Alma Mater Studiorum University of Bologna

Ph.D. in Control System Engineering and Operational Research

MAT/09

XXIII Cycle

Decomposition and Reformulation of Integer Linear Programming Problems

Fabio Furini

Coordinator
Prof. Paolo Toth

Relators
Prof. Paolo Toth
Prof. Alberto Caprara

Final exam 2011

Acknowledgments

To my lovely girlfriend Charlotte Mitchell and to my family who supported me in every occasions.

Bologna, 2011

Fabio Furini.

"From each according to his ability, to each according to his need"

Critique of the Gotha Program, Karl Marx.

Keywords

Combinatorial Optimisation, Integer Programming,Decomposition and Reformulation,Column Generation, Resource Allocation Problem, AVG Assignment Problem.

Abstract

This thesis deals with an investigation of *Decomposition and Reformulation* to solve *Integer Linear Programming Problems*. This method is often a very successful approach computationally, producing high-quality solutions for well-structured combinatorial optimization problems like vehicle routing, cutting stock, p-median and generalized assignment . However, until now the method has always been tailored to the specific problem under investigation. The principal innovation of this thesis is to develop a new framework able to apply this concept to a generic MIP problem. The new approach is thus capable of auto-decomposition and auto-reformulation of the input problem applicable as a resolving black box algorithm and works as a complement and alternative to the normal resolving techniques. The idea of *Decomposing and Reformulating* (usually called in literature Dantzig and Wolfe Decomposition DWD) is, given a MIP, to convexify one (or more) subset(s) of constraints (slaves) and working on the partially convexified polyhedron(s) obtained. For a given MIP several decompositions can be defined depending from what sets of constraints we want to convexify. In this thesis we mainly reformulate MIPs using two