67	6.16	10.25
	≤ 80	21.69	0.00	7.75	63.71	21.36	1.58	21.69
	≤ 120	34.33	0.00	8.93	73.56	34.13	0.56	34.33
	≤ 160	44.75	0.00	10.63	76.14	44.46	0.66	44.75
	≤ 200	62.90	0.00	9.50	84.76	62.57	0.52	62.90
6	≤ 40	5.00	20.00	6.00	0.00	6.00	0.00	6.00
	≤ 80	11.15	1.65	7.77	30.82	10.72	5.65	11.38
	≤ 120	17.47	0.32	8.93	48.27	17.07	2.63	17.53
	≤ 160	22.63	0.00	10.63	52.82	22.23	1.78	22.63
	≤ 200	31.80	0.00	9.50	69.85	31.28	1.60	31.80
9	≤ 40	3.75	39.29	6.00	0.00	6.00	0.00	6.00
	≤ 80	7.54	8.45	7.77	6.41	8.25	0.74	8.31
	≤ 120	11.93	1.56	8.93	25.47	11.63	3.93	12.13
	≤ 160	15.25	0.83	10.63	30.62	14.83	3.49	15.38
	≤ 200	21.50	0.00	9.50	55.49	20.86	2.99	21.50
12	≤ 40	3.00	51.43	6.00	0.00	6.00	0.00	6.00
	≤ 80	5.85	26.41	7.77	4.12	7.86	2.93	8.08
	≤ 120	8.93	11.83	8.93	13.88	9.99	2.83	10.27
	≤ 160	11.75	5.80	10.63	14.81	11.68	6.50	12.50
	≤ 200	16.20	1.11	9.50	41.99	15.67	4.36	16.40
15	≤ 40	2.50	58.57	6.00	0.00	6.00	0.00	6.00
	≤ 80	4.69	38.20	7.77	1.10	7.77	1.02	7.85
	≤ 120	7.27	22.95	8.93	9.48	9.35	4.31	9.73
	≤ 160	9.38	16.10	10.63	5.62	10.97	2.69	11.25
	≤ 200	13.10	3.48	9.50	30.56	12.93	4.85	13.60

set 1, the real-world problem instances, $\lceil \frac{n}{B} \rceil$ gives a good bound on the optimal value, while the quality of AC is very poor for this data set (see Table 2.2). This can be explained by the special structure in these instances, namely that the number of different lengths among the items (K) in these instances is very limited. Therefore, the value of AC will be very small for these instances, resulting in a poor quality of this lower bound. One could expect the same behavior of these lower bounds for data set 3, where we tried to mimic the structure found in the real-world problem instances. However, if we look at Table 2.4 we clearly see that this is not the case: the quality of both $\lceil \frac{n}{B} \rceil$ and AC is poor for these problem instances. This can be explained by the fact that the value of K is slightly larger for these

Table 2.5: Comparison of algorithms for data set 1

B	n	*Branch & Price*				*ENUM*			
		Nodes	*Time*			*Nodes*	*Time*		
			avg	*% ≤ 1 sec*	*max*		*avg*	*% ≤ 1 sec*	*max*
3	≤ 40	0.00	0.00	100.00	0.00	3.70	0.00	100.00	0.00
	≤ 80	29.07	0.69	86.67	6.19	33.93	2.52	93.33	37.76
	≤ 120	0.00	0.01	100.00	0.01	2.90	0.01	100.00	0.10
	≤ 160	0.00	0.02	100.00	0.03	3.14	2.04	85.71	14.27
	≤ 200	0.00	0.05	100.00	0.08	0.63	0.00	100.00	0.00
6	≤ 40	0.00	0.00	100.00	0.00	7.40	0.00	100.00	0.01
	≤ 80	0.07	0.03	93.33	0.36	23.87	0.00	100.00	0.03
	≤ 120	0.00	0.01	100.00	0.01	11.80	0.00	100.00	0.01
	≤ 160	0.00	0.02	100.00	0.03	4.29	0.01	100.00	0.09
	≤ 200	0.00	0.05	100.00	0.08	4.63	0.00	100.00	0.00
9	≤ 40	0.10	0.02	100.00	0.21	11.00	0.00	100.00	0.00
	≤ 80	10.08	4.87	86.67	66.94	24.27	0.00	100.00	0.01
	≤ 120	5.40	0.23	80.00	1.19	5.80	0.00	100.00	0.00
	≤ 160	1.43	2.26	85.71	15.66	20.57	0.00	100.00	0.01
	≤ 200	7.75	0.65	87.50	4.88	11.00	0.00	100.00	0.00
12	≤ 40	4.80	0.56	90.00	5.17	14.00	0.00	100.00	0.00
	≤ 80	9.00	6.64	80.00	80.19	25.94	0.00	100.00	0.00
	≤ 120	0.00	0.01	100.00	0.02	25.90	0.00	100.00	0.01
	≤ 160	7.57	0.96	85.71	6.63	31.43	0.00	100.00	0.01
	≤ 200	8.75	2.60	75.00	16.03	51.88	0.01	100.00	0.03
15	≤ 40	8.10	5.72	80.00	41.43	22.10	0.00	100.00	0.01
	≤ 80	0.07	1.35	86.67	18.82	29.07	0.00	100.00	0.01
	≤ 120	4.90	5.13	70.00	35.69	28.00	0.00	100.00	0.00
	≤ 160	10.71	6.89	71.43	43.01	61.14	0.00	100.00	0.02
	≤ 200	15.50	7.78	75.00	51.23	55.88	0.01	100.00	0.03

instances compared to the instances from data set 1, which results in a bad performance of $\lceil \frac{n}{B} \rceil$. However, since K is still much smaller than the optimal value for most instances, also the performance of AC is pretty bad. From Table 2.3 we see that for data set 2, the quality of AC is very good, in contrast to the performance of $\lceil \frac{n}{B} \rceil$ for this data set. Since these instances are randomly generated, the number of different lengths among all items is very large. Therefore the size of a maximum antichain is very close to the optimum for this data set.

Table 2.6: Performance of branch-and-price algorithm for data set 2

B	n	*Nodes*	*Time*		
			avg	*% ≤ 1 sec*	*max*
3	≤ 40	15.11	0.10	100.00	0.28
	≤ 80	46.08	7.23	38.46	77.00
	≤ 120	101.14	12.92	14.29	28.49
	≤ 160	165.75	41.43	25.00	143.06
	≤ 200	305.56	120.10	11.11	246.85
6	≤ 40	1.00	0.02	100.00	0.05
	≤ 80	12.31	4.50	76.92	47.75
	≤ 120	69.14	82.53	14.29	187.23
	≤ 160	51.83	65.36	25.00	201.33
	≤ 200	62.67	112.78	11.11	202.99
9	≤ 40	1.00	0.02	100.00	0.04
	≤ 80	1.00	0.14	100.00	0.29
	≤ 120	9.57	9.25	0.00	42.31
	≤ 160	10.67	25.40	0.00	78.39
	≤ 200	15.22	97.62	0.00	318.54
12	≤ 40	1.00	0.03	100.00	0.06
	≤ 80	1.00	0.37	84.62	1.83
	≤ 120	1.00	3.68	0.00	5.56
	≤ 160	13.17	43.54	0.00	374.16
	≤ 200	1.00	78.54	0.00	174.89
15	≤ 40	1.00	0.03	100.00	0.06
	≤ 80	1.00	0.37	84.62	1.81
	≤ 120	1.00	3.98	0.00	7.29
	≤ 160	1.00	16.20	0.00	48.59
	≤ 200	1.00	85.93	0.00	174.85

Tables 2.5, 2.6, and 2.7 compare the performance of the branch-and-price and the enumeration algorithm, in terms of computation time and number of nodes in the search tree. Again, the first two columns show the value of B and a range for the number of items. In the next columns we give the number of branching nodes visited in the search tree (for ENUM, this is the number of nodes in all trees of partial solutions), the average computation time (in seconds), the percentage of problem instances that is solved within one second of computation time, and the maximum computation time (in seconds). For Tables 2.5 and 2.7, these values are given both for the branch-and-price algorithm as well as for ENUM; Table 2.6 shows only the results

Table 2.7: Comparison of algorithms for data set 3

B	n	*Branch & Price*				*ENUM*			
		Nodes	*Time*			*Nodes*	*Time*		
			avg	*% ≤ 1 sec*	*max*		*avg*	*% ≤ 1 sec*	*max*
3	≤ 40	5.50	0.02	100.00	0.06	12.75	0.00	100.00	0.01
	≤ 80	65.00	2.18	38.46	5.02	15.00	0.12	100.00	0.56
	≤ 120	118.87	14.06	33.33	81.59	19.47	1.93	66.67	12.79
	≤ 160	107.38	17.47	37.50	38.21	18.13	0.06	100.00	0.08
	≤ 200	93.10	77.80	70.00	369.15	19.90	8.83	90.00	87.83
6	≤ 40	4.75	0.33	75.00	1.27	27.00	0.00	100.00	0.00
	≤ 80	–$^{(10)}$	–	30.77	–	45.85	0.02	100.00	0.29
	≤ 120	–$^{(13)}$	–	33.33	–	156.60	16.50	86.67	170.29
	≤ 160	–$^{(5)}$	–	25.00	–	52.75	0.20	100.00	0.43
	≤ 200	–$^{(4)}$	–	40.00	–	43.10	0.93	80.00	2.95
9	≤ 40	1.00	0.03	100.00	0.08	29.00	0.00	100.00	0.00
	≤ 80	42.31	47.07	30.77	163.56	49.85	0.00	100.00	0.00
	≤ 120	–$^{(13)}$	–	13.33	–	66.87	0.01	100.00	0.08
	≤ 160	–$^{(3)}$	–	0.00	–	67.13	0.07	100.00	0.54
	≤ 200	–$^{(1)}$	–	10.00	–	58.10	0.06	100.00	0.20
12	≤ 40	1.00	0.04	100.00	0.09	29.00	0.00	100.00	0.01
	≤ 80	–$^{(11)}$	–	46.15	–	57.00	0.00	100.00	0.01
	≤ 120	–$^{(10)}$	–	13.33	–	77.20	0.00	100.00	0.02
	≤ 160	–$^{(3)}$	–	25.00	–	118.25	0.12	100.00	0.73
	≤ 200	–$^{(1)}$	–	10.00	–	168.80	16.10	90.00	160.87
15	≤ 40	1.00	0.04	100.00	0.12	29.00	0.00	100.00	0.00
	≤ 80	–$^{(11)}$	–	61.54	–	59.15	0.00	100.00	0.01
	≤ 120	–$^{(9)}$	–	6.67	–	82.53	0.00	100.00	0.02
	≤ 160	–$^{(5)}$	–	0.00	–	117.88	0.02	100.00	0.14
	≤ 200	–$^{(0)}$	–	0.00	–	100.90	0.01	100.00	0.09

of the branch-and-price algorithm. Notice that all values are average values over all test instances in the specific range, except for the maximum computation time, which corresponds to a single problem instance.

We see that, in a number of cases, the number of branching nodes is equal to zero. For the branch-and-price algorithm, this means that the solution found by the heuristic equals $\lceil \frac{n}{B} \rceil$ (this happened 217 out of 250 times in Table 2.5, 12 out of 250 times in Table 2.6, and 45 out of 250 times in Table 2.7). For the ENUM algorithm it means that the lower and upper

bounds computed at the start of the algorithm are the same, which means that there exists an optimal solution with value $\lceil \frac{n}{B} \rceil$ (this happened 90 out of 250 times in Table 2.5, and never in Table 2.7).

From Table 2.5 we conclude that the instances from data set 1 can be solved to optimality very quickly by both algorithms; 90% of the instances are solved in less than one second for the branch-and-price algorithm, and for ENUM, 99% of all instances are solved in less than one second. One reason for the good performance of the branch-and-price algorithm is that in 86.8% of the instances, the heuristic for finding an initial solution in the branch-and-price algorithm provides us with an optimal solution that equals $\lceil \frac{n}{B} \rceil$. Indeed, the quality of the lower bound $\lceil \frac{n}{B} \rceil$ for data set 1 is striking, as can be seen from Table 2.2. Another reason for the success of the branch-and-price algorithm is that the matrices are very sparse (for example, for $B = 15$, each column has fewer than 15 nonzeros). For the ENUM algorithm, an optimal solution is found without having to branch in 36.0% of the instances. Thus, in most cases ENUM has to be executed, and then it finds an optimal solution very quickly, i.e., usually faster than the branch-and-price algorithm. As described before, from the results it is also clear that both $\lceil \frac{n}{B} \rceil$ and LP are good lower bounds for the integer optimum; the value of AC, however, is in many cases far from the optimum. The small gap between the LP and IP solutions is not surprising, since this is the case for set-partitioning models in general (Byun, 2001).

When we look at the results from the random instances in Table 2.6, we see that the computation times of the branch-and-price algorithm are slower than those from the real-world instances. Furthermore, the lower bound from the LP relaxation and the value of AC are very close to the integer optimum; for large B ($B \geq 9$) they even coincide. Not surprisingly, the lower bound $\lceil \frac{n}{B} \rceil$ performs much worse here, especially for large B. The heuristic for finding an initial solution performs much worse compared to the results from the first data set: for only 4.8% of the instances does the heuristic find

an optimal solution equal to $\lceil \frac{n}{B} \rceil$.

Finally, when we look at Table 2.7, we see that the branch-and-price algorithm is no longer capable of solving the larger instances to optimality in a reasonable amount of time (we use a time limit of one hour to solve a single problem instance). If a number of instances in a specific range could not be solved within this time limit, this is denoted by "$\_^{(\alpha)}$" where α denotes the number of instances that could be solved by the branch-and-price algorithm. So we present only these figures from these groups of instances that could all be solved optimally. We see that, for the smaller instances, the branch-and-price algorithm is still very fast, but as the number of items increases, the number of instances that cannot be solved also increases. This is in sharp contrast to the performance of ENUM. From these results we see that ENUM is able to solve all instances, and 95.6% of all instances are solved within one second of computation time. Also for this data set, the value of the LP relaxation is a good lower bound for the integer optimum.

2.6 Conclusion

In this chapter we described two exact algorithms for a pallet-loading problem. The first algorithm is a branch-and-price algorithm, based on an integer-programming formulation. The pricing problem can be formulated as a longest-path problem and can be solved efficiently by dynamic programming. The second algorithm is an enumeration algorithm based on the concept of bounded clique width. This algorithm was motivated by a special structure that is present in the real-world instances that were used for computational experiments. From the computational results we conclude that the problem instances from data sets 1 and 2 can be solved satisfactorily by the branch-and-price algorithm, while approximately 30% of the instances from data set 3 cannot be solved after one hour of computation time. The enumeration algorithm performs really well in case of the real-

world instances (99% of the instances are solved within a second). Also, the instances from data set 3 can be solved efficiently by ENUM, due to the small clique width of these problem instances (95% of these instances are solved in less than one second), but the random instances cannot be solved efficiently, due to the large number of different lengths in the input. From the results we also see that the LP relaxation provides us with a good lower bound on the integer optimum.

Chapter 3

Partitioning a Weighted Partial Order

The problem of partitioning a partially ordered set into a minimum number of chains is a well-known problem in operations research. In this chapter we study a generalization of this problem, where we not only assume that the chains have bounded size (i.e., we are given an additional parameter B that denotes the maximum number of elements that can be contained in a chain; see Chapter 2), but also that a weight w_i is given for each element i in the partial order such that $w_i \leq w_j$ if $i \prec j$. The weight of a chain is defined as the weight of the heaviest element in the chain. The problem is then to partition the partial order into a number of chains of bounded size, such that the sum of the weights of the chains in minimized. We refer to this problem as Minimum Weight Partition into B-chains (MWPB). We prove that this problem is $\mathcal{APX}$-hard, and we propose and analyze lower bounds for this problem. Based on these lower bounds, we exhibit a 2-approximation algorithm, and show that it is tight. We report computational results for a number of real-world and randomly generated problem instances.

The outline of this chapter is as follows. We describe the problem in Section 3.1. Section 3.2 deals with the complexity of MWPB. In Section 3.3

we propose a number of lower bounds on the value of the optimum, and in Section 3.4 we present a 2-approximation algorithm for solving MWPB. We tested the algorithm on a number of real-world problem instances, as well as on randomly generated instances. The results from these experiments are described in Section 3.6. In Section 3.7 we discuss an interesting variant of MWPB, where the orientation of an element is taken into account. Finally, in Section 3.8, we conclude.

3.1 Introduction

Consider a partially ordered set $(X, \prec)$. We say that two elements $i, j \in X$ are *comparable* if either $i \prec j$ or $j \prec i$. A *chain* C is defined as a subset of X such that all elements $i, j \in C$ are pairwise comparable. An *antichain* A is a subset of X such that no two elements $i, j \in A$ are comparable. The *size* of a chain (or antichain) is equal to the number of elements contained in it (Trotter, 1992).

Now, the problem of partitioning a partially ordered set $(X, \prec)$ into a minimal number of chains such that each element of X belongs to at least one chain, is a well-known, fundamental problem in operations research. This problem is solvable in polynomial time, and the size of a maximum antichain is equal to the minimum number of chains needed to cover all elements of X (Dilworth, 1950).

Shum and Trotter (1996) generalize this problem by assuming that an integer B is given that bounds the size of a chain. Thus, in this setting no more than B elements can be in a chain. They show that the corresponding decision problem is $\mathcal{NP}$-complete, even for a fixed $B = 3$. As far as we are aware, this is the only work that considers the problem of partitioning a partial order into chains of bounded size.

In this chapter we further generalize this problem by assuming that a weight

w_i for each $i \in X$ is given such that $w_i \leq w_j$ if $i \prec j$. Moreover, we define the weight of a chain C as $\max_{i \in C} w_i$, thus, the weight of a chain is equal to the weight of the heaviest element in the chain. Further, we denote a chain containing at most B elements as a B-chain. The problem is now to partition X into a minimum-weight set of B-chains. We refer to this problem as Minimum Weight Partition into B-chains (MWPB). Observe that when $w_i = 1$ for all $i \in X$ the problem dealt with by Shum and Trotter (1996) arises.

3.1.1 Relation to Graph-Coloring Problems

The problem of partitioning a partially ordered set into chains of bounded size is closely related to vertex coloring problems. In the standard vertex coloring problem we want to color the vertices of a graph such that for all edges, both endpoints have different colors. The objective is to minimize the number of colors needed (Golumbic, 1980). Now, given a positive integer k, a *bounded vertex coloring* of a graph $G = (V, E)$ is then defined as a usual vertex coloring in which each color is used at most k times. The bounded chromatic number of a graph G ($\gamma_k(G)$) is the smallest number of colors such that G admits a bounded coloring with color classes of size at most k. (Hansen et al., 1993).

The problem of partitioning a partial order into chains of bounded size can be formulated as a bounded vertex coloring problem. Indeed, we can represent the partial order as a graph, creating a vertex for each element in the partial order, and connecting two vertices if the corresponding elements are incomparable (the resulting graph is a cocomparability graph, see Golumbic (1980)). The solution to the bounded vertex coloring problem on such problem instances can be transformed to solutions to the original problem of partitioning a partial order: vertices having the same color make up a chain. Since any color can be used at most a limited number of times, we know that the chains are bounded in size as well.

Hansen et al. (1993) show that the bounded vertex coloring problem for arbitrary graphs is $\mathcal{NP}$-complete for $k \geq 3$ using a reduction from Partition into Isomorphic Subgraphs. They also conjecture that, if k is part of the input, it is $\mathcal{NP}$-complete to find γ_k for bipartite graphs. This conjecture is later proved by Bodlaender and Jansen (1993). Jarvis and Zhou (2001) show that the bounded vertex coloring problem is solvable in polynomial time for trees.

3.1.2 Applications

Applications of MWPB can be found in the field of mutual exclusion scheduling (Baker and Coffman, 1996; Jansen, 2003), also known as batch scheduling with job compatibilities (Boudhar, 2003; Finke et al., 2004)). In such a problem jobs are given, each with a given processing time p_i, and for each pair of jobs it is known whether they can be processed on the same machine. A machine can process at most B jobs simultaneously, and the time a machine needs to process its jobs equals the maximum processing time of the jobs assigned to that machine. The problem is then to assign the jobs to the machines, respecting the compatibilities, while minimizing the sum of the completion times of all machines. To represent the job compatibilities, often a graph is used; different types of graphs lead to different complexity results. In our setting, the graph corresponding to the job compatibilities is a comparability graph (see e.g. Golumbic (1980)).

Another application of MWPB from the field of pallet loading is described in Chapter 2. In this setting, the weight of an item equals the area of the item, and the weight of a chain is determined by the area of its largest item. The objective is then to minimize total area, that is, the sum of the weights of all chains.

3.1.3 Our Results

In this chapter we show the following:

- Strengthening a result from Shum and Trotter (1996), we show that MWPB is $\mathcal{APX}$-hard, rendering the existence of a PTAS unlikely.
- We propose two lower bounds, each of which can be arbitrarily bad when compared to the value of the optimum. The maximum of these lower bounds, however, is shown never to be less than half the optimum value.
- We describe a simple algorithm that yields a solution with a value guaranteed not to exceed twice the optimum value. The analysis is shown to be tight.
- We consider an extension of the special case of MWPB where the dimension is 2, to a setting where rotation of the elements is allowed.

3.2 Complexity of MWPB

The decision problem corresponding to MWPB can be formulated as follows:

Given an integer B, a partial order $(X, \prec)$, weights w_i with $w_i \leq w_j$ if $i \prec j$ $(\forall i, j \in X)$, and an integer K, does there exist a partition of X into B-chains such that the sum of the weights of the B-chains does not exceed K?

As stated in Section 3.1, Shum and Trotter (1996) prove that this decision problem is $\mathcal{NP}$-complete, even if $w_i = 1$, for all $i \in X$. We can strengthen their result:

Theorem 4. *MWPB is $\mathcal{APX}$-hard, even if*

- *$B = 3$, and*

- $w_i = 1 \ \forall i$, *and*
- *each element occurs in no more than* 3 *chains.*

Proof: We use a reduction from the Maximum 3-bounded 3-dimensional matching problem (3DM-3), and follow the reduction from Shum and Trotter (1996). Furthermore, we apply arguments used in Chekuri and Khanna (2005).

Problem 3DM-3 is defined as follows: given are three sets X, Y, Z with $|X| = |Y| = |Z| = n$, and a set of triples $T \subseteq X \times Y \times Z$ with $|T| = m$. Each element occurs at most three times in a triple in T (therefore we can assume that $m = O(n)$). The goal is to find a matching of largest cardinality. Kann (1991) showed that this problem is $\mathcal{APX}$-hard. In fact, he shows that it is $\mathcal{NP}$-hard to decide whether there exists a matching of size n, or whether every matching has size at most $(1 - \delta)n$ for some fixed $\delta > 0$.

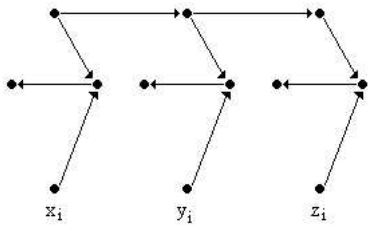

Figure 3.1: Subgraph for triple $t_i = \{x_i, y_i, z_i\}$, see Shum and Trotter (1996).

Now, consider an instance $\mathcal{I}$ of problem 3DM-3, and build the corresponding instance $\mathcal{I}'$ of problem MWPB with $B = 3$ as described in Shum and Trotter (1996) (See also Garey and Johnson (1979), page 69): to each triple $t_i \in T$, $t_i = \{x_i, y_i, z_i\}$, we associate a subgraph (called a *configuration*) as shown in Figure 3.1, where an arc from i to j implies that $i \prec j$. Observe that

i) each chain in the instance $\mathcal{I}'$ must be contained within a single configuration, and

ii) only if x_i and y_i and z_i are covered by other chains, it is possible to use 3 chains (recall that $B = 3$) for the points in the configuration; otherwise at least 4 chains are needed.

If instance $\mathcal{I}$ has a matching of size n, then there exists a solution to instance $\mathcal{I}'$ with $4n + 3(m - n) = n + 3m$ chains: we need 4 chains to cover each element in a configuration that corresponds to a chosen triple in the matching. For the elements in the remaining configurations, we need 3 chains to cover them, since all x_i, y_j, and z_k are covered by other chains.

Now let us consider the case that the maximum matching has size $(1 - \delta)n$ for some fixed $\delta > 0$. First of all, we need $4(1 - \delta)n$ chains to cover the elements contained in configurations corresponding to triples in the matching. Then we have covered $3(1-\delta)n = 3n-3\delta n$ x, y, z-elements. So there are $3\delta n$ x, y, z-elements remaining. Observe that there can be no configuration that contains more than 2 of these $3\delta n$ elements, since otherwise a better solution (i.e., a matching exceeding size $(1 - \delta)n$) exists. That means that the minimum number of configurations needed to cover these elements is $\frac{3\delta n}{2}$, and we need at least $4(\frac{3\delta n}{2})$ chains to cover all elements in these configurations. Finally, for the remaining elements we need at least 3 times the number of remaining configurations, which equals $3(m - (1 - \delta)n - \frac{3\delta n}{2})$ chains. So in total, we need at least $4(1-\delta)n+4(\frac{3\delta n}{2})+3(m-(1-\delta)n-\frac{3\delta n}{2}) = n+3m+\frac{1}{2}\delta n$ chains. Since $m = O(n)$, the $\mathcal{APX}$-hardness follows. □

3.3 Lower Bounds for MWPB

Consider an instance of MWPB, containing an integer B and n elements, each with a weight w_i, $1 \leq i \leq n$. We assume that the elements are ordered such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Let OPT denote the value of an optimal solution to the instance. We define three lower bounds lb_i, $i = 1, 2, 3$, as follows:

1. $lb_1 = w_1 + w_{B+1} + \ldots + w_{(\lceil \frac{n}{B} \rceil - 1)B+1}$. Since the size of a chain cannot

exceed B, lb_1 is obviously a lower bound for OPT.

2. When we omit the size constraint (i.e., if there is no restriction on the size of a chain), a relaxation of MWPB appears. Solving this relaxation gives a minimum-weight set of chains with value MWC. We set $lb_2 = MWC$.

3. $lb_3 = \max(lb_1, lb_2)$.

Theorem 5. *We can calculate the value of lb_2 by solving a min-cost flow problem.*

Proof: In order to compute the value of lb_2, we create a directed graph $D = (V, A)$. V contains $2n + 2$ nodes: 2 nodes i' and i'' for every $i \in X$, a source s and a sink t. We draw an arc from s to each node i', with cost 0. Then we add an arc from each node i'' to t with cost w_i. Next, we add an arc from a node i' to its copy i'' with cost 0, and we add arcs from nodes i'' to j' if $i \prec j$, also with cost 0. Finally we add an arc from s to t with cost 0. All nodes have supply zero, except for s which has supply n, and t, which has supply $-n$ (a demand of n). All arc capacities are equal to 1, except for the arc from s to t, which has capacity equal to n. The arcs from a node i' to its copy i'' have a lower bound on the flow of 1. Now, a min-cost flow in D can be easily translated to a solution to MWPB without the size constraint and vice versa. □

Notice that this algorithm solves a weighted generalization of the classical result of Dilworth (1950).

Example – Min-Cost Flow

Suppose we are given a partial order S containing four points, $S = \{a, b, c, d\}$, and suppose $a \prec b$, $a \prec d$ and $c \prec d$. The min-cost flow network corresponding to this example is shown in Figure 3.2. (The lower and upper bounds on the arcs, the arc costs and the demands are omitted from this figure.)

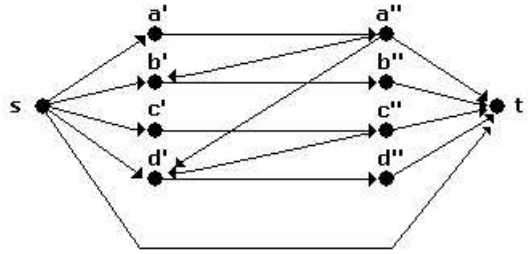

Figure 3.2: Min-Cost Flow Network.

If we solve this min-cost flow problem, we find a solution containing a number of paths that have positive flow value. In the solution to our original problem, we put two points in the same chain if, in the solution to the min-cost flow problem, these points appear on the same path with positive flow value. This corresponds to a minimum-weight set of chains, keeping in mind that we have disregarded the size requirement on the chains. ■

We now show that lb_1 and lb_2 can be arbitrarily bad, even in the unweighted case. Consider lb_1, and suppose we are given a problem instance with $B = n$, such that no two elements are comparable, and suppose that each element has weight 1. One easily verifies that OPT equals n, while lb_1 equals 1.

Next, consider lb_2, and suppose we have a problem instance with $B = 1$, such that all elements are comparable, and that each element has weight 1. Again, OPT equals n, while lb_2 gives a value of 1. So, we cannot give a constant performance guarantee for either of these lower bounds. However, no instance exists where both lower bounds are arbitrarily bad. Indeed, let us now consider the maximum of these two lower bounds, lb_3.

Claim 3. *For each instance of MWPB: $lb_3 \geq \frac{1}{2}OPT$. Moreover, this bound is tight.*

We postpone the proof of this inequality to Theorem 6; we first give an instance for which this bound is tight.

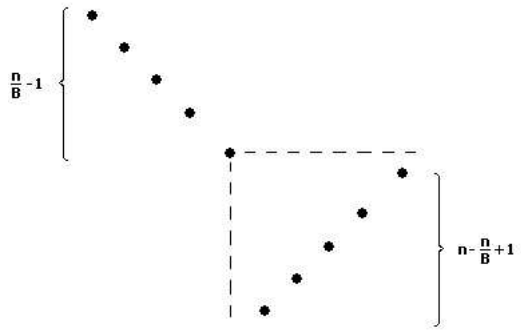

Figure 3.3: A Tight Example.

Consider the problem instance given in Figure 3.3, with $n = B^2$. So, we have B^2 elements, all with weight 1, such that $n - \frac{n}{B} + 1$ of these elements are pairwise comparable, and the remaining $\frac{n}{B} - 1$ elements are pairwise incomparable. Observe that, for such instances, $lb_3 = max(lb_1, lb_2) = B$, while $OPT = (\frac{n}{B} - 1) + \lceil (\frac{n - \frac{n}{B} + 1}{B}) \rceil = (B - 1) + \lceil \frac{B^2 - B + 1}{B} \rceil = (2B - 1)$.

3.4 A 2-approximation Algorithm for MWPB

In this section we propose a 2-approximation algorithm for MWPB, and show that it is tight.

Consider the following heuristic H:

Step 1. Omit the size constraint, and find a minimum-weight set of chains as described in Theorem 5.

Step 2. For each chain consisting of say K elements $i_1, \ldots, i_K$, with $i_1 \succ i_2 \succ \ldots \succ i_K$, partition it into $\lceil \frac{K}{B} \rceil$ B-chains such that elements $i_{(j-1)B+1}, i_{(j-1)B+2}, \ldots, i_{\min(jB,K)}$ form B-chain j, $j = 1, \ldots, \lceil \frac{K}{B} \rceil$.

Theorem 6. *H is a 2-approximation algorithm for problem MWPB.*

Proof: Recall that we assume that the elements are ordered such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Suppose we find a solution using heuristic H with value v_H, where in the first step we find a decomposition into p chains, $C_1, ..., C_p$. In the second step we partition each of these p chains into a number of B-chains. The maximal element of C_ℓ is referred to as i_ℓ, $1 \leq \ell \leq p$. All other maximal elements of B-chains are referred to as j_ℓ, $1 \leq \ell \leq k$. Assume, without loss of generality, that $j_1 \geq j_2 \geq \ldots \geq j_k$. Notice that we can associate to each item j_ℓ a set of B items that belong to the same chain found in Step 1 as j_ℓ, and are the smallest B items that dominate j_ℓ (an item u dominates another item v if $v \prec u$). Let us refer to this set of items as $S(j_\ell)$, $1 \leq \ell \leq k$ (notice that $j_\ell \notin S(j_\ell)$).

Claim 4. $j_\ell \geq \ell B$

Argument: Consider the sets $S(j_\ell)$, $\ell = 1, \ldots, k$. Since these sets are pairwise disjoint, the number of items that must precede j_ℓ, $1 \leq \ell \leq k$, equals at least ℓB. □

The inequality from Claim 4 implies $\sum_{\ell=1}^{k} w_{j_\ell} \leq \sum_{\ell=1}^{k} w_{\ell B} \leq lb_1$. And, obviously, $\sum_{\ell=1}^{p} w_{i_\ell} = lb_2$. Thus, we have that $v_H = \sum_{\ell=1}^{p} w_{i_\ell} + \sum_{\ell=1}^{k} w_{j_\ell} \leq lb_1 + lb_2 \leq 2OPT$, which proves Theorem 6. Also, notice that $lb_3 + lb_3 \geq lb_1 + lb_2 \geq v_H \geq OPT$, implying Claim 3. □

It is not clear yet whether the bound derived is tight. Indeed, the example for which lb_3 is shown to be worst possible is solved to optimality by H. We know that a solution found by heuristic H can be no worse than twice the optimal value. Can we find problem instances for which this gap is tight?

For the problem instances shown in Figure 3.4, we have $n = B^2$ elements, all with weight 1, and there are B elements that are pairwise incomparable, and the remaining $B(B-1)$ elements are pairwise comparable. For such an instance, we have $OPT = lb_1 = lb_2 = lb_3 = B$, while the heuristic H could give a solution with value $(B-1) + \lceil \frac{B^2-B+1}{B} \rceil = (2B-1)$, so these

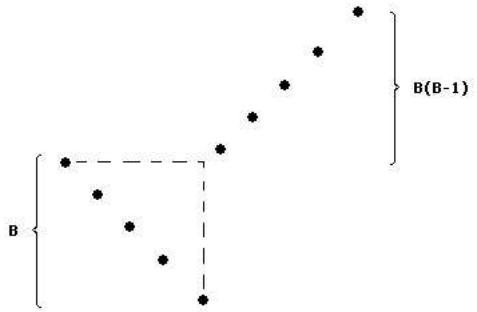

Figure 3.4: Worst-case instance with respect to H.

are asymptotic worst-case instances with respect to H.

3.5 Integrality Gap

In the last section we established a 2-approximation algorithm for partitioning a weighted partial order into chains of bounded size. An interesting question is whether we can do better than this. A natural approach is then to look at the integrality gap between the solution to the LP-relaxation corresponding to some IP model, and the optimal integral solution. This gap serves as a lower bound for any approximation algorithm that is based on straightforward rounding of the solution of the linear program. In this section we analyze the gap between the LP-relaxation corresponding to a straightforward set-partitioning model and the integer optimum. The set-partitioning model is similar to the model we used in the branch-and-price algorithm in Section 2.2.1 of Chapter 2.

We define a decision variable x_k for every B-chain k:

$$x_k = \begin{cases} 1 & \text{if } B\text{-chain } k \text{ is in the solution} \\ 0 & \text{otherwise.} \end{cases}$$

We define λ_k as the weight of a B-chain k. The set-partitioning model is then as follows:

$$\min \sum_k \lambda_k x_k \tag{3.1}$$

$$\text{s.t.} \sum_{k:i\in\mathcal{I}_k} x_k = 1 \qquad \forall i \tag{3.2}$$

$$x_k \in \{0,1\} \qquad \forall k \tag{3.3}$$

Here, $\mathcal{I}_k$ is the set of all elements contained in B-chain k. The objective (3.1) is to minimize sum of the weights of the B-chains. Constraints (3.2) state that each element has to be in exactly one B-chain, and constraints (3.3) are the zero-one constraints on the x_k variables. For the LP-relaxation, we replace constraints (3.3) by constraints (3.4):

$$x_k \geq 0 \qquad \forall k \tag{3.4}$$

Let us start with a small example:

Example – Integrality Gap

Suppose we are given a problem instance containing three elements α, β, and γ, each with a weight equal to one, such that all three elements are pairwise comparable ($\alpha \prec \beta \prec \gamma$), and B is equal to 2. In the solution to the LP-relaxation we find three different B-chains: one B-chain containing elements α and β, a second B-chain containing elements α and γ, and a third B-chain containing elements β and γ. All three B-chains occur in the LP-solution with value $\frac{1}{2}$, so the value of the solution to the LP-relaxation v_{LP} is $\frac{3}{2}$. However, the optimal integral solution has value equal to 2 for this problem instance: an example of such a solution has a B-chain containing elements α and β and a second B-chain containing the single element γ. This means that, for this simple problem instance, the gap between the LP-solution and the integer optimum is $\frac{4}{3}$. ■

This example shows that, even for a very small problem instance, the integrality gap is already equal to $\frac{4}{3}$. Now, we can expand the example as follows: assume we have a problem instance containing $B+1$ elements, each with a weight equal to one, such that all elements are pairwise comparable. In the solution to the LP-relaxation there will be a variable with positive value for each B-chain containing exactly B elements. There are exactly $B+1$ of these B-chains, and each of them occurs in the LP-solution with value $\frac{1}{B}$. This means that the solution to the LP-relaxation has value $\frac{B+1}{B}$, while the integer optimum has value equal to 2. For large B, this leads to an integrality gap of a factor 2, thereby showing that a straightforward rounding approach will not improve the heuristic H with respect to the performance guarantee.

3.6 Computational Results

We implemented the 2-approximation algorithm in C++, using the CPLEX network solver to solve the min-cost flow problems, and we tested it on a number of real-world and randomly generated problem instances. We use the same 3 data sets that we used for the experiments in Chapter 2: the first data set contains 50 real-world instances provided to us by Bruynzeel Storage Systems, the second data set contains 50 randomly generated instances, and the third data set contains 50 randomly generated instances that have small clique-width. The problem instances for all three data sets contain between 20 and 200 elements (see Chapter 2). We solve each problem instance for 5 different values of B, so we have 250 experiments for each data set. (In the real-world setting of Bruynzeel Storage System, B equals 12.) Since all computation times were less than 0.01 seconds, for all instances, we omit them from the tables with results.

In Table 3.1 we compare the lower bounds. As lb_3 is defined as the maximum of lb_1 and lb_2, we want to know how many times lb_3 equals lb_1, and

Table 3.1: Results for lower bounds

B	*Data set 1*		*Data set 2*		*Data set 3*	
	$lb_3 = lb_1$	$lb_3 = lb_2$	$lb_3 = lb_1$	$lb_3 = lb_2$	$lb_3 = lb_1$	$lb_3 = lb_2$
3	100%	0%	96%	4%	96%	4%
6	100%	0%	74%	26%	74%	26%
9	98%	2%	50%	50%	28%	72%
12	94%	6%	42%	58%	14%	86%
15	94%	6%	26%	74%	8%	92%

how many times it equals lb_2. So in Table 3.1 we give, for each of the three data sets, the percentage of the number of times that lb_3 equals lb_1 and the number of times that lb_3 equals lb_2.

Tables 3.2, 3.3, and 3.4 show a comparison between the values of the three lower bounds and the value of the 2-approximation algorithm. In the first column we give the value of B, and in the next four columns we show the average values of the three lower bounds (lb_1, lb_2, and lb_3) and the 2-approximation algorithm (v_H). Of course, as can be seen in the third column, the value of B does not influence lb_2. Finally, the column labelled Δ_3 shows the average (avg_{Δ_3}) and the maximum (max_{Δ_3}) difference between the values of v_H and lb_3. These differences are all given in percentages (i.e., $\frac{v_H - lb_3}{lb_3} \cdot 100$).

From these results we see that the performance of the lower bounds is very different for the different data sets. For the first data set, that contains the real-world problem instances, lb_1 is clearly better than lb_2: in 97.20% of all experiments, the value of lb_1 is larger than the value of lb_2. If we look at the second data set, we see that lb_1 still performs better compared to lb_2, but the percentage of experiments for which lb_1 is larger than lb_2 is only 57.60% for data set 2. However, for data set 3 we see that lb_2 performs slightly better than lb_1: in 56.00% of all experiments the value of lb_2 is larger than the value of lb_1. The differences concerning the quality of the lower bounds with respect to the different data sets can be explained by the fact that

the real-world problem instances from data set 1 have a special structure, that is not present in the problem instances of the other data sets (see also Chapter 2, Section 2.5.2). As a consequence of this special structure, we need very few chains to cover all elements in case B is large, which means that the quality of lb_2 is very poor for these instances.

Table 3.2: Results for data set 1: real-world instances

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	1085.06	96.02	1085.06	1096.12	1.45	5.49
6	556.12	96.02	556.12	574.74	4.18	17.39
9	379.96	96.02	380.22	396.66	6.18	27.55
12	293.76	96.02	294.82	316.08	8.12	32.05
15	240.40	96.02	242.18	259.64	8.89	25.16

Table 3.3: Results for data set 2: random instances

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	6518.64	1508.44	6589.06	6935.18	11.12	31.73
6	3317.18	1508.44	3338.78	3877.36	19.22	39.22
9	2259.08	1508.44	2314.28	2867.24	18.71	37.17
12	1726.68	1508.44	1818.90	2348.64	18.45	43.90
15	1411.90	1508.44	1585.92	2066.94	17.49	46.88

Table 3.4: Results for data set 3: instances with small clique-width

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	3215.70	1357.98	3219.66	3607.64	12.76	29.47
6	1686.28	1357.98	1765.54	2200.00	23.61	38.59
9	1177.96	1357.98	1451.42	1762.92	20.28	43.54
12	926.42	1357.98	1389.50	1574.78	13.72	37.79
15	775.38	1357.98	1364.50	1470.34	9.16	39.91

Next we compare the values of lb_3 with the values of the approximation algorithm. The difference between the value of the approximation algorithm

and the value of lb_3 could, in theory, get as large as 100%, however, the maximum difference among all experiments from the three data sets is equal to 32.05% for data set 1, 46.88% for data set 2, and 43.54% for data set 3. The average difference for the three data sets equal 5.76% for data set 1, 17.00% for data set 2, and 15.91% for data set 3. So again we see that the results for data set 1 are clearly better compared to the results of data sets 2 and 3, which can be explained by the special structure that is present in the problem instances of data set 1.

Table 3.5: Comparison with optimal solutions: data set 1

B	OPT	v_H	*avg* Δ *(%)*	*max* Δ *(%)*	*% equal*
3	31.60	31.90	1.84	12.50	72.00
6	16.04	16.54	4.42	25.00	54.00
9	10.88	11.56	9.61	66.67	38.00
12	8.38	8.86	6.99	33.33	54.00
15	6.80	7.18	6.98	33.33	62.00

Table 3.6: Comparison with optimal solutions: data set 2

B	OPT	v_H	*avg* Δ *(%)*	*max* Δ *(%)*	*% equal*
3	33.98	39.20	17.17	33.33	2.00
6	18.38	23.64	23.94	44.44	20.00
9	16.68	18.10	6.66	43.75	46.00
12	16.04	16.22	0.82	13.04	86.00
15	16.04	16.04	0.00	0.00	100.00

Table 3.7: Comparison with optimal solutions: data set 3

B	OPT	v_H	*avg* Δ *(%)*	*max* Δ *(%)*	*% equal*
3	36.58	39.10	7.94	16.00	2.00
6	18.94	21.68	15.05	36.36	8.00
9	13.52	16.02	18.18	42.86	20.00
12	11.28	13.30	16.56	50.00	28.00
15	10.26	11.40	11.11	42.86	44.00

In order to gain more insight in the quality of the 2-approximation algo-

rithm, we want to compare the results of the approximation algorithm with the optimal values. However, since we don't have the optimal solutions to the problem instances used in the experiments, we do the following: we create new instances by setting the weight of each element of our problem instances to 1, and we solve the resulting instances using heuristic H. We compare the results with the results obtained using the exact algorithms described in Chapter 2. Tables 3.5, 3.6, and 3.7 show these results. In the first column we give the value of B, the next two columns give the average values of the optimal solution (OPT) and the approximation algorithm (v_H). Then we give the average difference between these two values (in percentages), the maximum difference (in percentages), and the number of problem instances for which the two values are equal. Table 3.5 shows the results for data set 1, Table 3.6 for data set 2, and finally Table 3.7 shows the results for data set 3.

From these results we see that, in most cases, the solutions found by the approximation algorithm are very close to the optimal values. For data set 1, the approximation algorithm gives the optimal solution in 56.00% of all problem instances. For data set 2 this happened in 50.80% of all instances, and for data set 3 this happened in 20.40% of all instances. While the maximum difference between the solution of the approximation algorithm and the optimum for the instances we consider is as large as 66.67%, the average difference between these two values is equal to ca. 6% for data set 1, ca. 10% for data set 2, and ca. 14% for data set 3. These results suggest that the approximation algorithm performs satisfactory for problem instances where the weight of each element is equal to one.

3.7 Rotation Problem

In some applications, for example in the field of pallet loading problems as discussed in Chapter 2, the items are allowed to be *rotated* (i.e., the length and the width of an item are swapped) if that allows us to find a bet-

ter solution. It is not difficult to exhibit instances where allowing rotation improves the solution value. So a relevant problem is, when rotation is allowed, to choose the best orientation for each item. Of course this question is motivated by a two-dimensional representation of the items. However, each partial order can be embedded in d-dimensional space for some d (Ore, 1962), and, therefore the rotation variant of MWPB is interesting in higher dimensions as well. In this variant the objective is still to partition X into a minimum-weight set of B-chains. However, now we are allowed to choose an orientation for each element of X. We refer to this problem as MWPB-R.

Theorem 7. *There exists an optimal solution to MWPB-R such that, for all points p that are contained in this optimal solution, it holds that*

$$x_p^1 \geq x_p^2 \geq \ldots \geq x_p^d \tag{3.5}$$

where d is the number of dimensions, and x_p^i is the i-th coordinate of point p.

Informally said, Theorem 7 states that there is no loss in rotating each item such that for each item the i-th largest coordinate becomes its size in the i-th dimension. Before we give a proof of this theorem, we give an example for the 2-dimensional case.

Example – The 2-dimensional case

Given are n pairs of points $\{(x_i^1, x_i^2), (x_i^2, x_i^1)\}$ $(i = 1, \ldots, n)$ in the 2-dimensional plane. Exactly one point from each of these n pairs must be present in a solution. Suppose we have an optimal solution that contains the chain $C = \{P, Q, R\}$ (see Figure 3.5). Theorem 7 states that there exists an optimal solution such that for all the selected points in this solution we have that $x_i^1 \geq x_i^2$. That means that we can exchange all the points that lie above the line $x^1 = x^2$ with their corresponding copies that lie below the line $x^1 = x^2$. This corresponds to the chain $C' = \{P, Q', R\}$. To see why

this is true, consider the following:

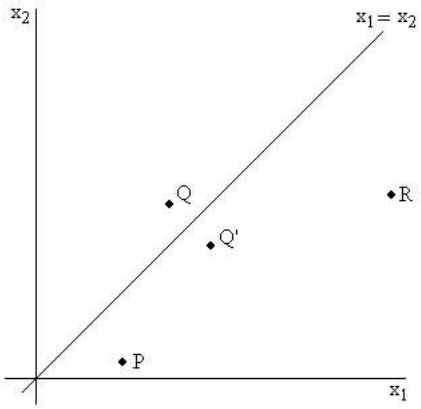

Figure 3.5: Example for the 2-dimensional case.

Since the chain containing points P, Q, and R is in the original solution, we know that $P \prec Q \prec R$, and so we have:

$$x_P^1 \leq x_Q^1 \leq x_R^1$$
$$x_P^2 \leq x_Q^2 \leq x_R^2$$

We also know that points P and R lie below the line $x^1 = x^2$, and point Q lies above this line, so we also have:

$$x_P^1 \geq x_P^2$$
$$x_Q^1 \leq x_Q^2$$
$$x_R^1 \geq x_R^2$$

Now, in order for C' to be feasible, we must have that $P \prec Q' \prec R$, so we must show that

$$x_P^1 \leq x_Q^2 \leq x_R^1 \tag{3.6}$$

$$x_P^2 \leq x_Q^1 \leq x_R^2 \tag{3.7}$$

Equation (3.6) is true since $x_P^1 \leq x_Q^1 \leq x_Q^2 \leq x_R^2 \leq x_R^1$. Equation (3.7) is true since $x_P^2 \leq x_P^1 \leq x_Q^1 \leq x_Q^2 \leq x_R^2$. ■

Now we continue with the proof of Theorem 7.

Proof: In order to prove Theorem 7, we must show that if we have a chain C containing points that do not satisfy condition (3.5), then the copies of these points that do satisfy condition (3.5) form a chain. So, given two arbitrary points U and V such that $U \prec V$, we must show that, for their copies satisfying condition (3.5), called U' and V', it holds that $U' \prec V'$. So we have to prove that if $U \prec V$, then $U' \prec V'$. That means that we have to show, for each $i = 1, \ldots, d$, that $x_{U'}^i \leq x_{V'}^i$.

Take the smallest i for which this does not hold, so we have $x_{V'}^i < x_{U'}^i$. This means that the i^{th}-largest coordinate of point V' is smaller than the i^{th}-largest coordinate of point U', which contradicts $U \prec V$. □

Observe that in the argument above B plays no role. Hence, using Theorem 5, we can solve instances of MWPB-R with $B \geq n$ to optimality in polynomial time.

Notice that rotating the items as described by (3.5) may increase the number of pairs of items that are comparable. It would be interesting to see how this would influence the performance of lb_2 and the heuristic H on the problem instances of Section 3.6.

3.8 Conclusions

In this chapter we discuss the problem of partitioning a weighted partially ordered set into chains of bounded size. We proposed three lower bounds for this problem, and presented a 2-approximation algorithm for solving it. The approximation algorithm is tested on a number of real-world and randomly generated problem instances. From the results of the experiments we see that, although the value of heuristic H could be up to twice the value of lb_3, the largest difference between these values over all 750 experiments is 46.88%, and the average difference equals 12.89%. We conclude from these results that the approximation algorithm performs reasonably well.

Chapter 4

Connectivity Measures for Internet Topologies

It is a cliché to state that stability and robustness of the Internet are fundamental for securing today's efficient communication. Maintaining the speed and the reliability of Internet-based communication is a prime challenge for service providers, their clients, and other involved institutions. In this chapter, we study the structure and the connectivity of the Internet.

The topology of the Internet has initially been modelled as an undirected graph, where vertices correspond to so-called Autonomous Systems (ASs), and edges correspond to physical links between pairs of ASs. However, in order to capture the impact of routing policies, it has recently become apparent that one needs to classify the edges according to the existing economic relationships (customer-provider, peer-to-peer or siblings) between the ASs. This leads to a directed graph model in which traffic can be sent only along so-called valley-free paths. Four different algorithms have been proposed in the literature for inferring AS relationships using publicly available data from routing tables. We investigate the differences in the graph models produced by these algorithms, focussing on connectivity measures. To this aim, we compute the maximum number of vertex-disjoint valley-free

paths between ASs as well as the size of a minimum cut separating a pair of ASs. Although these problems are solvable in polynomial time for ordinary graphs, they are $\mathcal{NP}$-hard in our setting. We formulate the two problems as integer programs, and we propose a number of exact algorithms for solving them. For the problem of finding the maximum number of vertex-disjoint paths, we discuss two algorithms; the first one is a branch-and-price algorithm based on the IP formulation, and the second algorithm is a non LP based branch-and-bound algorithm. The minimum cut problem is solved using a branch-and-cut algorithm, based on the IP formulation of this problem. Using these algorithms, we obtain exact solutions for both problems in reasonable time. It turns out that there is a large gap in terms of the connectivity measures between the undirected and directed models. This finding supports our conclusion that economic relationships need to be taken into account when building a topology of the Internet.

In Section 4.1 we introduce the problem. Section 4.2 gives formal definitions of the concepts that we require in this chapter, and we discuss a primal-dual formulation of the problem. Section 4.3 deals with the computation of vertex-disjoint valid paths and in Section 4.4 we describe how we compute minimum cuts with respect to valid paths. Section 4.5 reviews the known 2-approximation algorithms for solving both problems. In Section 4.6, we present our experimental results concerning the number of vertex-disjoint valid paths and the sizes of minimum cuts in the four different models with inferred relationships and the undirected model. We discuss their implications and also the differences that we observe in the depth of the provider hierarchy in the different models. Statistics about directed cycles in the graphs are given and some examples where they can be used to detect misclassifications are shown. Finally, in Section 4.7 we summarize our results and main conclusions.

4.1 Introduction

In order to understand the potential vulnerability of Internet-based communication, we need to get an idea of the routes that are being used for sending traffic, of the routes that could be used for sending traffic, and how different ways of sending traffic vary with respect to their susceptibility to failing servers and/or failing connections.

A first step is then retrieving how traffic is being sent over the Internet. This, however, is already not so easy to find out (Chang et al., 2004). To explain this, let us view the Internet as a set of Autonomous Systems (ASs; an AS is a subnetwork under separate administrative control), which are connected by physical links. ASs exchange routing information using the Border Gateway Protocol (BGP); this is a protocol that governs the communication between a pair of ASs. More specifically, each AS uses a local routing policy that determines which routes are announced to which neighboring ASs. For commercial reasons, details about these local policies of individual ASs are not publicly available. Obviously, this makes it difficult to create an accurate model that can be used in the analysis of the robustness of the Internet.

The goal of this chapter is to contribute to the development of an accurate model for the Internet topology. We do this by comparing different methods that have been proposed in the literature to infer the topology of the Internet using observed traffic-data. The comparison focusses on two connectivity measures, namely the number of (disjoint) paths between a given pair of ASs, and the size of a minimum cut separating a pair of ASs. Let us proceed by describing the relevant issues in more detail.

Routing policies depend mostly on the economic relationships between ASs. They represent an important aspect of Internet structure. Huston (1999a, 1999b) discussed the main trends in the diversity of commercial agreements between ASs. We will refer to local policies governed by the BGP as BGP

routing policies, or BGP policies for short. The impact of economic relationships on the engineering level, more precisely on BGP policies, has been recognized as one of the reasons for BGP path inflation (i.e., the phenomenon that traffic uses paths that are much longer than necessary; see Gao and Wang (2002)) and one of the important factors in route convergence analysis (i.e., the fact that, when a previously valid path to a destination D becomes invalid, it takes a long time until the network has obtained a new valid path to D (Labovitz et al., 2001)). Thus, the previously adopted undirected model of the Internet, which ignores BGP policies, is only a crude approximation of reality and might produce a distorted picture of the routes used in practice. On the other hand, incorporating all of the peculiarities of the manifold contracts between ASs in a new model would add too much complexity (assuming one would know these contracts). Therefore, a coarse classification of AS relationships into three categories—customer-provider, peer-to-peer and siblings—has been proposed (Gao, 2001). More recent work has restricted attention to customer-provider and peer-to-peer relationships only (Subramanian et al., 2002).

If ASs A and B are in a customer-provider relationship, i.e., if A is a customer of B, then B announces all its routes to A, but A announces to B only its own routes and routes of its own customers. If they are peers, they exchange their own routes and routes of their customers, but not routes of their providers or other peers. If ASs A and B are siblings, then A announces all its routes to B and B announces all its routes to A. These policies arise because customers do not want to act as transit ASs for their providers, i.e., a provider cannot route traffic through a customer to a different provider of that customer. As a consequence, only *valley-free* paths are valid, i.e., paths that first go "up" in the hierarchy and then "down" towards the destination. (A formal definition will be given in Section 4.2. In this chapter we will use the terms "valley-free path" and "valid path" interchangeably.)

Thus, one arrives at the following model. The Internet is a graph con-

taining the ASs as vertices. The graph can have directed and undirected edges. There are three different ways in which two vertices A and B can be connected:

i) an undirected edge between A and B. This is interpreted as "A and B are peers."

ii) a directed edge from A to B, and a directed edge from B to A. This is interpreted as "A and B are siblings."

iii) a directed edge from A to B, and no directed edge from B to A. This is interpreted as "A is customer of B."

Two ASs with at least one physical link between them are connected by a single edge (or a single pair of edges, in the case of siblings) in this model, no matter how many physical links there are between these two ASs. For comparison, note that the previously adopted *undirected graph model* of the Internet consisted of an undirected graph with an undirected edge between two ASs if there is at least one physical link between them.

Since information about the economic relationships between ASs is not publicly available (such information is often treated like a business secret), it is not so easy to determine the relationships that ASs have with each other. This problem of inferring AS relationship is known as the *Type of Relationship* (or *ToR*) problem: given an undirected graph $G = (V, E)$, and a set of paths P, label each edge as customer-provider, peer-to-peer or sibling, in such a way that the number of valid paths in G is maximized. The resulting graph is called a ToR graph (see Section 4.2 for formal definitions).

Four algorithms have been proposed for inferring AS relationships from BGP routing table information (Gao, 2001; Subramanian et al., 2002; Di Battista et al., 2003; Erlebach et al., 2002). However, it is not known how the topologies produced by these algorithms differ from each other. Also, it is unknown

whether the directed model of the Internet is indeed better than the previously adopted undirected model. Therefore, in view of the large impact of BGP policies in the Internet, we perform a thorough comparison of these graph models in this chapter. Since the main effect of BGP policies is that they restrict the set of paths that traffic can take in the network, we consider mainly path-related criteria. In particular, we compute the maximum number of vertex-disjoint valley-free paths between two ASs and the minimum number of vertices that must be removed from the graph so that no valley-free path between these two ASs remains. These are natural adaptations of classical measures of connectivity in graphs to the valley-free path model. It is well known that in the standard graph models (in the standard model, a path consists of a sequence of forward arcs in the directed case and of a sequence of undirected edges in the undirected case) the maximum number of disjoint paths between s and t is equal to the size of a minimum s-t cut (provided that s and t are not adjacent); moreover, the corresponding solutions can be computed efficiently (see Ahuja et al. (1993)). In a ToR graph this is not the case. It is $\mathcal{NP}$-hard to compute the maximum number of vertex-disjoint s-t paths; it is also $\mathcal{NP}$-hard to compute the minimum size of a valid s-t cut. The best known approximation ratio is 2 for both problems. Also, the minimum size of an s-t cut can be up to twice the maximum number of vertex-disjoint s-t paths. Thus, the max-flow min-cut equality holds only approximately for ToR graphs (Erlebach et al., 2005). We are able to obtain optimal solutions with a moderate amount of computation time using exact approaches, one of which involves applying a branch-and-price algorithm. We compare the results from the exact methods with those of the 2-approximation algorithms from Erlebach et al. (2005). We also compute disjoint paths and minimum cuts in the undirected Internet graph and compare the results with those of the different directed models.

Furthermore, we investigate directed customer-provider cycles, which are a somewhat unexpected structure, in the directed models. We claim that these cycles can help to detect misclassified relationships and thus improve

the accuracy of the Internet topology. We also give statistics about the minimum and maximum length of the observed cycles. Finally, we report statistics concerning the number of ASs that are connected by directed paths in the different directed models, a quantity that is related to the depth of the provider hierarchy and to the *customer-preference* aspect of current inter-domain routing (Feigenbaum et al., 2002). The latter means that paths through customers are preferred over paths through peers, and these to paths through providers.

In summary, our investigations address the following research questions:

- Do the differences between the undirected graph model and the directed graph models with respect to connectivity properties confirm the importance of incorporating BGP policies in the model?
- How do the directed graph models produced by the four algorithms proposed by Gao (2001), Subramanian et al. (2002), Di Battista et al. (2003), and Erlebach et al. (2002) compare to each other? Here, we are mainly interested in comparing connectivity measures and the depth of the provider hierarchy.
- How many directed customer-provider cycles occur in the different directed graph models, and can they be helpful in the detection of misclassified edges?
- In a graph on the scale of the Internet (containing up to 11,000 vertices and 30,000 edges), is it feasible to compute exact solutions to the $\mathcal{NP}$-hard problems of finding a maximum number of vertex-disjoint valley-free paths between two ASs or the size of a minimum cut? Such computations could prove useful in future investigations of robustness issues of the Internet.
- How does the performance of the 2-approximation algorithms proposed by Erlebach et al. (2005) compare to the performance of the exact al-

gorithms, both in the quality of the solutions found and in the running times?

4.1.1 Related Work and Motivation

Our starting point for the interpretation of BGP policies is the work of Gao (2001) that addressed the problem of unavailable information about the exact relationships between ASs. A heuristic algorithm was proposed for inferring AS relationships from BGP routing tables. In addition, it was observed that a path between a pair of ASs follows a particular structure: no path contains more than one peer-to-peer relationship, and once a provider-customer or peer-to-peer relationship is encountered in the path, no customer-provider relationship can follow. If we ignore sibling relationships for the moment and imagine that providers are at a higher level than their customers and peers are at the same level, the valid paths are "only up," "only down," or "first up and then down." Valid paths can have only one "peak" (which can consist of a single AS or of two ASs connected by a peer-to-peer relationship) and they must not contain "valleys." Therefore, such paths are also called *valley-free* paths. We use the same characterization of valid paths in this chapter.

Further work trying to infer AS relationships is presented by Subramanian et al. (2002). They formalize the problem by posing it as the optimization problem of giving an orientation to the edges of an undirected AS graph with the objective of maximizing the number of paths in the given BGP tables that become valid for this orientation. They pose the complexity of this problem as an open question. They also give a heuristic algorithm that infers relationships by first ranking all ASs and then applying certain rules to decide about the relationships between pairs of ASs using the rank values. Independently obtained results from Di Battista et al. (2003) and Erlebach et al. (2002) resolve the open question of Subramanian et al. (2002) and prove this inference problem to be $\mathcal{NP}$-hard. Two heuristic algorithms for

calculating approximately optimal orientations with respect to the number of valid paths are also presented by Di Battista et al. (2003) and Erlebach et al. (2002), respectively.

Rimondini et al. (2004) compare the algorithms from Subramanian et al. (2002) and Di Battista et al. (2003) with respect to two measurements. First, the AS relationships that are found by a certain algorithm on data sets from different moments in time are considered (called the *stability* in the paper). Second, the AS relationships found by the two algorithms on the same data set are taken into account (this is referred to as *algorithm independence*). They conclude that both algorithms produce highly stable results, and that the AS relationships found by both algorithms are very similar. This leads the authors to the conclusion that the valley-free path approach leads to reliable results.

In another paper by Xia and Gao (2004), a comparison of the algorithms from Gao (2001) and Subramanian et al. (2002) is done. Also, in this chapter, a new algorithm for inferring AS relationships is proposed, which is also taken into account in the comparison. The authors evaluate the accuracy of the three algorithms using partial AS relationships obtained from BGP community attribute and IRR (Internet Routing Registry) databases. They conclude that the new algorithm proposed in the paper outperforms the algorithms from Gao (2001) and Subramanian et al. (2002).

In this chapter, we compare the AS relationships computed by all four algorithms proposed by Gao (2001), Subramanian et al. (2002), Di Battista et al. (2003), and Erlebach et al. (2002), and we try to identify important characteristics of the relationship classifications produced by these algorithms. The motivation for our investigations comes from several papers that showed the impact of BGP policies on important features of Internet routing such as path inflation and routing convergence (see Labovitz et al. (2001), and Tangmunarunkit et al. (2001a, 2001b, 2003). In addition, recent results

about measurements on the AS level of the Internet have shown that there is a need for a simple and accurate algorithm to infer relationships; see Spring et al. (2003) about path inflation in inter- and intra-domain routing, Akella et al. (2003a) about multi-homing (i.e., the phenomenon that customers tend to have more than one external link to different providers, in order to guarantee the reliability of their network), and Akella et al. (2003b) about scaling properties of the Internet regarding link congestion.

Teixeira et al. (2003) compute the number of vertex- and edge-disjoint paths for the undirected model of the Internet AS topology, as well as for the topology of one Internet Service Provider. They did not take routing policies into account. Here, we investigate how the number of vertex-disjoint paths and the size of a minimum cut can be computed exactly and in reasonable time for Internet graphs that are constrained by BGP policies. These results may be helpful for future research on more resilient and efficient inter-domain routing.

4.2 Problem Description

In order to formulate the problem, we first state some preliminaries in Section 4.2.1. Then, in Section 4.2.2 we give a mathematical formulation for the problems of finding the maximum number of vertex-disjoint paths and minimum cut sizes.

4.2.1 Preliminaries

Subramanian et al. (2002), Di Battista et al. (2003), and Erlebach et al. (2002) refer to the problem of inferring the AS relationships in the Internet as the *Type of Relationship (ToR) problem*. Following this terminology, we construct a graph $G = (V, E)$, called a *ToR graph*, as follows: the vertices of G are the ASs. As mentioned before, a directed edge from u to v, where $u, v \in V$, together with a directed edge from v to u means that u and v are

siblings. A directed edge from u to v means that u is a customer of v, and an undirected edge means that u and v are in a peer-to-peer relationship. In a ToR graph, a directed edge from u to v is denoted by (u, v), and an undirected edge between u and v by $\{u, v\}$.

We define a path $p = (v_1, v_2, \ldots, v_r)$ from v_1 to v_r in a ToR graph $G = (V, E)$ to be *valid* if it satisfies one of the two following conditions:

1. There exists some j, $1 \leq j \leq r$, such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq j-1$ and $(v_i, v_{i-1}) \in E$ for $j + 1 \leq i \leq r$.

2. There exists some j, $1 \leq j \leq r$, such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq j - 1$, $\{v_j, v_{j+1}\} \in E$, and $(v_i, v_{i-1}) \in E$ for $j + 2 \leq i \leq r$.

Otherwise, a path is called *invalid*. This definition of valid paths captures the notion of "valley-free" paths arising from BGP routing policies. From now on, whenever we talk about paths in a ToR graph, we refer to valid paths. A path from s to t is also called an *s-t path*. Note that the reverse of an s-t path is a t-s path, hence the direction of a valid path is not important. Two s-t paths are called *vertex-disjoint* if they do not share any vertices except s and t.

Let $p = (v_1, v_2, ..., v_r)$ be a valid path from s to t. We can divide p into a forward part and a backward part at some node v_j, such that $(v_i, v_{i+1}) \in E$, $i = 1, 2, \ldots, j - 1$ (we know by definition that such a j exists; if j is not unique, we simply choose the maximal value for j). If p contains only directed edges, we say that a node v_l is on the forward part of p if $l < j$, v_l is on the backward part of p if $l > j$ and v_l is the node where p changes direction if $l = j$. If p contains an undirected edge, we say that a node v_l is on the forward part of p if $l \leq j$ and v_l is on the backward part if $l > j$.

Let $G = (V, E)$ be a ToR graph. For two non-adjacent vertices s and t in G, a *minimum valid s-t cut* in G is a set of vertices $C \subseteq V \setminus \{s, t\}$ of minimum

cardinality such that there is no valid s-t path in the ToR graph $G \setminus C$ (i.e., in the graph that is obtained from G by deleting the vertices in C and their incident edges). Note that a minimum valid s-t cut is a smallest set of ASs whose failure disconnects s and t if only valid paths are allowed.

A *directed cycle* $v = (v_1, v_2, \ldots, v_r)$, $r > 2$, in a ToR graph $G = (V, E)$ is defined in the usual sense, i.e., the vertices $v_1, v_2, \ldots, v_r$ are distinct and we have $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, r-1$ and $(v_r, v_1) \in E$.

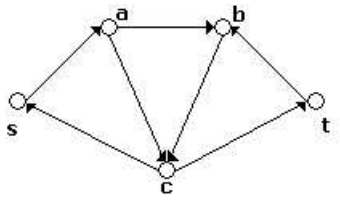

Figure 4.1: Gap between number of disjoint paths and minimum cut size.

As mentioned before, the maximum number of vertex-disjoint paths can be strictly less than the number of nodes in a minimum cut; we give an example from Erlebach et al. (2005) to illustrate this. In Figure 4.1 we see that the maximum number of vertex-disjoint paths is equal to 1, while the size of a minimum cut equals 2. Indeed, one can verify that the set of valid s-t paths equals $\{(s, a, b, t), (s, a, b, c, t), (s, a, c, t), (s, a, c, b, t), (s, c, b, t)\}$, and thus the maximum number of vertex-disjoint paths is equal to 1. Furthermore, one can easily verify that a minimum cut has at least size 2, since after removing one of the nodes a, b or c, there is still a valid path connecting s and t.

4.2.2 Problem Formulation

Let us now give two integer programming formulations; the first formulation (denoted by P) models the problem of finding a maximum number of vertex-disjoint paths between s and t, the second formulation (denoted by

D) models the problem of finding a minimum-sized set of nodes such that each path between s and t contains at least one node from this set.

Let $G = (V, E)$ be a ToR graph and s, t two distinct vertices of G. Assume that there is no edge between s and t (otherwise, we remove this edge, compute the maximum number of vertex-disjoint s-t paths, and add one to the result). Denote by $\mathcal{P}$ the set of all valid s-t paths in G, and let $\mathcal{V}_p$ be the set of all vertices contained in path $p \in \mathcal{P}$, except for s and t. Further, we define a decision variable x_p for each valid path p, as follows:

$$x_p = \begin{cases} 1 & \text{if valid path } p \text{ is in the solution} \\ 0 & \text{otherwise.} \end{cases}$$

Using a set-packing formulation, we get the following integer programming formulation:

$$\text{(P)} \quad \max \sum_{p \in \mathcal{P}} x_p \tag{4.1}$$

$$s.t. \sum_{p: v \in \mathcal{V}_p} x_p \leq 1 \qquad \forall v \in V \setminus \{s, t\} \tag{4.2}$$

$$x_p \in \{0, 1\} \qquad \forall p \in \mathcal{P} \tag{4.3}$$

The objective (4.1) is to maximize the number of paths between s and t. Constraints (4.2) state that each vertex (except for s and t) can belong to at most one path, and constraints (4.3) are the zero-one constraints on the x_p variables.

The second formulation has a variable y_v for every $v \in V \setminus \{s, t\}$:

$$y_v = \begin{cases} 1 & \text{if vertex } v \text{ is in the } s\text{-}t \text{ cut} \\ 0 & \text{otherwise.} \end{cases}$$

The second formulation can now be given as follows:

$$\text{(D)} \quad \min \sum_{v \in V \setminus \{s,t\}} y_v \tag{4.4}$$

$$s.t. \sum_{v \in \mathcal{V}_p} y_v \geq 1 \qquad \forall p \in \mathcal{P} \tag{4.5}$$

$$y_v \in \{0,1\} \qquad \forall v \in V \setminus \{s,t\} \tag{4.6}$$

A property of formulations (P) and (D) is that the LP-relaxation of (P) and the LP-relaxation of (D) constitute a primal-dual pair of linear programs. Further, notice that formulation (P) has exponentially many variables (since the number of valid s-t paths can be exponential in the number of vertices); equivalently, formulation (D) has exponentially many constraints.

4.3 Vertex-Disjoint Paths in ToR Graphs

In this section we present two exact algorithms for solving problem P; i.e., for finding the maximum number of vertex-disjoint paths in ToR graphs. The first one is a branch-and-price algorithm based on the integer programming formulation (4.1)-(4.3) (Section 4.3.1), and the second algorithm is a branch-and-bound method in which a max-flow computation has to be performed in each node of the search tree (Section 4.3.2).

4.3.1 A Branch-and-Price Algorithm

Branch-and-price is a technique for solving integer programs with a huge number of variables. We refer to Barnhart et al. (1998) or Vanderbeck and Wolsey (1996) for a thorough description of this technique. Here we apply it to solving instances of formulation (P). There are (at least) two important issues to be considered when developing a branch-and-price algorithm: (i) how to solve the pricing problem (this enables us to conclude that either we have solved the LP-relaxation of (P), or we have identified a new variable (column) to be added to the restricted master; (ii) how to branch. We need to develop a partition of the solution space in such a way that the efficient solvability of the pricing problem is preserved.

Column Generation

We start by generating a feasible solution (consisting of a set of vertex-disjoint paths) as follows. We apply a simple breadth-first search to find a valid path between s and t, we add this path to the solution, and remove all nodes in this path (except s and t) from our graph. Then a new iteration starts, and we repeat the breadth-first search until no more valid paths can be found. The resulting set of paths found by this *iterated breadth-first search* is denoted by $\mathcal{P}'$, and its value (number of disjoint paths) is referred to as $V_{\mathcal{P}'}$. We consider the restriction of the LP-relaxation of (P) to the variables x_p for $p \in \mathcal{P}'$ (the *restricted master problem*). We solve the restricted master using an LP-solver and obtain a solution to the restricted primal program and its corresponding dual. Let us call the dual solution y^*. Now, we need to check whether y^* is also a feasible solution to the dual program that includes constraints for all paths $p \in \mathcal{P}$. In other words, we need to check whether there exists a valid s-t path p in the graph such that $\sum_{v \in p} y^*_v < 1$. This problem is known as the *pricing problem* (Vanderbeck and Wolsey, 1996). We can solve the pricing problem in polynomial time, thereby implying that the LP-relaxation of formulation (P) (as well as the LP-relaxation of (D)) can be solved in polynomial time.

Claim 5. *The LP-relaxation of* (P) *can be solved in polynomial time.*

Proof. We prove the claim by showing that the pricing problem can be reduced to a shortest path problem. The result then follows from the "separation = optimization" result (Grötschel et al., 1988).

Consider the so-called 2-layer graph that has been proposed by Erlebach et al. (2005) (we first assume that there are no undirected edges in G): two copies G_1 and G_2 of graph G are created, but in G_2 all edge-directions are reversed. Then, so called "vertical edges" are added, i.e., directed edges from the vertices in G_1 to their copies in G_2. Finally, s in G_1 is identified with its copy in G_2 and all edges that end in s are removed, and t in G_1 is identified with its copy in G_2 and all edges that start in t are removed. In

this way, all valid s-t paths in G correspond to directed paths from s to t in the 2-layer model: If a valid s-t path in G first uses some forward edges and then some backward edges, its forward part in the 2-layer model lies in G_1, then it switches to the second layer using a vertical edge, and then it goes again forward to t in G_2 because of the inverted edge-directions. (See Figure 4.2 for an illustration.)

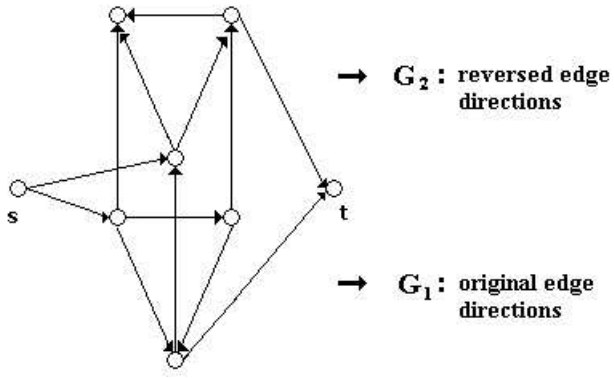

Figure 4.2: The 2-layer graph of the ToR graph depicted in Figure 4.1.

> **Remark.** Notice that we can also convert a path p' from the 2-layer graph to a valid path p in the original graph as follows: first we replace all nodes in p' by their original copies from the original graph, and we delete all identical successive nodes (resulting from the use of a vertical edge). Then we delete all cycles in the resulting path, which gives us a valid path p in the original graph.

We can deal with undirected edges in the following way. For an undirected edge $\{a, b\}$, we do not add corresponding edges to G_1 or G_2, but instead add directed edges (a_1, b_2) and (b_1, a_2) to the 2-layer model, where a_1, a_2 (b_1, b_2) are the copies of a (b) in G_1 and G_2, respectively. This ensures that valid s-t paths in G that include an undirected edge also have a corresponding path in the 2-layer model.

Next, we define the edge weights of the 2-layer graph as follows: edges entering a copy of vertex v get weight y_v^*, except for vertical edges, which get weight 0. Observe that a shortest directed path from s to t in the 2-layer model gives us a valid s-t path in G that minimizes the sum of the y_v^* values of the vertices on the path. Since the shortest path problem can be solved in polynomial time (using for instance Dijkstra's algorithm), we can solve the pricing problem in polynomial time. □

If the solution of the pricing problem produces a valid s-t path p such that the sum of y_v^* values on this path is less than 1, we add path p to the restricted master and repeat the procedure. If there is no such path, we are done and have obtained an optimal solution to the LP-relaxation of (P). If the obtained solution is fractional, i.e., contains variables whose values are strictly between 0 and 1, we use a branching strategy in order to arrive at an integral solution.

Branching

If the optimal solution to the linear programming relaxation is fractional, a natural approach is to try different ways of fixing these variables to integers and solving the problem recursively for each of these alternatives (branching). Here it is important to preserve the form of the pricing problem and its efficient solvability in the branching procedure. We achieve this as follows.

Given a feasible, optimal solution x^* to the LP-relaxation of (P), we call a vertex *fractional* if it has at least three incident edges that lie on different valid paths with value $x_p^* > 0$. Notice that if a solution is fractional, it has at least one fractional vertex. (If we have a fractional solution, there must be at least one path which has a fractional value. Consider such a path d. Now, if no fractional vertex exists, each vertex has at most two incident edges that lie on valid paths with $x_p^* > 0$. However, this means that d can have value equal to 1 without violating any constraints, resulting in an integral solution.) Our branching strategy is as follows: for a fractional vertex

w, we delete all edges incident to w except two that could lie consecutively on some valid path. Each possible way of doing this forms a branch. Thus, for instance, if w has k incoming edges and ℓ outgoing edges, the number of branches is $k\ell + \binom{k}{2}$. If there are many fractional vertices, we choose one for branching that has a maximum number of incident edges lying on fractional paths.

In this way we exclude the current fractional solution, but do not exclude any integral solution, and the problem structure is preserved: in each branch, we solve a problem of the same type on a graph with fewer edges.

For each branch, if the value of the fractional solution is not larger than the value of the best integral solution found so far, we do not enter that branch. Otherwise, we explore all branches in a depth-first traversal. In this way we are sure to arrive at an optimal integral solution to (P).

We remark that our approach can be adapted easily to a version of the problem where each vertex v has an integral capacity c_v and we allow up to c_v valid paths passing through it. (Here, valid paths could occur more than once in the solution.) To solve this version of the problem, we simply replace each vertex v by c_v copies and then apply our algorithm as described above.

The branch-and-price algorithm for a given a ToR graph G and two distinct vertices s and t is summarized by the pseudo-code given below.

Branch-and-Price Algorithm VertexDisjointPaths

---

1. Calculate an initial solution consisting of a set of paths $\mathcal{P}'$ with value $V_{\mathcal{P}'}$ using the iterated breadth-first search, and let $V^* = V_{\mathcal{P}'}$. Create a list L and add to L a branching node corresponding to the input graph G.

2. $L = \varnothing$?

   YES: STOP. An optimal solution is found with value V^*.

   NO: Select the next branching node G from L (i.e., the branching node that was added most recently to L), remove it from L, calculate a set $\mathcal{P}'$ of edge-disjoint s-t paths in G using iterated breadth-first search, and continue with step 3.

3. Solve the LP-relaxation using only those variables that correspond to a path in $\mathcal{P}'$.

4. Solve the pricing problem. Is there a variable (a path) with negative reduced costs?

   YES: Add this variable to $\mathcal{P}'$ and go to step 3.

   NO: An optimal solution to the LP-relaxation is found with value V_{LP}. Continue with step 5.

5. $V_{LP} > V^*$?

   YES: Continue with step 6.

   NO: Go to step 2.

6. Is the solution to the LP-relaxation integral?

   YES: $V^* = V_{LP}$. Go to step 2.

   NO: Select a fractional vertex v. For each possible way of deleting all edges incident to v except for two edges that could lie consecutively on some valid path, create a new branching node (i.e., the graph obtained by deleting the respective edges) and add it to L. Go to step 2.

Valid Inequalities

In order to strengthen the LP-relaxation, a natural strategy is to add valid inequalities. In this section we will discuss a class of inequalities that is valid for formulation (P) of the vertex-disjoint paths problem. We will refer to these inequalities as *triangle inequalities*.

As the name suggests, we consider triangles in the ToR graphs. We define a triangle as a subset of three vertices which are connected with customer-provider edges in such a way that they do not form a directed cycle. For example, if there are three vertices a, b and c, and there is an edge from a to b, an edge from a to c and a third edge from b to c, this is a triangle. For each such triangle $t = (a, b, c)$, we define $T_{abc} = \{p \in \mathcal{P} |\ p \text{ contains } \{a, b\} \text{ or } \{a, c\} \text{ or } \{b, c\}\}$.

Now, for every triangle t in a ToR graph the following inequality states that the sum of all valid paths using one of the three edges in t must be smaller than or equal to one:

$$\sum_{p:p \in T_{abc}} x_p \leq 1 \qquad \forall \text{ triangles } (a, b, c) \in V^3 \tag{4.7}$$

It is clear that inequalities (4.7) are valid for (P). One can view inequalities (4.7) (as well as inequalities (4.2)) as a manifestation of clique-inequalities for the node packing problem. Indeed, when we build a graph in which there is a node for every path $p \in \mathcal{P}$, and where two nodes are connected if and only if the two corresponding paths share a vertex in the ToR graph, it is obvious that the node packing problem on this graph is exactly problem P. Notice that inequalities (4.2) and (4.7) need not constitute all clique inequalities in the node packing graph. In Section 4.6.2 we report shortly on the computational effectiveness of these inequalities.

4.3.2 A Branch-and-Bound Algorithm

Our second algorithm for solving the vertex-disjoint paths problem is a non LP-based branch-and-bound algorithm, in which we use the same 2-layer graph representation as explained in Section 4.3.1.

We start with an initial solution, computed by the iterated breadth-first search discussed in Section 4.3.1. The value of this solution is a lower bound on the integer optimum. Next, we compute a maximum flow in the 2-layer graph, where we assign a capacity of 1 to each vertex. The flow we find is not necessarily vertex-disjoint (since it may happen that the maximum flow found uses a node in G_1 and its copy in G_2), so it is an upper bound on the optimal solution. We first check whether the flow found by the maximum flow procedure is vertex-disjoint, or equal to the lower bound, in which case we have found the integer optimum. Otherwise, we have to branch, which we do as follows:

In every node in the search tree, we select a vertex v from the original graph that is used more than once in the flow found by the maximum flow procedure. This vertex v has a copy v_1 in the first layer, and a copy v_2 in the second layer of the 2-layer graph. Now, we generate two new branches as follows:

In the first branch, we delete vertex v_1, and all its adjacent edges, from the 2-layer graph. In the second branch, we delete all incoming edges of v_2, except for the vertical edge (v_1, v_2), from the 2-layer graph. Next, we perform a maximum flow calculation in each branching node, and repeat this procedure until we have found the integer optimum. The correctness of the branching step follows from the observation that if a node occurs in a path of the solution, it is either on the backward part of the path, which is permitted in the first branch, or it is on the forward path or it is the node where the path changes direction (see Section 4.2.1), which is permitted in

the second branch.

The branch-and-bound algorithm is summarized by the pseudo-code given below. In our implementation, we actually compute min-cost maximum flows (all edges are assigned a cost of one) instead of standard maximum flows, as we expect that maximum flows using a minimum total number of edges can reduce the number of branching nodes required.

Branch-and-Bound Algorithm VertexDisjointPaths

---

1. Calculate an initial solution consisting of a set of paths $\mathcal{P}'$ with value $V_{\mathcal{P}'}$ using iterated breadth-first search, and let $V^* = V_{\mathcal{P}'}$. Create a list L and let $L = \varnothing$.

2. Construct the 2-layer graph H, and add to L a branching node corresponding to H.

3. $L = \varnothing$?

   YES: STOP. An optimal solution is found with value V^*.

   NO: Select the next node H from L (i.e., the branching node that was added most recently to L), remove this node from L and continue with step 4.

4. Calculate a maximum flow MF with value V_{MF} in the 2-layer graph H.

5. $V_{MF} > V^*$?

   YES: Continue with step 6.

   NO: Go to step 3.

6. Does the maximum flow MF in H correspond to vertex-disjoint paths in G?

   YES: $V^* = V_{MF}$. Go to step 3.

NO: Select a vertex v that is used more than once in MF. Create two new branching nodes as follows:

i. Delete copy v_1 of v from the 2-layer graph.

ii. Delete all incoming edges of copy v_2 of v from the 2-layer graph, except for the edge (v_1, v_2).

Add the branching node corresponding to each of these two branches to L. Go to step 3.

4.4 Minimum Cuts in ToR Graphs

We solve the minimum cut problem using the dual (D) presented in Section 4.2.2. Since (D) might have exponentially many constraints, we first compute a set $\mathcal{P}'$ of vertex-disjoint s-t paths using the iterated breadth-first search as described in Section 4.3.1 and start by solving the LP-relaxation of (D) using only the constraints for paths in $\mathcal{P}'$. Solving this small linear program gives us a solution y^*. Then we check whether there is a valid path such that $\sum_{v \in p} y^*_v < 1$, again using a shortest-path algorithm in the 2-layer model of the graph (i.e., we solve the separation problem with respect to constraints (4.5)). If such a path is found, we add the corresponding constraint to our linear program (D) and repeat the procedure until no more valid paths with $\sum_{v \in p} y^*_v < 1$ can be found. The resulting solution y is an optimal solution to the LP-relaxation of (D). In case the resulting solution y is fractional, we branch.

The branching is more straightforward than for the vertex-disjoint paths problem. If there is a vertex v such that $0 < y_v < 1$, we add a constraint $y_v = 0$ (an *exclusion* constraint) in one branch and $y_v = 1$ (an *inclusion* constraint) in the other branch to the linear program and solve it again, thus having two branches for a fractional vertex. If there are many fractional vertices, we simply branch on the first one that we find. Similarly to the previous case, we do not enter a branch where the optimal fractional

solution is at least as large as the smallest integral solution found so far. The branch-and-cut algorithm is summarized by the pseudo-code given below.

Branch-and-Cut Algorithm MinCut

---

1. Let $V^* = \infty$. Create a list L and add to L a branching node corresponding to an empty set of inclusion/exclusion constraints.

2. $L = \varnothing$?

   YES: STOP. An optimal solution is found with value V^*.

   NO: Select the next node C from L (i.e., the branching node that was added most recently to L), remove this node from L, calculate a set of vertex-disjoint s-t paths $\mathcal{P}'$ using iterated breadth-first search, and continue with step 3.

3. Solve the LP-relaxation using only the constraints that correspond to a path in $\mathcal{P}'$ and the inclusion/exclusion constraints from C.

4. Solve the separation problem. Is there a variable (a path) with negative reduced costs?

   YES: Add this variable to $\mathcal{P}'$ and go to step 3.

   NO: An optimal solution to the LP-relaxation is found with value V_{LP}. Continue with step 5.

5. $V_{LP} < V^*$?

   YES: Continue with step 6.

   NO: Go to step 2.

6. Is the solution to the LP-relaxation integral?

   YES: $V^* = V_{LP}$. Go to step 2.

   NO: Select a vertex v such that y_v is fractional. Create two new branching nodes as follows:

- In the first node, add the exclusion constraint $y_v = 0$ to C.
- In the second node, add the inclusion constraint $y_v = 1$ to C.

Add these two nodes to L.

Remark. We remark that the same approach can be used to solve the generalization of the problem where each vertex v has a weight w_v and the objective is to find a valid s-t cut of minimum weight. The only difference is that the objective function becomes $\min \sum_{v \in V \setminus \{s,t\}} w_v y_v$.

Notice that we use two different approaches for solving the linear programming relaxations of formulations (P) and (D). We found that there were no significant differences in running-time between these two approaches.

4.5 Approximation Algorithms

In this section we discuss two 2-approximation algorithms for finding the maximum number of vertex-disjoint paths and the size of minimum cuts. Both algorithms are presented by Erlebach et al. (2005). In order to make the presentation self-contained, we repeat the description of these algorithms in this section. Section 4.5.1 deals with the algorithm for the problem of finding the maximum number of vertex-disjoint paths, and in Section 4.5.2 we give the algorithm for calculating the size of a minimum cut.

4.5.1 Vertex-Disjoint Paths

Before stating the approximation algorithm, we need some definitions. If the forward part of a path p_1 intersects a backward part of a path p_2 at a node v, we speak of a crossing at v. The two paths p_1 and p_2 can be recombined at the crossing to form a new path, consisting of the first part of p_1 and the last part of p_2. Given a graph $G = (V, E)$ and two vertices s

and t, the algorithm is as follows.

2-Approximation algorithm DisjointPaths (Erlebach et al., 2005)

---

1. Construct the 2-layer graph, and calculate a maximum flow in this graph.
2. Add, for each path in this maximum flow, the corresponding path in the original graph G to $\mathcal{P}'$.
3. Define $\mathcal{F}$ as the set of all forward parts of paths in $\mathcal{P}'$, and $\mathcal{B}$ as the set of all backward parts.
4. Label all forward parts and all crossings as unscanned. Recombine the forward and backward parts as follows:
   (a) Select an unscanned forward part p_f from $\mathcal{F}$ that has at least one unscanned crossing.
   (b) Select the first unscanned crossing c on p_f, and let p_b in $\mathcal{B}$ correspond to a backward part containing c.
   (c) Recombine p_f and p_b at c. Label p_f and all previous crossings on p_b as scanned. If p_b was already recombined with some other forward part p'_f, mark p'_f as unscanned.
   (d) Are there any unscanned forward parts with unscanned crossings left?

   YES: Go to step 4a.

   NO: STOP. A solution is found that is vertex-disjoint.

4.5.2 Minimum Cut Sizes

We now give the approximation algorithm for finding the minimum cut between two vertices s and t. Assume again we have a ToR graph $G = (V, E)$ and two vertices $s, t \in V$. We also assume there is no direct edge in G

between s and t, since a s-t cut does not exist in that case. The algorithm is then as follows:

2-Approximation algorithm MinCut (Erlebach et al., 2005)

---

1. Construct the 2-layer graph, and calculate a minimum cut in this graph.
2. From the cut found in step 1, construct a cut $\mathcal{C}$ in G as follows: $\mathcal{C}$ contains all vertices $v \in V$ for which at least one copy is in the cut found in step 1.
3. STOP. $\mathcal{C}$ is a cut in G containing at most twice the number of nodes as in a minimum cut.

4.6 Computational Experiments

In this section we first give a description of the data we used for our experiments (Section 4.6.1). Next, in Section 4.6.2, we discuss some issues concerning the implementation of the algorithms, and finally we present our results. In Section 4.6.3 we give computational results and discuss the performance of the different algorithms. The algorithms described in Sections 4.3, 4.4 and 4.5 are executed on a number of different ToR graphs, and we compare these results with those in the undirected model (where routing policies that are consequences of established economic relationships are not included) in order to quantify what the differences are with respect to the size of a minimum cut and the number of disjoint paths. Finally, in Section 4.6.4, we focus on the interpretation of the results.

4.6.1 Description of the Data

We use BGP tables from five different dates (April 2001, February 2002, April 2002, January 2003 and February 2004), available from the University

of Oregon Route Views project web-site (OREGON), to construct undirected graphs and four types of ToR graphs. This means that we have five different graphs for each of the five points in time, giving five undirected graphs and 20 ToR graphs in total. The undirected graphs are obtained by creating an undirected edge between two ASs if they appear consecutively in some path in the BGP tables. We also used one undirected graph model representing the Internet of April 1–16, 2002 that we obtained from CAIDA's Internet Topology Data Kit, ITDK0204 (CAIDA). We refer to this graph as the *CAIDA graph*, to the undirected graphs based on Oregon Route Views data as *undirected BGP graphs*, and to the graphs that include AS relationships as *ToR graphs*. The types of ToR graphs are denoted by A, B, C, and D as follows:

- ToR graphs of type A are obtained using the algorithm from Erlebach et al. (2002). They contain only customer-provider edges, no peer-to-peer or sibling edges.
- ToR graphs of type B are obtained using the algorithm from Di Battista et al. (2003) by running the software bgpSat publicly available from their web-page (BGPSAT). A majority of the edges are classified by bgpSat as customer-provider edges, but the classification of some edges is left undetermined. We classify the latter edges as peer-to-peer edges. Thus, type B graphs contain customer-provider edges and a few peer-to-peer edges.
- ToR graphs of type C are obtained from the web-page (CIMVP) and have been produced with the algorithm from Subramanian et al. (2002). The algorithm classifies edges as peer-to-peer edges, customer-provider edges, or unknown edges. We treat the unknown edges as sibling edges.
- ToR graphs of type D are obtained with the algorithm from Gao (2001) (using the implementation (LRIP)) and contain customer-provider edges, peer-to-peer edges, and sibling edges.

Table 4.1: Comparison of edge classifications.

ToR Graphs	*Percentages of identically classified edges*				
	18.04.2001	*04.02.2002*	*06.04.2002*	*09.01.2003*	*10.02.2004*
A vs. B	95.53	95.41	95.40	95.88	95.08
A vs. C	91.70	91.57	92.21	92.24	91.02
A vs. D	90.96	91.43	91.67	93.16	91.23
B vs. C	89.71	90.30	90.55	90.40	90.28
B vs. D	89.37	90.46	90.59	91.50	90.24
C vs. D	89.60	90.55	90.72	91.35	90.75

All of the inference algorithms that we have used for the construction of ToR graphs are heuristics. Thus, it is interesting to see how many edges between ASs are classified in the same way by the different algorithms. In Table 4.1 the percentages of identically classified edges are given for all six combinations of ToR graphs. For example, 95.53% of the edges are classified in the same way in A and B graphs from April 2001, as is shown in the first entry of the table. From this table we see that approximately 90% of all edges are classified the same.

Since computing the maximum number of vertex-disjoint paths and the minimum cut size for *all* pairs of ASs would have taken prohibitively long (even after pruning vertices of degree 1, the graphs still contain roughly 7,000 to 11,000 vertices), we confine our calculations to approximately 1000 pairs of ASs per graph. For this reason, we select 47 ASs as representatives and carry out the computations for all possible 1081 pairs of these ASs. We have selected the ASs by taking 47 vertices among the vertices of largest degree in the biggest R component of the undirected BGP graph of April 2002. (A partition of Internet graphs into P, Q, R and I components was proposed by Vukandinović et al. (2002). The biggest R component is the biggest connected component in the graph that is obtained after deleting all vertices of degree 1 and their neighbors.) All of the 47 selected ASs are vertices in that component that have at least 7 neighbors within that component. Their AS numbers and descriptions are given in Table 4.2. As one can see, the

ASs are geographically well spread—they are from Europe, USA, and Asia. Furthermore, there are representatives of bigger and smaller ISPs (Internet Service Providers), telecom nodes (e.g. Japanese and Belgian telecom), well-connected universities and research centers (e.g. University of Stanford, University of Oregon, and National Center for Supercomputing Applications), exchange points (e.g. London and Hongkong Internet Exchange), etc. This means that we have chosen well-connected ASs with diverse functionalities and good geographic coverage while avoiding the highest-degree nodes in the Internet (which are neighbors of leaves) as well as nodes with very small degree.

4.6.2 Implementation Issues

We have implemented the algorithms in C++ using CPLEX 9.0 to solve linear programs and the LEDA library to process graphs. Our experiments were done on a Sun Fire 480R workstation with two 900MHz processors (our code uses only one of them) and 4GB main memory.

For all computations we have removed vertices with degree 1, since they do not affect the number of disjoint paths or the cut sizes for any other pair of ASs. After pruning the leaf vertices, the graphs contain about 7,000 to 11,000 vertices and 20,000 to 30,000 edges.

For the computation of disjoint paths and cuts, we replaced each peer-to-peer edge $\{u, v\}$ by two edges (u, d) and (v, d), where d is a new dummy vertex. In this way the valley-free path policy is preserved, while the graph consists of directed edges only.

At the start of the branch-and-price algorithm for computing the maximum number of disjoint s-t paths, we do some additional preprocessing on the graphs. First, we delete all vertices (except s and t) for which the indegree

Table 4.2: The 47 selected ASs with AS numbers and short descriptions obtained from Internet registries.

AS nr.	Description	AS nr.	Description
32	Stanford University	5413	GX Networks
237	Merit Network Inc.(USA)	5459	LINX-AS,London Internet Exchange Ltd.
600	OARnet(USA)	6079	RCN Corporation (USA)
680	DFN-IP service G-WiN	6402	One Call Communications, Inc.(USA)
1136	KPN Telecom OVN IO	6774	BELBONE BELGACOM
1237	Korea Institute of Science and Technology Information	6830	UPC Distribution Services european broadband ISP services
2500	Japan Network Information Center WIDE Project	7091	ViaNet Communications (USA)
2514	NTT PC Communications, Inc., Japan	7623	HPCNET-AS High Performance Computing NETwork(HPCNET)Korea
2518	C&C Internet Service mesh(NEC Corporation), Japan	7660	APAN-JP Asia Pacific Advanced Network - Japan
2647	SITA France	7679	QTNET Kyushu Telecommunication Network Co.,Inc.
2687	IBM, NH USA	8426	CLARANET-AS ClaraNET UK AS of European ISP
2818	BBC Internet Services, UK	8553	AVENSYS Avensys Networks Ltd UK
3112	OARnet(USA)	9270	APAN-KR-AS Asia Pacific Adv. Neets Korea Consort. Net. Oper.Center
3304	KPNBELGIUM	9335	CIP-JAPAN-AS-AP ATT IPlus Asia and Pacific IP Network
3333	RIPE NCC Operations	9497	DIGITELONE Digital Telecommunications Philippines Inc.
3491	CAIS Internet(USA)	10099	HKUNICOM1-AP Voice over IP, ISP
3557	INTERNET SOFTWARE CONSORTIUM, INC. (USA)	10764	National Center for Supercomputing Applications
3582	University of Oregon	11854	Internap Network Services (USA)
3754	NYSERNet(USA)	12359	INTELIDEAS Intelideas Autonomous System Madrid, Spain
4197	ERX-GLOBALONLINE, Japan	12457	ONO-SERVICE-PROVIDER, Spain
4635	Hong Kong Internet Exchange–Route Server 1	13129	Global Access Telecommunications, Inc.
4725	ODN JAPAN TELECOM CO.,LTD.	13646	Cignal Global Communications, Inc.(USA)
5000	Internet Online Services (USA)	14390	Core Communications, Inc (USA)
5056	Iowa Network Services(USA)		

is equal to zero. These vertices can never belong to a valid path, so removing them will not affect the solution we find. Next, we check whether s and t belong to the same biconnected component of the underlying undirected graph. (A biconnected component of an undirected graph G is a subgraph of G such that we can remove any vertex of this subgraph without disconnecting it (Harary, 1969).) If so, we can run the algorithm on this component only (which is usually much smaller than the original graph), and in this way we still get the optimal solution. If s and t do not belong to the same biconnected component, the number of valid paths between s and t will be either 0 or 1. So we check whether there exists a valid path from s to t, in which case the number of vertex-disjoint paths is equal to one. If no valid s-t path exists, our solution is equal to zero. Finally, we found that adding the valid inequalities discussed in Section 4.3.1 actually slows down the branch-and-price algorithm. In fact, the number of branching nodes needed to solve the problem decreases, as expected, but the time needed to process a single node increases more heavily than the decrease in number of branching nodes, so the computational results presented next are those obtained without the additional valid inequalities.

For the minimum cut problem, we need to get a well-defined notion of minimum s-t cuts also for adjacent vertices. We handle such vertex pairs as follows: we remove the direct edge (or pair of edges, in the case of a sibling relationship between s and t) between the two vertices s and t, compute the size of a minimum s-t cut in the graph without that edge, and add 1 to the result. We do this in the undirected graphs as well as in the ToR graphs. Note that in the undirected model, the number of disjoint paths between two vertices is equal to the size of a minimum cut separating these two vertices. In ToR graphs, these values can differ.

4.6.3 Computational Results

Next, we are interested in the number of disjoint paths and the minimum cut size between pairs of ASs in the different graphs. First we discuss the performance of the two exact algorithms for solving the vertex-disjoint paths problem. Then we give results for the performance of the algorithm for solving the minimum cut problem, and finally we give results from the approximation algorithms.

Vertex-Disjoint Paths

We have tested both the branch-and-price and the branch-and-bound algorithm described in Section 4.3 to calculate the maximum number of vertex-disjoint paths for any pair of ASs. In Tables 4.3 and 4.4 we give the results of these computations.

Tables 4.3 and 4.4 give the computational results for the branch-and-price and branch-and-bound algorithms, respectively. The first two columns show the graph type and date. The third column contains the value of the integer optimum. The last four columns show the computation times (in seconds), the number of branching nodes needed to solve the problems, the percentage of problem instances that are solved in less than one second, and the percentage of instances that are solved in more than 10 seconds. All values in these tables are average values over the 1081 pairs of ASs, so they contain results of over 20,000 problem instances. While we could run the branch-and-price algorithm to completion on all pairs in all graphs, we had to terminate the branch-and-bound algorithm on a few pairs (at most 10 out of 1081 pairs in each of the graphs) after several hours of computation time. The running-time and the number of branching nodes shown for the branch-and-bound algorithm in Table 4.4 are thus the averages over the pairs for which the algorithm could be run to completion.

Table 4.3: Results for branch-and-price algorithm.

Type	*Date*	*OPT*	*Branch&Price*			
			Time	*#BN*	*%≤1*	*%>10*
A	18.04.2001	6.38	0.83	1.96	98.06	0.65
	04.02.2002	7.88	1.26	2.02	94.36	1.30
	06.04.2002	8.66	2.61	4.55	93.15	1.85
	09.01.2003	7.35	0.80	1.58	94.91	0.65
	10.02.2004	8.10	6.26	6.77	86.12	3.79
B	18.04.2001	6.42	3.34	7.59	91.77	3.15
	04.02.2002	8.49	7.61	11.21	81.41	5.46
	06.04.2002	9.39	10.69	10.94	78.08	7.77
	09.01.2003	7.52	2.82	5.06	86.40	3.61
	10.02.2004	8.44	12.12	10.90	79.19	5.64
C	18.04.2001	6.14	1.25	2.29	94.54	1.30
	04.02.2002	7.98	2.04	2.76	86.86	2.68
	06.04.2002	8.46	2.96	3.71	86.77	2.41
	09.01.2003	6.61	0.88	1.57	93.06	0.83
	10.02.2004	7.80	1.94	1.97	80.48	2.50
D	18.04.2001	6.34	2.63	3.06	83.63	4.26
	04.02.2002	8.01	2.20	2.42	85.38	3.70
	06.04.2002	8.69	5.96	4.31	81.41	3.98
	09.01.2003	7.30	1.80	2.20	88.53	2.41
	10.02.2004	7.92	8.36	4.16	73.64	4.81

From Tables 4.3 and 4.4 we conclude that, on the average, both algorithms perform well on the selected pairs of ASs. The running-times of the branch-and-bound algorithm are much more variable. On 71.78% of all instances, the branch-and-bound algorithm was faster than the branch-and-price algorithm. On the other hand, the branch-and-price algorithm could solve all instances in reasonable time (the average running-time over all instances is 3.92 seconds, while the instance with the longest running-time took slightly more than one hour), while the running-time of the branch-and-bound algorithm increased drastically for a few instances, thus leading to a larger average running-time on most graphs. The number of branching nodes needed to find the integer optimum is much larger for the branch-and-bound algorithm in comparison to the branch-and-price algorithm. For the branch-and-price algorithm, the average number of branching nodes is surprisingly small, since

Table 4.4: Results for branch-and-bound algorithm.

Type	*Date*	*OPT*	*Branch&Bound*			
			Time	*#BN*	*%≤1*	*%>10*
A	18.04.2001	6.38	9.44	34.42	94.26	1.11
	04.02.2002	7.88	2.63	10.84	88.99	2.59
	06.04.2002	8.66	4.40	16.25	87.60	3.61
	09.01.2003	7.35	2.28	5.31	94.63	1.85
	10.02.2004	8.10	15.40	32.86	86.96	4.44
B	18.04.2001	6.42	2.16	10.45	83.44	3.33
	04.02.2002	8.49	25.62	71.69	71.14	8.33
	06.04.2002	9.39	42.63	115.76	68.55	12.86
	09.01.2003	7.52	11.62	33.28	82.79	3.05
	10.02.2004	8.44	18.13	40.19	72.34	8.97
C	18.04.2001	6.14	3.81	11.02	86.12	4.53
	04.02.2002	7.98	18.50	43.59	69.29	6.94
	06.04.2002	8.46	4.28	10.68	69.47	3.33
	09.01.2003	6.61	1.73	4.38	84.55	2.59
	10.02.2004	7.80	70.27	133.88	71.14	10.73
D	18.04.2001	6.34	30.43	67.72	71.51	9.44
	04.02.2002	8.01	47.57	116.08	73.64	9.90
	06.04.2002	8.69	45.04	88.61	58.19	13.97
	09.01.2003	7.30	14.20	31.43	79.56	5.00
	10.02.2004	7.92	59.74	79.00	66.51	16.37

in about 89% of the problem instances the solution to the LP-relaxation is integral and we do not need to branch at all.

Minimum Cuts

The algorithm described in Section 4.4 to calculate the size of a minimum cut for a pair of ASs has also been executed on the ToR graphs. The results of these computations can be found in Table 4.5. The first two columns show the graph type and the date, the third column contains the optimal value of the minimum cuts, and finally we give the computation times (in seconds), the number of branching nodes needed to find the integer optimum, the percentage of problem instances that are solved in less than one second, and the percentage of instances that are solved in more than 10 seconds. Again,

all values are average values over all 1081 pairs of ASs for a specific graph type and date.

Table 4.5: Results for the minimum cut problem in ToR graphs.

Type	*Date*	*OPT*	*Time*	*#BN*	*%≤1*	*%>10*
A	18.04.2001	6.38	0.69	1.03	94.73	0.09
	04.02.2002	7.88	0.95	1.01	77.15	0.00
	06.04.2002	8.66	1.04	1.02	67.07	0.09
	09.01.2003	7.35	1.01	1.01	70.68	0.19
	10.02.2004	8.11	1.90	1.06	49.68	0.74
B	18.04.2001	6.43	0.82	1.11	89.18	0.46
	04.02.2002	8.52	1.86	1.33	59.85	2.41
	06.04.2002	9.42	1.85	1.16	47.92	2.22
	09.01.2003	7.53	1.23	1.03	60.13	0.09
	10.02.2004	8.44	1.83	1.05	41.81	1.11
C	18.04.2001	6.15	1.02	1.06	81.22	0.37
	04.02.2002	7.99	1.43	1.07	60.22	0.46
	06.04.2002	8.47	1.58	1.12	57.45	0.93
	09.01.2003	6.61	1.14	1.02	65.22	0.09
	10.02.2004	7.81	2.15	1.14	34.97	1.39
D	18.04.2001	6.35	1.26	1.17	71.42	0.74
	04.02.2002	8.02	1.34	1.05	63.74	0.28
	06.04.2002	8.70	3.04	1.32	49.58	1.02
	09.01.2003	7.30	1.22	1.01	50.32	0.09
	10.02.2004	7.93	2.08	1.11	27.10	1.76

As can be seen from Table 4.5, the algorithm for finding the minimum cut sizes in ToR graphs is very fast, also compared to the computation times for the algorithms for finding the maximum number of vertex-disjoint paths. Again, the number of branching nodes needed to find the integer optimum is small, since the solution to the LP-relaxation is integral in 98.5% of the problem instances.

Approximation Algorithms

In Table 4.6 we give results for the 2-approximation algorithms presented in Section 4.5. In the first two columns we show the graph types and the

different dates. Columns three to five contain information on the number of vertex-disjoint paths, namely the optimal value, the value found by the approximation algorithm and the computation times, and in the last three columns we give the same results for the sizes of minimum cuts. Again, all values are average values over all 1081 problem instances for a specific graph type and date.

Table 4.6: Results for approximation algorithms.

Type	*Date*	*Disjoint paths*			*Cut sizes*		
		OPT	*Approx*	*Time*	*OPT*	*Approx*	*Time*
A	18.04.2001	6.38	5.32	0.18	6.38	6.40	0.19
	04.02.2002	7.88	6.61	0.23	7.88	8.03	0.24
	06.04.2002	8.66	7.23	0.24	8.66	8.84	0.25
	09.01.2003	7.35	6.36	0.26	7.35	7.41	0.28
	10.02.2004	8.10	6.86	0.32	8.11	8.23	0.34
B	18.04.2001	6.42	5.28	0.18	6.43	6.53	0.19
	04.02.2002	8.49	6.82	0.24	8.52	8.70	0.25
	06.04.2002	9.39	7.45	0.26	9.42	9.57	0.27
	09.01.2003	7.52	6.18	0.28	7.53	7.66	0.30
	10.02.2004	8.44	6.93	0.35	8.44	8.66	0.37
C	18.04.2001	6.14	5.18	0.22	6.15	6.19	0.23
	04.02.2002	7.98	6.70	0.27	7.99	8.14	0.28
	06.04.2002	8.46	7.08	0.28	8.47	8.64	0.29
	09.01.2003	6.61	5.79	0.29	6.61	6.70	0.31
	10.02.2004	7.80	6.71	0.37	7.81	8.01	0.41
D	18.04.2001	6.34	5.19	0.22	6.35	6.49	0.23
	04.02.2002	8.01	6.57	0.27	8.02	8.09	0.27
	06.04.2002	8.69	7.20	0.27	8.70	8.97	0.28
	09.01.2003	7.30	6.24	0.28	7.30	7.37	0.30
	10.02.2004	7.92	6.74	0.35	7.93	8.08	0.38

From these results we conclude that both approximation algorithms perform really well. For the problem of finding the maximum number of disjoint paths we find that in 41.14% of all instances we get an optimal solution, and for 67.63% the difference between the optimal value and the value found by the approximation algorithm is at most 1. For the problem of finding the minimum cut sizes, 90.82% of the instances are solved optimally, and in

97.63% of the instances the difference between the optimum and the value of the approximation algorithm is at most 1. So, the heuristic for finding the minimum cut sizes is extremely well suited for these type of instances. The computation times for both algorithms are really fast: all problem instances for both problems are solved within less than one second of computation time.

4.6.4 Interpretation of the Results

In this section we describe how the results that we obtained can be interpreted. First we discuss the connectivity of the Internet as measured by the number of disjoint paths and minimum cut sizes. Then, in all four types of ToR graphs we also compute the number of edges that are contained in directed customer-provider cycles as well as the fraction of pairs of ASs that are connected with directed customer-provider paths in order to gain more insight into the AS hierarchy produced by the different inference algorithms.

Connectivity Measures for the Internet

In Table 4.7 we compare the number of vertex-disjoint paths for the different types of ToR graphs, the undirected BGP graphs and the CAIDA graph. In the second column we give the average number of vertex-disjoint paths, averaged over all pairs of ASs and all dates of the specified graph type. The third column gives the minimum number of paths found, and the last column shows the maximum number of vertex-disjoint paths (the CAIDA graph is available only for one date, and 3 of our 47 selected ASs are missing from that graph; AS pairs involving a missing AS node were thus ignored for the CAIDA graph).

In Table 4.8 we compare the size of the minimum cuts in ToR graphs with results from the undirected models, and for all graph types we give the average over all pairs and dates, the minimum value of a minimum cut, and the maximum value. For the undirected BGP graphs and the CAIDA graph,

Table 4.7: Vertex-disjoint paths in ToR and undirected graphs.

Graph Type	*avg VDP*	*min VDP*	*max VDP*
A	7.67	1	55
B	8.05	1	65
C	7.40	0	60
D	7.65	1	48
undirected BGP	13.46	2	107
CAIDA	12.74	6	108

Table 4.8: Minimum cut sizes in ToR and undirected graphs.

Graph Type	*avg CS*	*min CS*	*max CS*
A	7.68	1	56
B	8.07	1	65
C	7.40	0	60
D	7.66	1	48
undirected BGP	13.46	2	107
CAIDA	12.74	6	108

these values are the same as for the vertex-disjoint paths problem, since the max-flow min-cut equality holds for the undirected graphs.

If we compare the connectivity of the ToR graphs with the undirected models, we see a big difference (see Tables 4.7 and 4.8). The number of disjoint paths, and the cut sizes, are much larger in the undirected models. For about 72% of all pairs, the number of disjoint paths (and the minimum cut size) is at least 1.5 times bigger in the undirected models, as compared to the ToR graphs, and for approximately 44%, these values in the undirected models are at least twice as large than in the ToR graphs.

When we look at the differences in connectivity between the four different ToR graphs we see that there is no striking difference between the number of disjoint paths and the sizes of minimum cuts. Generally speaking, graphs of type B have the highest connectivity and graphs of type C have the lowest connectivity (see third column of Tables 4.3, 4.4, and 4.5). However, the connectivity of the different ToR graphs seems to be similar.

In Figure 4.3, the four types of ToR graphs are represented together with the undirected BGP graph and the CAIDA graph, all graphs taken from April 2002. We obtained similar results for the other four dates, but since we had the CAIDA graph only for April 2002, we chose to use this date for the illustration. The number of disjoint paths and the minimum cut size are shown for each of the 1081 AS pairs in all six graphs. The values are sorted in order of non-decreasing values in the undirected BGP graph. As the figure shows, there is no striking difference among the ToR graphs. The values for the undirected BGP graph, however, are significantly higher than those for the ToR graphs. This clear difference between the undirected and ToR models indicates that, in order to get an accurate picture of the Internet structure and connectivity, it is important to take routing policies into account.

The values for the CAIDA graph, which has about 6% more edges than the undirected BGP graph, are somewhat incomparable to those of the undirected BGP graph. For about 35% of the AS pairs, the CAIDA graph has more disjoint paths (up to 100 more paths for one pair), and for about 59% of the pairs, the undirected BGP graph has more disjoint paths (up to 69 more paths for one pair). This indicates that some parts of the Internet are denser (higher number of edges) in the CAIDA graph, while other parts are denser in the undirected BGP graph.

Let us now discuss trends over time. The trends for the number of disjoint paths between the different time periods are shown in Figure 4.4 for each of the four types of ToR graphs and for the undirected BGP graphs. There are four plots, each of them corresponding to a particular time period. In each plot, there is a bar for each of the five graph types. The white part of the bar represents the number of pairs of ASs for which the number of disjoint paths increased in this time period; the shaded part of the bar corresponds to the number of pairs for which the number of disjoint path stayed the

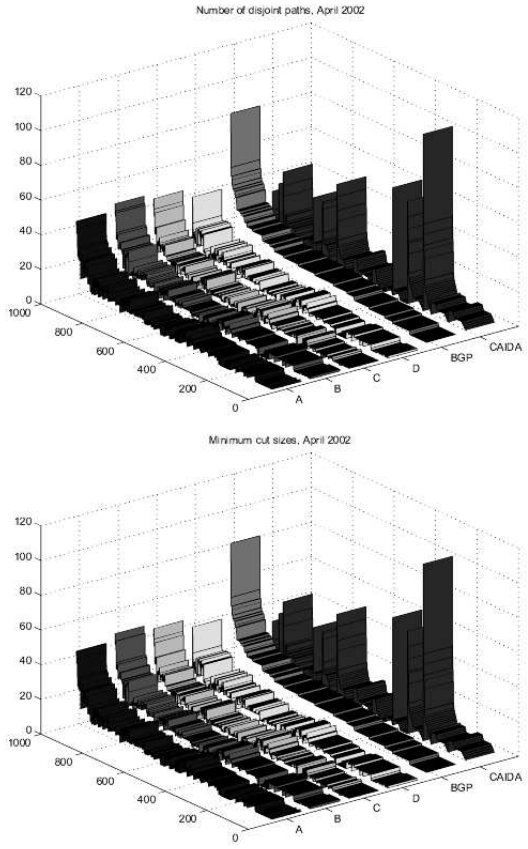

Figure 4.3: Comparison of ToR and undirected graphs.

same, and the black part is the number of pairs for which the number of paths decreased in this time period. The results for the minimum cut sizes are similar, so we omit them here.

The figure shows that the ToR graphs behave similarly for all time periods. In the first two time periods, the AS pairs with increasing connectivity form the majority. Then, in the third time period, more than half of the AS pairs display decreasing connectivity. Finally, in the fourth time period, the ToR graphs have roughly the same number of AS pairs with increasing and decreasing connectivity, respectively, while about 70% of the AS pairs display

increasing connectivity in the undirected BGP graphs.

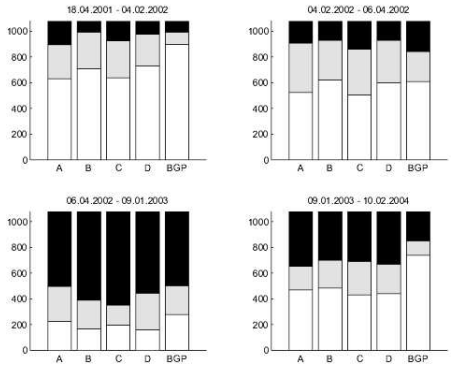

Figure 4.4: Trends over time for ToR and undirected BGP graphs.

When we study the differences between the number of disjoint paths and the cut size in ToR graphs, we find that these numbers are equal for about 99% of all AS pairs in each of the ToR graphs (see third column of Tables 4.3, 4.4, and 4.5). The absolute difference between the minimum cut size and the number of disjoint paths was never larger than 2 for any of the AS pairs in any of the ToR graphs. Thus, the minimum cut size does not differ significantly from the maximum number of disjoint valid paths in our ToR graphs. Notice that this difference could be as large as a *factor* of 2 in general graphs (Erlebach et al., 2005).

Directed Customer-Provider Cycles

We call a directed cycle (as defined in Section 4.2) in a ToR graph a *customer-provider cycle* if it contains only customer-provider edges. If the Internet was a strictly hierarchical network (i.e., if levels can be assigned to the ASs in such a way that, in any customer-provider relationship, the customer is on a lower level than the provider), one would expect that there are no customer-provider cycles in ToR graphs at all. Therefore, one might use the existence of such cycles as a sign of a misclassification.

We check the existence of such cycles in each of the ToR graphs as follows. First, we remove all sibling edges and peer-to-peer edges from the graph. Then, for each customer-provider edge from AS_i to AS_j, we calculate a shortest directed path (i.e, a path with the smallest number of edges) from AS_j to AS_i. Such a path exists if and only if the edge from AS_i to AS_j is contained in at least one directed cycle. If such a path is found, it gives us a shortest customer-provider cycle containing the edge.

We find that there are no customer-provider cycles in the ToR graphs of type C, except in the graph for 09.01.2003; for the latter date, the type C graph contains a single customer-provider cycle with four nodes (AS11563, AS19035, AS17819, AS1668). In Table 4.9, we give the results that we obtained for ToR graphs of type A, B and D. For each of the graphs, we show the total number of customer-provider edges that are contained in cycles, the minimum length of the shortest cycle containing a customer-provider edge, and the maximum length of the shortest cycle containing a customer-provider edge. We find that type B graphs have the largest number of edges contained in customer-provider cycles, type A graphs have about half as many, and type D graphs have much fewer edges contained in cycles than both A and B graphs.

As the ToR graphs of type A and B contain no sibling edges and either no or very few peer-to-peer edges, a larger number of edges contained in customer-provider cycles could be expected in these graphs. Table 4.9 confirms that significantly more edges are contained in customer-provider cycles in these graphs. Most of the cycles in the graphs of type A and B are caused by edges classified as customer-provider in A or B graphs, but classified in D graphs as peer-to-peer, sibling or provider-customer edges.

In the A graph from 18.04.2001, there are 2571 edges contained in cycles. Each of these edges is contained in a shortest cycle. Among these 2571

Table 4.9: Results for directed customer-provider cycles.

Type	*Date*	*Total*	*Min*	*Max*
A	18.04.2001	2571	3	9
	04.02.2002	2441	3	9
	06.04.2002	2278	3	10
	09.01.2003	2182	3	10
	10.02.2004	3453	3	8
B	18.04.2001	4046	3	8
	04.02.2002	4710	3	8
	06.04.2002	4825	3	9
	09.01.2003	4858	3	9
	10.02.2004	6802	3	9
D	18.04.2001	318	3	20
	04.02.2002	16	3	5
	06.04.2002	9	3	3
	09.01.2003	69	3	11
	10.02.2004	428	3	14

shortest customer-provider cycles (we consider a cycle multiple times if it is the shortest customer-provider cycle of several edges), 1909 have an edge classified as peer-to-peer in the corresponding D graph, 574 of the remaining ones have an edge classified as sibling edge in the D graph, and 67 of the remaining ones have an edge classified as provider-customer edge in D. Only 21 of the 2571 cycles are also present in the D graph. Qualitatively similar results are obtained for all dates for the A and B graphs.

Analyzing the directed cycles in the D graphs, we found that all customer-provider cycles can be eliminated by deleting a very small number of edges (12, 4, 3, 8, and 11 edges, respectively, in the five D graphs from 18.04.2001 to 10.02.2004).

We checked manually 10 edges that were contained in more than 50 discovered cycles (up to 348 cycles) in the D graph from 10.02.2004, using the Nemecis tool (NEMECIS) to access data from Internet Routing Registries. For three of these edges there was no information in the Internet Registries,

6 of them were classified as peer-to-peer edges (i.e., at least one of the two ASs registered this particular edge as peer-to-peer) and only one edge was registered as customer-provider (confirming its classification in the D graph).

Although it is still possible that some of the directed customer-provider cycles are not caused by misclassifications, we think that they are a good starting point for the detection of misclassifications, in particular if their analysis is combined with a comparison between the different ToR graphs and checking of entries in Internet Registries. Such cycles could be used to introduce peer-to-peer edges and sibling edges into the ToR graphs of type A and B, which contain essentially only customer-provider edges (in our type B graphs, the only peer-to-peer edges are those that were left unclassified by the algorithm from Di Battista et al. (2003)).

The Depth of the Provider Hierarchy in ToR Graphs

Finally, in order to examine the typical nature of AS paths in the different ToR graphs, we investigated how many pairs of vertices can be connected by directed paths, i.e., by paths going "only up" or "only down." A path $AS_1, \ldots, AS_k$ between two ASs AS_1 and AS_k such that each AS_i is a customer of AS_{i-1}, for $2 \leq i \leq k$, is called a *customer chain*. In our experiments, we check for all pairs of ASs in ToR graphs (except the pairs involving leaf vertices) whether one of the two ASs is connected to the other via a customer chain. For such pairs of vertices, it is possible to use paths only through customers (at least in one of the two directions) and thus take advantage of the "customer-preference" policy. Namely, routing through a customer brings profit, through a peer is neutral, and through a provider incurs costs for the sender (Spring et al., 2003).

The statistics about customer chains in all four types of ToR graphs are given in Table 4.10. This table shows for each of the five dates the percent-

Table 4.10: Percentage of pairs of ASs connected by customer chains.

Date	*A*	*B*	*C*	*D*
18.04.2001	10.01%	14.93%	0.52%	2.25%
04.02.2002	7.52%	14.03%	0.56%	0.67%
06.04.2002	7.02%	14.05%	0.53%	0.59%
09.01.2003	6.84%	13.62%	0.47%	0.84%
10.02.2004	7.60%	14.65%	0.53%	1.42%

ages of pairs of ASs that are connected by customer chains in all our types of ToR graphs.

About 6–10% of all pairs in type A graphs and 13–15% of all pairs in type B graphs are connected by customer chains. For graphs of type C and D the number is significantly smaller, which was to be expected because they contain substantially more edges that are not customer-provider edges. This indicates that in graphs of type A and B, the hierarchy seems to be similar and tends to be deep. In type C and D graphs, the hierarchy seems to be wider, as there are many more pairs that are connected only through paths going "up and then down."

4.7 Conclusions

We have compared different types of graphs with inferred AS relationships (ToR graphs) regarding connectivity measures and path characteristics. We have studied the maximum number of disjoint valid paths and the minimum cut size for selected AS pairs. Since both problems are $\mathcal{NP}$-hard, we have designed and implemented several algorithms that allowed us to compute optimal values for all pairs among a set of representative ASs. For the problem of finding the maximum number of disjoint paths between any pair of ASs, we have implemented two exact algorithms, the first one being a branch-and-price algorithm based on an integer programming formulation of the problem, and the second one being a branch-and-bound algorithm in

which we perform a max-flow calculation in each node of the search tree. From the results we conclude that the latter algorithm is often faster than the first but may require excessive computation times on certain inputs, while the computation times for the branch-and-price algorithm are always acceptable and do not display such a variability. For the problem of finding the minimum cut sizes, we have implemented a branch-and-cut algorithm that performs really well, with average computation times around one to two seconds for instances with up to 11,000 nodes and 30,000 edges.

The results of these algorithms allow us to quantify the differences in connectivity between ToR graphs and the traditional undirected model of the Internet, which ignores routing policies. We find that about 44% of the selected AS pairs have more than twice as many disjoint paths in the undirected model than in the ToR graphs, which implies that the use of ToR graphs is crucial for Internet analysis and simulations that are sensitive to connectivity properties, e.g. in studies concerning topological robustness, multi-path routing, etc. We have also investigated the increase of connectivity over time and found that the number of disjoint paths between ASs seems to grow for fewer AS pairs in the ToR graphs than in the undirected graph model.

Comparing the ToR graphs with each other, we find that on the average they do not differ much with respect to the number of disjoint paths and the minimum cut sizes between AS pairs. On the other hand, concerning the hierarchy (observed indirectly by counting the number of AS pairs connected through customer chains) it turns out that A and B graphs are relatively similar to each other, but different from C and D graphs—their hierarchy appears to be deeper than that of C and D graphs. In addition, we find that the investigation of short directed customer-provider cycles in the ToR graphs can help to detect misclassifications and may lead to new approaches for introducing peer-to-peer or sibling relationships into A and B graphs, which can make these models more realistic.

While our investigations provide some insight into the properties of the ToR graphs produced by the different available inference algorithms, it is not possible for us to identify one of these algorithms as better than the others. Researchers who employ ToR graphs in their research should be aware of the differences in the ToR models produced by different algorithms and make sure that their conclusions are not biased by the choice of ToR graph. Our findings can help in making informed decisions about the choice of a ToR graph model.

Furthermore, our approach of adapting the classical connectivity measures, maximum number of disjoint paths and minimum cut size, to valley-free paths in ToR graphs can be useful in further research on robustness issues in the Internet. Besides, it may be possible to adapt our branch-and-price approach to incorporate other types of constraints on valid paths, thus allowing the analysis of connectivity properties of other networks with special routing constraints as well.

The known algorithms for inferring AS relationships from Gao (2001), Subramanian et al. (2002), Di Battista et al. (2003), and Erlebach et al. (2002) all need data from BGP routing tables as input. As the data from BGP routing tables is not always complete or accurate (the impact of this is demonstrated convincingly by the huge difference in the number of disjoint paths for certain AS pairs in the undirected BGP graph and the CAIDA graph, see Figure 4.3), it would be an interesting question for future research whether good inference of AS relationships is also possible without knowledge of BGP routing tables. Such an inference algorithm could then also be used for classifying AS relationships in more complete undirected AS graphs (such as the union of the undirected BGP graph and the CAIDA graph) or in synthetic graph models obtained from Internet topology generators. A different approach in the latter direction has been explored by Chang et al. (2003), where a new optimization-driven model for Internet

growth is presented that allows the generation of synthetic AS graphs containing only customer-provider relationships.

Finally, let us summarize our findings by answering the research question posed in Section 4.1.

- The undirected graph model of the Internet topology is not suited for studying stability issues of the Internet. Of course, any graph-theoretical model of the Internet is an approximation of reality. However, some approximations are better than others: our results show that from the viewpoint of connectivity, using the undirected graph model may lead to serious misjudging of the connectivity.
- The different heuristics (Gao, 2001; Subramanian et al., 2002; Di Battista et al., 2003; Erlebach et al., 2002) used for constructing a topology do not differ much with respect to the connectivity measures. On the average, they all have a similar number of disjoint paths and minimum cut size between pairs of ASs.
- Graphs of type A and B do not have sibling edges and no or very few peer-to-peer edges, and this causes the existence of a relatively large number of customer-provider cycles. Graphs of type D contain a significant number of peer-to-peer and sibling edges, and they have few customer-provider cycles. Graphs of type C contain no customer-provider cycles at all (except for a single cycle of length 4 for one date). Depending on the question one wants to investigate, this could be relevant. The directed customer-provider cycles in ToR graphs can be useful for the detection of misclassified edges, especially if the analysis is combined with a comparison between the different ToR graphs.
- We obtain optimal solutions to the problems of finding the maximum number of vertex-disjoint paths and minimum cut sizes, using the exact algorithms proposed in this chapter. The algorithms require, on average, a small amount of time to find these optimal values. However,

for a small number of instances, the branch-and-bound algorithm was unable to find the optimal solution.

- The performance of the approximation algorithms is reasonable. Especially for the problem of finding minimum cut sizes, the approximation algorithm performs really well. Both these algorithms are much faster than the exact algorithms.

Chapter 5

Topics for Future Research

To conclude this thesis, we describe a number of topics for future research in this chapter. In Section 5.1 we discuss some topics for future research on partitioning partially ordered sets, and Section 5.2 deals with the connectivity of the Internet.

5.1 Partitioning Partially Ordered Sets

In Chapters 2 and 3 we analyzed the problem of partitioning a (weighted) partially ordered set into chains of bounded size. This problem is a generalization of a fundamental problem in operations research, with many practical applications. Since this generalization has not yet been studied extensively, there are many questions left that need to be answered. In the following, we describe a number of these matters that are interesting for future research.

5.1.1 The Clique Width of Graphs

An outcome from this dissertation is that the concept of (bounded) clique width is relevant for our setting described in Chapter 2. This is a relatively new concept from the field of graph theory. Since we are dealing with a new

concept, some issues are not yet completely understood. For instance, it is still unknown how to compute the clique width for an arbitrary graph. It is widely believed that it is $\mathcal{NP}$-complete to determine the clique width of an arbitrary graph, but so far there is no proof of this claim. There are some positive results for specific graph classes, for example, the clique width of trees is easy to compute, as well as the clique width of co-graphs (a graph G is a co-graph if it can be constructed from isolated vertices by disjoint union and join operations (Brandstädt et al., 2002)). However, for partial orders it is unknown how to compute the corresponding clique width. Moreover, it is even unknown whether it is possible to do this efficiently. It would be interesting to find out whether it is possible to compute the clique-width of an arbitrary partial order efficiently, and if so, how this can be done. This would certainly be relevant for the design of new algorithms.

5.1.2 Improving the Approximation Ratio

Another topic concerns the approximation ratio of 2 that was established in Chapter 3. We have shown that the gap between the value of the LP-relaxation of a straightforward set-partitioning formulation and the value of an optimal solution is as large as a factor 2. This implies that we can't improve the approximation ratio of 2 using a straightforward rounding approach.

As a start one could analyze the problem with unit weights (as described in Chapter 2) in more detail. So far, we only focussed on exact solution methods for solving this problem, but it is interesting to find good approximation algorithms that can deal with larger problem instances. So far, the best approximation ratio for this problem is equal to 2, since the approximation algorithm described in Chapter 3 can be applied to the case with unit-weights. However, it seems not unlikely that one can do better than an approximation ratio of 2 for this problem.

It also would be nice to be able to obtain the optimal solutions to the problem of partitioning a weighted partial order, as described in Chapter 3, in order to gain more insight to the quality of the 2-approximation algorithm for solving this problem.

5.1.3 The Price of Stability

In the application from the field of pallet loading described in Chapter 2, there is a restriction stating that we can only put smaller items on top of larger ones. This stability restriction guarantees that the pallets are stable, and that they arrive at the clients in good shape. However, in many applications this restriction is unnecessary. Therefore, we want to consider the problem disregarding this stability restriction.

For the problem with unit weights (i.e., the objective is to minimize the total number of pallets), it might be interesting to look at the effect of the stability restriction, however, the solution to the problem without this stability constraint is trivial (for the case that the pallets must have bounded size, as well as for the case that there is no restriction on the number of items on a pallet). However, for the weighted problem (i.e., in terms of the application this means that all items have a weight corresponding to its area, and the objective is to minimize total area; see also Section 3.1.2), this is not the case. We can solve the problem in which the pallets can hold as many items as possible (see Section 2.4), but for the problem with the size-constraint, we have no results yet; even the complexity of this variant is unknown.

5.1.4 The Online Problem

In my thesis we only consider the off-line version of the problem of partitioning a partial order, that is, the version where all elements are given before we start solving the problem. In the on-line version of this problem,

we are given a number of elements, and while we are solving the problem, more and more elements become available. Obviously, it is more difficult to solve the on-line problem, since we do not have all information at the start of the algorithm. However, in production environments we are often dealing with on-line problems, since it is not known in advance which orders are produced at what time. Therefore it could be interesting to explore the on-line problem in more detail. More specifically, we are interested in finding efficient algorithms for solving the on-line problem. *Competitive analysis* is a tool to measure the quality of an on-line algorithm: the performance of the on-line algorithm is compared to the optimal off-line solutions. We are interested in finding on-line algorithm for which we are able to give a performance guarantee with respect to the competitive ratio.

5.2 Connectivity of the Internet

In Chapter 4 we focussed on creating an accurate model of the Internet. We evaluated four different algorithms for inferring AS relationships, and we compared the topologies produced by these algorithms with each other and with the previously adopted undirected model. Although the results of our analysis clearly show that incorporating AS relationships leads to more accurate models of the Internet, we were unable to identify one of the directed graph models as the best one. It might be interesting to extend the analysis in some way in order to obtain a model of the Internet that incorporated the strengths of all four inference algorithms that are considered in this thesis.

A part of our analysis dealt with the detection of directed customer-provider cycles, which can be seen as misclassifications. An interesting problem would be to determine the minimum number of customer-provider edges that need to be deleted from the graph (or given a different label) in order to ensure that the resulting graph is acyclic. This problem is known as the feedback arc set problem. Solving this problem might provide us with a more accu-

rate model of the AS relationships.

List of Figures

List of Tables

Bibliography

Aarts, E. H. L. and Lenstra, J. K., editors (1997). *Local Search in Combinatorial Optimization*. John Wiley & Sons, Chichester.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ.

Akella, A., Chawla, S., Kannan, A., and Seshan, S. (2003a). Scaling properties of the internet graph. In *Proceedings of the* 22^{nd} *ACM Symposium on Principles of Distributed Computing (PODC 2003), Boston, MA*.

Akella, A., Maggs, B., Seshan, S., Shaikh, A., and Sitaraman, R. (2003b). A measurement-based analysis of multihoming. In *Proceedings of the ACM SIGCOMM 2003 Conference, Karlsruhe, Germany*.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (1999). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, Berlin.

Baker, B. S. and Coffman, Jr., E. G. (1996). Mutual exclusion scheduling. *Theoretical Computer Science*, 162:225–245.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46:316–329.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

BGPSAT. Di Battista, G., Patrignani, M., and Pizzonia, M. Computing the types of the relationships between Autonomous Systems. Project webpage: http://www.dia.uniroma3.it/∼compunet/relationships/.

Bischoff, E. E. (1991). Stability aspects of pallet loading. *OR Spektrum*, 13:189–197.

Bodlaender, H. L. and Jansen, K. (1993). On the complexity of scheduling incompatible jobs with unit-times. *Mathematical Foundations of Computer Science*, LNCS 711:291–300.

Boudhar, M. (2003). Scheduling a batch processing machine with bipartite compatibility graphs. *Mathematical Methods of Operations Research*, 57:513–527.

Brandstädt, A., Dragan, F. F., Le, H.-O., and Mosca, R. (2002). New graph classes of bounded clique-width. To appear in *Theory of Computing Systems*.

Brandstädt, A. and Lozin, V. V. (2003). On the linear structure and clique-width of bipartite permutation graphs. *Ars Combinatoria*, LXVII:273–281.

Byun, C. (2001). Lower bounds for large-scale set partitioning problems. ZIB-Report 01-06, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany.

CAIDA, The Cooperative Association for Internet Data Analysis. Internet Topology Data Kit #0204 (Data collected as part of CAIDA's skitter initiative). http://www.caida.org/tools/measurement/skitter/.

Caprara, A. and Fischetti, M. (1997). Branch-and-Cut Algorithms. In Dell'Amico, M., Maffioli, F., and Martello, S., editors, *Annotated Bib-*

liographies in Combinatorial Optimization. John Wiley & Sons, Chichester.

Chang, H., Govidan, R., Jamin, S., Shenker, S. J., and Willinger, W. (2004). Towards capturing representative AS-level Internet topologies. *Computer Networks Journal*, 44:737–755.

Chang, H., Jamin, S., and Willinger, W. (2003). What causal forces shape Internet connectivity at the AS-level? Technical Report CSE-475-03, EECS Department, University of Michigan, Ann Arbor, MI.

Chekuri, C. and Khanna, S. (2005). A PTAS for the multiple knapsack problem. To appear in *SIAM Journal on Computing*.

CIMVP. Agarwal, S., Subramanian, L., Rexford, J., and Katz, R.H. Characterizing the Internet hierarchy from Multiple Vantage Points. Project webpage: http://www.cs.berkeley.edu/∼sagarwal/research/BGP-hierarchy/.

Courcelle, B., Engelfriet, J., and Rozenberg, G. (1993). Handle-rewriting hypergraph grammars. *Journal of Computer and System Science*, 46:218–270.

Courcelle, B. and Olariu, S. (2000). Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101:77–114.

Denardo, E. V. (1982). *Dynamic Programming Models and Applications*. Prentice-Hall, Englewood Cliffs, NJ.

Desrosiers, J., Dumas, Y., Desrochers, M., Soumis, F., Sanso, B., and Trudeau, P. (1991). A breakthrough in airline crew scheduling. *Cahiers du GERAD* G-91-11.

Di Battista, G., Patrignani, M., and Pizzonia, M. (2003). Computing the types of the relationships between autonomous systems. In *Proceedings of the IEEE INFOCOM 2003 Conference, San Francisco, CA*.

Dilworth, R. P. (1950). A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–166.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159.

Erlebach, T., Hall, A., Panconesi, A., and Vukadinović, D. (2005). Cuts and disjoint paths in the valley-free path model of Internet BGP routing. In *Proceedings of the First Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN 2004), Alberta, Canada.* LNCS 3405:49-62.

Erlebach, T., Hall, A., and Schank, T. (2002). Classifying customer-provider relationships in the Internet. In *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN 2002), Cambridge, MA.*

Espelage, W., Gurski, F., and Wanke, E. (2001). How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *Proceedings of the* 27^{th} *International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2001), Boltenhagen, Germany.* LNCS 2204:117-128.

Feigenbaum, J., Papadimitiou, C., Sami, R., and Shenker, S. (2002). A BGP-based mechanism for lowest-cost routing. In *Proceedings of the* 21^{nd} *ACM Symposium on Principles of Distributed Computing (PODC 2002), Monterey, CA.*

Felsner, S. and Wernisch, L. (1998). Maximum k-chains in planar point sets: combinatorial structure and algorithms. *SIAM Journal of Computing*, 28:192–209.

Finke, G., Jost, V., Queyranne, M., and Sebö, A. (2004). Batch processing with interval graph compatibilities between tasks. In *Proceedings of Discrete Optimization Methods in Production and Logistics (DOM 2004), Omsk-Irkutsk, Russia.*

G, Y.-G. and Kang, M.-K. (2001). A fast algorithm for two-dimensional pallet loading problems of large size. *European Journal of Operational Research*, 134:193–202.

Gao, L. (2001). On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9:733–745.

Gao, L. and Wang, F. (2002). The extent of AS path inflation by routing policies. In *Proceedings of IEEE Global Internet Symposium 2002, Taipei, Taiwan*.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Golumbic, M. C. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, NY.

Grötschel, M., Lovász, L., and Schrijver, A. (1988). *Geometric algorithms and combinatorial optimization*. Springer-Verlag, Berlin.

Hansen, P., Hertz, A., and Kuplinsky, J. (1993). Bounded vertex colorings of graphs. *Discrete Mathematics*, 111:305–312.

Harary, F. (1969). *Graph Theory*. Addison-Wesley, Massachussets.

Huston, G. (1999a). Interconnection, peering and settlements - Part I. *Internet Protocol Journal*, 2:2–16.

Huston, G. (1999b). Interconnection, peering and settlements - Part II. *Internet Protocol Journal*, 2:2–23.

Jansen, K. (2003). The mutual exclusion scheduling problem for permutation and comparability graphs. *Information and Computation*, 180:71–81.

Jarvis, M. and Zhou, B. (2001). Bounded vertex coloring of trees. *Discrete Mathematics*, 232:145–151.

Johnson, E. L., Nemhauser, G. L., and Savelsbergh, M. W. P. (2000). Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing*, 12:2–23.

Kann, V. (1991). Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27–35.

Labovitz, C., Ahuja, A., Wattenhofer, R., and Venkatachary, S. (2001). The impact of Internet policy and topology on delayed routing convergence. In *Proceedings of the IEEE INFOCOM 2001 Conference, Anchorage, AK*.

Ladányi, L., Ralphs, T. K., and Trotter, Jr., L. E. (2001). Branch, cut, and price: Sequential and parallel. In Naddef, D. and Juenger, M., editors, *Computational Combinatorial Optimization*. Springer, Berlin.

Lee, K., Chang, S. Y., and Hong, Y. (2004). Continuous slab caster scheduling and interval graphs. *Production Planning and Control*, 15:495–501.

Letchford, A. N. and Amaral, A. (2001). Analysis of upper bounds for the pallet loading problem. *European Journal of Operational Research*, 132:582–593.

LRIP. Figueiredo, D.R., Ge, Z., and Jaiswal, S. Logical Relationship Inference Program (implementation of algorithms from Gao, 2001). Webpage: http://www-net.cs.umass.edu/∼ratton/AS/.

Moonen, L. S. (2001). Optimaliseren van een productieproces. Master's thesis, Maastricht University (in Dutch).

Moonen, L. S. and Spieksma, F. C. R. (2005). Exact algorithms for a loading problem with bounded clique width. To appear in *INFORMS Journal on Computing*.

Morabito, R. and Morales, S. (1998). A simple and effective procedure for the manufacturer's pallet loading problem. *Journal of the Operational Research Society*, 49:819–828.

Moret, B. M. (1998). *The Theory of Computation*. Addison-Wesley, MA.

NEMECIS. University of California Riverside website. http://ira.cs.ucr.edu:8080/Nemecis.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley, NY.

Ore, O. (1962). *Theory of Graphs*. American Mathematical Society Colloquium Publications, Volume 38, Providence, RI.

OREGON. Oregon website: http://www.routeviews.org.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, NY.

Papadimitriou, C. H. and Yannakakis, M. (1991). Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440.

Rimondini, M., Pizzonia, M., Di Battista, G., and Patrignani, M. (2004). Algorithms for the inference of the commercial relationships between autonomous systems: Results analysis and model validation. In *Proceedings of the International Workshop on Inter-domain Performance and Simulation (IPS 2004), Budapest, Hungary*.

Scheithauer, G. and Terno, J. (1996a). The G4-heuristic for the pallet loading problem. *Journal of the Operational Research Society*, 47:511–522.

Scheithauer, G. and Terno, J. (1996b). A heuristic approach for solving the multi-pallet packing problem. Technical report, Dresden University of Technology, Dresden, Germany.

Schrijver, A. (2003). *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, Berlin.

Shum, H. and Trotter, Jr., L. E. (1996). Cardinality-restricted chains and antichains in partially ordered sets. *Discrete Applied Mathematics*, 65:421–439.

Sierksma, G. (1996). *Linear and Integer Programming: Theory and Practice*. Dekker, NY.

Silver, E. A. (2004). An overview of heuristic solution methods. *Journal of the Operational Research Society*, 55:936–956.

Spring, N., Mahajan, R., and Anderson, T. (2003). Quantifying the causes of path inflation. In *Proceedings of the ACM SIGCOMM 2003 Conference, Karlsruhe, Germany*.

Subramanian, L., Agarwal, S., Rexford, J., and Katz, R. (2002). Characterising the internet hierarchy from multiple vantage points. In *Proceedings of the IEEE INFOCOM 2002 Conference, New York, NY*.

Tangmunarunkit, H., Govidan, R., and Shenker, S. (2001a). Internet path inflation due to policy routing. In *Proceedings of the International Conference and Exhibits on the Convergence of IT and Communications (SPIE ITcom 2001), Denver, CO*.

Tangmunarunkit, H., Govidan, R., and Shenker, S. (2003). Internet topology: discovery and policy impact. In Park, K. and Willinger, W., editors, *The Internet as a Large-Scale Complex System*. Oxford University Press.

Tangmunarunkit, H., Govidan, R., Shenker, S., and Estrin, D. (2001b). The impact of routing policy on Internet paths. In *Proceedings of the IEEE INFOCOM 2003 Conference, San Francisco, CA*.

Teixeira, R., Marzullo, K., Savage, S., and Voelker, G. (2003). Characterizing and measuring path diversity of Internet topologies. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2003), San Diego, CA*.

Terno, J., Scheithauer, G., Sommerweiß, U., and Riehme, J. (2000). An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research*, 123:372–381.

Trotter, W. T. (1992). *Combinatorics and partially ordered sets: dimension theory*. The John Hopkins University Press, Baltimore.

Vance, P. H., Atamtürk, A., Barnhart, C., Gelman, E., Johnson, E. L., Krishna, A., Mahidhara, D., Nemhauser, G. L., and Rebello, R. (1997). A heuristic branch-and-price approach for the airline crew pairing problem. Technical Report LEC-97-06, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Vanderbeck, F. (1994). *Decomposition and column generation for integer programs*. PhD thesis, Faculté des Sciences Appliquées, Université Catholique de Louvain, Louvain-la-Neuve, Belgium.

Vanderbeck, F. and Wolsey, L. A. (1996). An exact algorithm for IP column generation. *Operations Research Letters*, 19:151–159.

Vazirani, V. (2001). *Approximation Algorithms*. Springer-Verlag, Berlin.

Vukadinović, D., Huang, P., and Erlebach, T. (2002). On the spectrum and structure of Internet topology graphs. In *Proceedings of Innovative Internet Computing Systems, Kühlungsborn, Germany*. LNCS 2346:83-95.

Wanke, E. (1994). k-NLC graphs under polynomial algorithms. *Discrete Applied Mathematics*, 54:251–266.

Wolsey, L. A. (1998). *Integer Programming*. Wiley, NY.

Xia, J. and Gao, L. (2004). On the evaluation of as relationship inferences. In *Proceedings of IEEE Global Communications Conference (GLOBECOM 2004), Dallas, TX*.

Doctoral Dissertations from the Faculty of Economic and Applied Economic Sciences

From August 1, 1971

1. GEPTS, Stefaan (1971)
   Stability and efficiency of resource allocation processes in discrete commodity spaces. Leuven, KUL, 1971. 86 pp.

2. PEETERS, Theo (1971)
   Determinanten van de internationale handel in fabrikaten. Leuven, Acco, 1971. 290 pp.

3. VAN LOOY, Wim (1971)
   Personeelsopleiding: een onderzoek naar investeringsaspekten van opleiding. Hasselt, Vereniging voor wetenschappelijk onderzoek in Limburg, 1971. VII, 238 pp.

4. THARAKAN, Mathew (1972)
   Indian exports to the European community: problems and prospects. Leuven, Faculty of economics and applied economics, 1972. X,343 pp.

5. HERROELEN, Willy (1972)
   Heuristische programmatie: methodologische benadering en praktische toepassing op complexe combinatorische problemen. Leuven, Aurelia scientifica, 1972. X, 367 pp.

6. VANDENBULCKE, Jacques (1973)
   De studie en de evaluatie van data-organisatiemethodes en data-zoekmethodes. Leuven, s.n., 1973. 3 V.

7. PENNYCUICK, Roy A. (1973)
   The economics of the ecological syndrome. Leuven, Acco, 1973. XII, 177 pp.
8. KAWATA, T. Bualum (1973)
   Formation du capital d'origine belge, dette publique et stratégie du développement au Zaire. Leuven, KUL, 1973. V, 342 pp.
9. DONCKELS, Rik (1974)
   Doelmatige oriëntering van de sectorale subsidiepolitiek in België: een theoretisch onderzoek met empirische toetsing. Leuven, K.U.Leuven, 1974. VII, 156 pp.
10. VERHELST, Maurice (1974)
    Contribution to the analysis of organizational information systems and their financial benefits. Leuven, K.U.Leuven, 1974. 2 V.
11. CLEMEUR, Hugo (1974)
    Enkele verzekeringstechnische vraagstukken in het licht van de nutstheorie. Leuven, Aurelia scientifica, 1974. 193 pp.
12. HEYVAERT, Edward (1975)
    De ontwikkeling van de moderne bank- en krediettechniek tijdens de zestiende en zeventiende eeuw in Europa en te Amsterdam in het bijzonder. Leuven, K.U.Leuven, 1975. 186 pp.
13. VERTONGHEN, Robert (1975)
    Investeringscriteria voor publieke investeringen: het uitwerken van een operationele theorie met een toepassing op de verkeersinfrastructuur. Leuven, Acco, 1975. 254 pp.
14. Niet toegekend.
15. VANOVERBEKE, Lieven (1975)
    Microeconomisch onderzoek van de sectoriële arbeidsmobiliteit. Leuven, Acco, 1975. 205 pp.
16. DAEMS, Herman (1975)
    The holding company: essays on financial intermediation, concentration and capital market imperfections in the Belgian economy. Leuven, K.U.Leuven, 1975. XII, 268 pp.
17. VAN ROMPUY, Eric (1975)
    Groot-Brittannië en de Europese monetaire integratie: een onderzoek naar de gevolgen van de Britse toetreding op de geplande Europese monetaire unie. Leuven, Acco, 1975. XIII, 222 pp.

18. MOESEN, Wim (1975)
Het beheer van de staatsschuld en de termijnstructuur van de intrestvoeten met een toepassing voor België. Leuven, Vander, 1975. XVI, 250 pp.

19. LAMBRECHT, Marc (1976)
Capacity constrained multi-facility dynamic lot-size problem. Leuven, KUL, 1976. 165 pp.

20. RAYMAECKERS, Erik (1976)
De mens in de onderneming en de theorie van het producenten-gedrag: een bijdrage tot transdisciplinaire analyse. Leuven, Acco, 1976. XIII, 538 pp.

21. TEJANO, Albert (1976)
Econometric and input-output models in development planning: the case of the Philippines. Leuven, KUL, 1976. XX, 297 pp.

22. MARTENS, Bernard (1977)
Prijsbeleid en inflatie met een toepassing op België. Leuven, KUL, 1977. IV, 253 pp.

23. VERHEIRSTRAETEN, Albert (1977)
Geld, krediet en intrest in de Belgische financiële sector. Leuven, Acco, 1977. XXII, 377 pp.

24. GHEYSSENS, Lieven (1977)
International diversification through the government bond market: a risk-return analysis. Leuven, s.n., 1977. 188 pp.

25. LEFEBVRE, Chris (1977)
Boekhoudkundige verwerking en financiële verslaggeving van huurkooptransacties en verkopen op afbetaling bij ondernemingen die aan consumenten verkopen. Leuven, KUL, 1977. 228 pp.

26. KESENNE, Stefan (1978)
Tijdsallocatie en vrijetijdsbesteding: een econometrisch onderzoek. Leuven, s.n., 1978. 163 pp.

27. VAN HERCK, Gustaaf (1978)
Aspecten van optimaal bedrijfsbeleid volgens het marktwaardecriterium: een risico-rendementsanalyse. Leuven, KUL, 1978. IV, 163 pp.

28. VAN POECK, Andre (1979)
World price trends and price and wage development in Belgium: an investigation into the relevance of the Scandinavian model of inflation for Belgium. Leuven, s.n., 1979. XIV, 260 pp.

29. VOS, Herman (1978)
De industriële technologieverwerving in Brazilië: een analyse. Leuven, s.n., 1978. onregelmatig gepagineerd.

30. DOMBRECHT, Michel (1979)
Financial markets, employment and prices in open economies. Leuven, KUL, 1979. 182 pp.

31. DE PRIL, Nelson (1979)
Bijdrage tot de actuariële studie van het bonus-malussysteem. Brussel, OAB, 1979. 112 pp.

32. CARRIN, Guy (1979)
Economic aspects of social security: a public economics approach. Leuven, KUL, 1979. onregelmatig gepagineerd

33. REGIDOR, Baldomero (1979)
An empirical investigation of the distribution of stock-market prices and weak-form efficiency of the Brussels stock exchange. Leuven, KUL, 1979. 214 pp.

34. DE GROOT, Roger (1979)
Ongelijkheden voor stop loss premies gebaseerd op E.T. systemen in het kader van de veralgemeende convexe analyse. Leuven, KUL, 1979. 155 pp.

35. CEYSSENS, Martin (1979)
On the peak load problem in the presence of rationizing by waiting. Leuven, KUL, 1979. IX, 217 pp.

36. ABDUL RAZK ABDUL (1979)
Mixed enterprise in Malaysia: the case study of joint venture between Malysian public corporations and foreign enterprises. Leuven, KUL, 1979. 324 pp.

37. DE BRUYNE, Guido (1980)
Coordination of economic policy: a game-theoretic approach. Leuven, KUL, 1980. 106 pp.

38. KELLES, Gerard (1980)
Demand, supply, price change and trading volume on financial markets of the matching-order type. = Vraag, aanbod, koersontwikkeling en omzet op financiële markten van het Europese type. Leuven, KUL, 1980. 222 pp.

39. VAN EECKHOUDT, Marc (1980)
De invloed van de looptijd, de coupon en de verwachte inflatie op het opbrengstverloop van vastrentende finaciële activa. Leuven, KUL, 1980. 294 pp.

40. SERCU, Piet (1981)
Mean-variance asset pricing with deviations from purchasing power parity. Leuven, s.n., 1981. XIV, 273 pp.

41. DEQUAE, Marie-Gemma (1981)
Inflatie, belastingsysteem en waarde van de onderneming. Leuven, KUL, 1981. 436 pp.

42. BRENNAN, John (1982)
An empirical investigation of Belgian price regulation by prior notification: 1975 - 1979 - 1982. Leuven, KUL, 1982. XIII, 386 pp.

43. COLLA, Annie (1982)
Een econometrische analyse van ziekenhuiszorgen. Leuven, KUL, 1982. 319 pp.

44. Niet toegekend.

45. SCHOKKAERT, Eric (1982)
Modelling consumer preference formation. Leuven, KUL, 1982. VIII, 287 pp.

46. DEGADT, Jan (1982)
Specificatie van een econometrisch model voor vervuilingsproblemen met proeven van toepassing op de waterverontreiniging in België. Leuven, s.n., 1982. 2 V.

47. LANJONG, Mohammad Nasir (1983)
A study of market efficiency and risk-return relationships in the Malaysian capital market. s.l., s.n., 1983. XVI, 287 pp.

48. PROOST, Stef (1983)
De allocatie van lokale publieke goederen in een economie met een centrale overheid en lokale overheden. Leuven, s.n., 1983. onregelmatig gepagineerd.

49. VAN HULLE, Cynthia (1983)
Shareholders' unanimity and optimal corporate decision making in imperfect capital markets. s.l., s.n., 1983. 147 pp. + appendix.

50. VAN WOUWE, Martine (2/12/83)
Ordening van risico's met toepassing op de berekening van ultieme ruïnekansen. Leuven, s.n., 1983. 109 pp.

51. D'ALCANTARA, Gonzague (15/12/83)
SERENA: a macroeconomic sectoral regional and national account econometric model for the Belgian economy. Leuven, KUL, 1983. 595 pp.

52. D'HAVE, Piet (24/02/84)
De vraag naar geld in België. Leuven, KUL, 1984. XI, 318 pp.

53. MAES, Ivo (16/03/84)
The contribution of J.R. Hicks to macro-economic and monetary theory. Leuven, KUL, 1984. V, 224 pp.

54. SUBIANTO, Bambang (13/09/84)
A study of the effects of specific taxes and subsidies on a firms' R&D investment plan. s.l., s.n., 1984. V, 284 pp.

55. SLEUWAEGEN, Leo (26/10/84)
Location and investment decisions by multinational enterprises in Belgium and Europe. Leuven, KUL, 1984. XII, 247 pp.

56. GEYSKENS, Erik (27/03/85)
Produktietheorie en dualiteit. Leuven, s.n., 1985. VII, 392 pp.

57. COLE, Frank (26/06/85)
Some algorithms for geometric programming. Leuven, KUL, 1985. 166 pp.

58. STANDAERT, Stan (26/09/86)
A study in the economics of repressed consumption. Leuven, KUL, 1986. X, 380 pp.

59. DELBEKE, Jos (03/11/86)
Trendperioden in de geldhoeveelheid van België 1877-1983: een theoretische en empirische analyse van de "Banking school" hypothese. Leuven, KUL, 1986. XII, 430 pp.

60. VANTHIENEN, Jan (08/12/86)
Automatiseringsaspecten van de specificatie, constructie en manipulatie van beslissingstabellen. Leuven, s.n., 1986. XIV, 378 pp.

61. LUYTEN, Robert (30/04/87)
A systems-based approach for multi-echelon production/inventory systems. s.l., s.n., 1987. 3V.

62. MERCKEN, Roger (27/04/87)
De invloed van de data base benadering op de interne controle. Leuven, s.n., 1987. XIII, 346 pp.

63. VAN CAYSEELE, Patrick (20/05/87)
Regulation and international innovative activities in the pharmaceutical industry. s.l., s.n., 1987. XI, 169 pp.

64. FRANCOIS, Pierre (21/09/87)
De empirische relevantie van de independence from irrelevant alternatives. Assumptie indiscrete keuzemodellen. Leuven, s.n., 1987. IX, 379 pp.

65. DECOSTER, André (23/09/88)
Family size, welfare and public policy. Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1988. XIII, 444 pp.

66. HEIJNEN, Bart (09/09/88)
Risicowijziging onder invloed van vrijstellingen en herverzekeringen: een theoretische analyse van optimaliteit en premiebepaling. Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1988. onregelmatig gepagineerd.

67. GEEROMS, Hans (14/10/88)
Belastingvermijding. Theoretische analyse van de determinanten van de belastingontduiking en de belastingontwijking met empirische verificaties. Leuven, s.n., 1988. XIII, 409, 5 pp.

68. PUT, Ferdi (19/12/88)
Introducing dynamic and temporal aspects in a conceptual (database) schema. Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1988. XVIII, 415 pp.

69. VAN ROMPUY, Guido (13/01/89)
A supply-side approach to tax reform programs. Theory and empirical evidence for Belgium. Leuven, KUL. Faculteit Economische en toegepaste economische wetenschappen, 1989. XVI, 189, 6 pp.

70. PEETERS, Ludo (19/06/89)
Een ruimtelijk evenwichtsmodel van de graanmarkten in de E.G.: empirische specificatie en beleidstoepassingen. Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1989. XVI, 412 pp.

71. PACOLET, Jozef (10/11/89)
Marktstructuur en operationele efficiëntie in de Belgische financiële sector. Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1989. XXII, 547 pp.

72. VANDEBROEK, Martina (13/12/89)
Optimalisatie van verzekeringscontracten en premieberekeningsprincipes. Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1989. 95 pp.

73. WILLEKENS, Francois (1990)
Determinance of government growth in industrialized countries with applications to Belgium. Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1990. VI, 332 pp.

74. VEUGELERS, Reinhilde (02/04/90)
Scope decisions of multinational enterprises. Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1990. V, 221 pp.

75. KESTELOOT, Katrien (18/06/90)
Essays on performance diagnosis and tacit cooperation in international oligopolies. Leuven, K.U.Leuven. Faculteit Economische en toegepaste economische wetenschappen, 1990. 227 pp.

76. WU, Changqi (23/10/90) Strategic aspects of oligopolistic vertical integration. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1990. VIII, 222 pp.

77. ZHANG, Zhaoyong (08/07/91)
A disequilibrium model of China's foreign trade behaviour. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1991. XII, 256 pp.

78. DHAENE, Jan (25/11/91)
Verdelingsfuncties, benaderingen en foutengrenzen van stochastische grootheden geassocieerd aan verzekeringspolissen en -portefeuilles. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1991. 146 pp.

79. BAUWELINCKX, Thierry (07/01/92)
Hierarchical credibility techniques. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 130 pp.

80. DEMEULEMEESTER, Erik (23/3/92)
Optimal algorithms for various classes of multiple resource-constrained project scheduling problems. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 180 pp.

81. STEENACKERS, Anna (1/10/92)
Risk analysis with the classical actuarial risk model: theoretical extensions and applications to Reinsurance. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 139 pp.

82. COCKX, Bart (24/09/92)
    The minimum income guarantee. Some views from a dynamic perspective. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. XVII, 401 pp.

83. MEYERMANS, Eric (06/11/92)
    Econometric allocation systems for the foreign exchange market: Specification, estimation and testing of transmission mechanisms under currency substitution. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. XVIII, 343 pp.

84. CHEN, Guoqing (04/12/92)
    Design of fuzzy relational databases based on fuzzy functional dependency. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1992. 176 pp.

85. CLAEYS, Christel (18/02/93)
    Vertical and horizontal category structures in consumer decision making: The nature of product hierarchies and the effect of brand typicality. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 348 pp.

86. CHEN, Shaoxiang (25/03/93)
    The optimal monitoring policies for some stochastic and dynamic production processes. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 170 pp.

87. OVERWEG, Dirk (23/04/93)
    Approximate parametric analysis and study of cost capacity management of computer configurations. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 270 pp.

88. DEWACHTER, Hans (22/06/93)
    Nonlinearities in speculative prices: The existence and persistence of nonlinearity in foreign exchange rates. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 151 pp.

89. LIN, Liangqi (05/07/93)
    Economic determinants of voluntary accounting choices for R & D expenditures in Belgium. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 192 pp.

90. DHAENE, Geert (09/07/93)
    Encompassing: formulation, properties and testing. Leuven, K.U.Leuven,

Faculteit Economische en toegepaste economische wetenschappen, 1993. 117 pp.

91. LAGAE, Wim (20/09/93)
Marktconforme verlichting van soevereine buitenlandse schuld door private crediteuren: een neo-institutionele analyse. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 241 pp.

92. VAN DE GAER, Dirk (27/09/93)
Equality of opportunity and investment in human capital. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1993. 172 pp.

93. SCHROYEN, Alfred (28/02/94)
Essays on redistributive taxation when monitoring is costly. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 203 pp. + V.

94. STEURS, Geert (15/07/94)
Spillovers and cooperation in research and development. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 266 pp.

95. BARAS, Johan (15/09/94)
The small sample distribution of the Wald, Lagrange multiplier and likelihood ratio tests for homogeneity and symmetry in demand analysis: a Monte Carlo study. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 169 pp.

96. GAEREMYNCK, Ann (08/09/94)
The use of depreciation in accounting as a signalling device. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 232 pp.

97. BETTENDORF, Leon (22/09/94)
A dynamic applied general equilibrium model for a small open economy. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 149 pp.

98. TEUNEN, Marleen (10/11/94)
Evaluation of interest randomness in actuarial quantities. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1994. 214 pp.

99. VAN OOTEGEM, Luc (17/01/95)
An economic theory of private donations. Leuven. K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 236 pp.

100. DE SCHEPPER, Ann (20/03/95)
Stochastic interest rates and the probabilistic behaviour of actuarial functions. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 211 pp.

101. LAUWERS, Luc (13/06/95)
Social choice with infinite populations. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 79 pp.

102. WU, Guang (27/06/95)
A systematic approach to object-oriented business modeling. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 248 pp.

103. WU, Xueping (21/08/95)
Term structures in the Belgian market: model estimation and pricing error analysis. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 133 pp.

104. PEPERMANS, Guido (30/08/95)
Four essays on retirement from the labor force. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 128 pp.

105. ALGOED, Koen (11/09/95)
Essays on insurance: a view from a dynamic perspective. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 136 pp.

106. DEGRYSE, Hans (10/10/95)
Essays on financial intermediation, product differentiation, and market structure. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 218 pp.

107. MEIR, Jos (05/12/95)
Het strategisch groepsconcept toegepast op de Belgische financile sector. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1995. 257 pp.

108. WIJAYA, Miryam Lilian (08/01/96)
Voluntary reciprocity as an informal social insurance mechanism: a game the-

oretic approach. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 124 pp.

109. VANDAELE, Nico (12/02/96)
The impact of lot sizing on queueing delays: multi product, multi machine models. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 243 pp.

110. GIELENS, Geert (27/02/96)
Some essays on discrete time target zones and their tails. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 131 pp.

111. GUILLAUME, Dominique (20/03/96)
Chaos, randomness and order in the foreign exchange markets. Essays on the modelling of the markets. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 171 pp.

112. DEWIT, Gerda (03/06/96)
Essays on export insurance subsidization. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 186 pp.

113. VAN DEN ACKER, Carine (08/07/96)
Belief-function theory and its application to the modeling of uncertainty in financial statement auditing. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 147 pp.

114. IMAM, Mahmood Osman (31/07/96)
Choice of IPO Flotation Methods in Belgium in an Asymmetric Information Framework and Pricing of IPO's in the Long-Run. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 221 pp.

115. NICAISE, Ides (06/09/96)
Poverty and Human Capital. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1996. 209 pp.

116. EYCKMANS, Johan (18/09/97)
On the Incentives of Nations to Join International Environmental Agreements. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1997. XV + 348 pp.

117. CRISOLOGO-MENDOZA, Lorelei (16/10/97)
Essays on Decision Making in Rural Households: a study of three villages

in the Cordillera Region of the Philippines. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1997. 256 pp.

118. DE REYCK, Bert (26/01/98)
Scheduling Projects with Generalized Precedence Relations: Exact and Heuristic Procedures. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. XXIV + 337 pp.

119. VANDEMAELE Sigrid (30/04/98)
Determinants of Issue Procedure Choice within the Context of the French IPO Market: Analysis within an Asymmetric Information Framework. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 241 pp.

120. VERGAUWEN Filip (30/04/98)
Firm Efficiency and Compensation Schemes for the Management of Innovative Activities and Knowledge Transfers. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. VIII + 175 pp.

121. LEEMANS Herlinde (29/05/98)
The Two-Class Two-Server Queueing Model with Nonpreemptive Heterogeneous Priority Structures. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 211 pp.

122. GEYSKENS Inge (4/09/98)
Trust, Satisfaction, and Equity in Marketing Channel Relationships. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 202 pp.

123. SWEENEY John (19/10/98)
Why Hold a Job ? The Labour Market Choice of the Low-Skilled. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1998. 278 pp.

124. GOEDHUYS Micheline (17/03/99)
Industrial Organisation in Developing Countries, Evidence from Côte d'Ivoire. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 251 pp.

125. POELS Geert (16/04/99)
On the Formal Aspects of the Measurement of Object-Oriented Software Specifications. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 507 pp.

126. MAYERES Inge (25/05/99)
The Control of Transport Externalities: A General Equilibrium Analysis. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. XIV + 294 pp.

127. LEMAHIEU Wilfried (5/07/99)
Improved Navigation and Maintenance through an Object-Oriented Approach to Hypermedia Modelling. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 284 pp.

128. VAN PUYENBROECK Tom (8/07/99)
Informational Aspects of Fiscal Federalism. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 192 pp.

129. VAN DEN POEL Dirk (5/08/99)
Response Modeling for Database Marketing Using Binary Classification. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 342 pp.

130. GIELENS Katrijn (27/08/99)
International Entry Decisions in the Retailing Industry: Antecedents and Performance Consequences. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 336 pp.

131. PEETERS Anneleen (16/12/99)
Labour Turnover Costs, Employment and Temporary Work. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 207 pp.

132. VANHOENACKER Jurgen (17/12/99)
Formalizing a Knowledge Management Architecture Meta-Model for Integrated Business Process Management. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 1999. 252 pp.

133. NUNES Paulo (20/03/2000)
Contingent Valuation of the Benefits of Natural Areas and its Warmglow Component. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. XXI + 282 pp.

134. VAN DEN CRUYCE Bart (7/04/2000)
Statistische discriminatie van allochtonen op jobmarkten met rigide lonen. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. XXIII + 441 pp.

135. REPKINE Alexandre (15/03/2000)
Industrial restructuring in countries of Central and Eastern Europe: Combining branch-, firm- and product-level data for a better understanding of Enterprises' behaviour during transition towards market economy. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. VI + 147 pp.

136. AKSOY, Yunus (21/06/2000)
Essays on international price rigidities and exchange rates. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. IX + 236 pp.

137. RIYANTO, Yohanes Eko (22/06/2000)
Essays on the internal and external delegation of authority in firms. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. VIII + 280 pp.

138. HUYGHEBAERT, Nancy (20/12/2000)
The Capital Structure of Business Start-ups. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2000. VIII + 332 pp.

139. FRANCKX Laurent (22/01/2001)
Ambient Inspections and Commitment in Environmental Enforcement. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VIII + 286 pp.

140. VANDILLE Guy (16/02/2001)
Essays on the Impact of Income Redistribution on Trade. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VIII + 176 pp.

141. MARQUERING Wessel (27/04/2001)
Modeling and Forecasting Stock Market Returns and Volatility. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 V + 267 pp.

142. FAGGIO Giulia (07/05/2001)
Labor Market Adjustment and Enterprise Behavior in Transition. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 150 pp.

143. GOOS Peter (30/05/2001)
The Optimal Design of Blocked and Split-plot experiments. Leuven, K.U.Leuven,

Faculteit Economische en toegepaste economische wetenschappen, 2001 X + 224 pp.

144. LABRO Eva (01/06/2001)
Total Cost of Ownership Supplier Selection based on Activity Based Costing and Mathematical Programming. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 217 pp.

145. VANHOUCKE Mario (07/06/2001)
Exact Algorithms for various Types of Project Scheduling Problems. Nonregular Objectives and time/cost Trade-offs. 316 Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 316 pp.

146. BILSEN Valentijn (28/08/2001)
Entrepreneurship and Private Sector Development in Central European Transition Countries. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 XVI + 188 pp.

147. NIJS Vincent (10/08/2001)
Essays on the dynamic Category-level Impact of Price promotions. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001

148. CHERCHYE Laurens (24/09/2001)
Topics in Non-parametric Production and Efficiency Analysis. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VII + 169 pp.

149. VAN DENDER Kurt (15/10/2001)
Aspects of Congestion Pricing for Urban Transport. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001 VII + 203 pp.

150. CAPEAU Bart (26/10/2001)
In defence of the excess demand approach to poor peasants' economic behaviour. Theory and Empirics of non-recursive agricultural household modelling. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001, XIII + 286 blz.

151. CALTHROP Edward (09/11/2001)
Essays in urban transport economics. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2001.

152. VANDER BAUWHEDE Heidi (03/12/2001)
Earnings management in an Non-Anglo-Saxon environment. Leuven, K.U.Leuven,

Faculteit Economische en toegepaste economische wetenschappen, 2001, 408 pp.

153. DE BACKER Koenraad (22/01/2002)
Multinational firms and industry dynamics in host countries : the case of Belgium. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, VII + 165 pp.

154. BOUWEN Jan (08/02/2002)
Transactive memory in operational workgroups. Concept elaboration and case study. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 319 pp. + appendix 102 pp.

155. VAN DEN BRANDE Inge (13/03/2002)
The psychological contract between employer and employee : a survey among Flemish employees. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, VIII + 470 pp.

156. VEESTRAETEN Dirk (19/04/2002)
Asset Price Dynamics under Announced Policy Switching. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 176 pp.

157. PEETERS Marc (16/05/2002)
One Dimensional Cutting and Packing : New Problems and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002

158. SKUDELNY Frauke (21/05/2002)
Essays on The Economic Consequences of the European Monetary Union. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002

159. DE WEERDT Joachim (07/06/2002)
Social Networks, Transfers and Insurance in Developing countries. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, VI + 129 pp.

160. TACK Lieven (25/06/2002)
Optimal Run Orders in Design of Experiments. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, XXXI + 344 pp.

161. POELMANS Stephan (10/07/2002)
Making Workflow Systems work. An investigation into the Importance of

Task-appropriation fit, End-user Support and other Technological Characteristics. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 237 pp.

162. JANS Raf (26/09/2002)
Capacitated Lot Sizing Problems : New Applications, Formulations and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002.

163. VIAENE Stijn (25/10/2002)
Learning to Detect Fraud from enriched Insurance Claims Data (Context, Theory and Applications). Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002, 315 pp.

164. AYALEW Tekabe (08/11/2002)
Inequality and Capital Investment in a Subsistence Economy.Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. V + 148 pp.

165. MUES Christophe (12/11/2002)
On the Use of Decision Tables and Diagrams in Knowledge Modeling and Verification. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. 222 pp.

166. BROCK Ellen (13/03/2003)
The Impact of International Trade on European Labour Markets. K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002.

167. VERMEULEN Frederic (29/11/2002)
Essays on the collective Approach to Household Labour Supply. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. XIV + 203 pp.

168. CLUDTS Stephan (11/12/2002)
Combining participation in decision-making with financial participation : theoretical and empirical perspectives. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2002. XIV + 247 pp.

169. WARZYNSKI Frederic (09/01/2003)
The dynamic effect of competition on price cost margins and innovation. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

170. VERWIMP Philip (14/01/2003)
Development and genocide in Rwanda ; a political economy analysis of peasants and power under the Habyarimana regime. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

171. BIGANO Andrea (25/02/2003)
Environmental regulation of the electricity sector in a European Market Framework. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003, XX + 310 pp.

172. MAES Konstantijn (24/03/2003)
Modeling the Term Structure of Interest Rates Across Countries. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003, V+246 pp.

173. VINAIMONT Tom (26/02/2003)
The performance of One- versus Two-Factor Models of the Term Structure of Interest Rates. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

174. OOGHE Erwin (15/04/2003)
Essays in multi-dimensional social choice. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003, VIII+108 pp.

175. FORRIER Anneleen (25/04/2003)
Temporary employment, employability and training. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

176. CARDINAELS Eddy (28/04/2003)
The role of cost system accuracy in managerial decision making. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003. 144 pp.

177. DE GOEIJ Peter (02/07/2003)
Modeling Time-Varying Volatility and Interest Rates. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003. VII+225 pp.

178. LEUS Roel (19/09/2003)
The generation of stable project plans. Complexity and exact algorithms. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

179. MARINHEIRO Carlos (23/09/2003)
EMU and fiscal stabilisation policy : the case of small countries. Leuven,

K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003

180. BAESSENS Bart (24/09/2003)
Developing intelligent systems for credit scoring using machine learning techniques. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003

181. KOCZY Laszlo (18/09/2003)
Solution concepts and outsider behaviour in coalition formation games. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003

182. ALTOMONTE Carlo (25/09/2003)
Essays on Foreign Direct Investment in transition countries : learning from the evidence. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003

183. DRIES Liesbeth (10/11/2003)
Transition, Globalisation and Sectoral Restructuring: Theory and Evidence from the Polish Agri-Food Sector. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

184. DEVOOGHT Kurt (18/11/2003)
Essays On Responsibility-Sensitive Egalitarianism and the Measurement of Income Inequality. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003

185. DELEERSNYDER Barbara (28/11/2003)
Marketing in Turbulent Times. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

186. ALI Daniel (19/12/2003)
Essays on Household Consumption and Production Decisions under Uncertainty in Rural Ethiopia. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2003.

187. WILLEMS Bert (14/01/2004)
Electricity networks and generation market power. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

188. JANSSENS Gust (30/01/2004)
Advanced Modelling of Conditional Volatility and Correlation in Financial Markets. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

189. THOEN Vincent (19/01/2004)
On the valuation and disclosure practices implemented by venture capital providers. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

190. MARTENS Jurgen (16/02/2004)
A fuzzy set and stochastic system theoretic technique to validate simulation models. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

191. ALTAVILLA Carlo (21/05/2004)
Monetary policy implementation and transmission mechanisms in the Euro area., Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

192. DE BRUYNE Karolien (07/06/2004)
Essays in the location of economic activity. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

193. ADEM Jan (25/06/2004)
Mathematical programming approaches for the supervised classification problem., Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

194. LEROUGE Davy (08/07/2004)
Predicting Product Preferences : the effect of internal and external cues., Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

195. VANDENBROECK Katleen (16/07/2004)
Essays on output growth, social learning and land allocation in agriculture : micro-evidence from Ethiopia and Tanzania. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

196. GRIMALDI Maria (03/09/2004)
The exchange rate, heterogeneity of agents and bounded rationality. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

197. SMEDTS Kristien (26/10/2004)
Financial integration in EMU in the framework of the no-arbitrage theory. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

198. KOEVOETS Wim (12/11/2004)
Essays on Unions, Wages and Employment. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

199. CALLENS Marc (22/11/2004)
Essays on multilevel logistic Regression. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

200. RUGGOO Arvind (13/12/2004)
Two stage designs robust to model uncertainty. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2004.

201. HOORELBEKE Dirk (28/01/2005)
Bootstrap and Pivoting Techniques for Testing Multiple Hypotheses. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

202. ROUSSEAU Sandra (17/02/2005)
Selecting Environmental Policy Instruments in the Presence of Incomplete Compiance. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

203. VAN DER MEULEN Sofie (17/02/2005)
Quality of Financial Statements : Impact of the external auditor and applied accounting standards. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

204. DIMOVA Ralitza (21/02/2005)
Winners and Losers during Structural Reform and Crisis : the Bulgarian Labour Market Perspective. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

205. DARKIEWICZ Grzegorz (28/02/2005)
Value-at-risk in Insurance and Finance : the Comonotonicity Approach. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

206. DE MOOR Lieven (20/05/2005)
The Structure of International Stock Returns : Size, Country and Sector Effects in Capital Asset Pricing. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

207. EVERAERT Greetje (27/06/2005)
Soft Budget Constraints and Trade Policies : The Role of Institutional

and External Constraints. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

208. SIMON Steven (06/07/2005)
The Modeling and Valuation of complex Derivatives : the Impact of the Choice of the term structure model. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.

209. MOONEN Linda (23/09/2005)
Algorithms for some Graph-Theoretical Optimization Problems. Leuven, K.U.Leuven, Faculteit Economische en toegepaste economische wetenschappen, 2005.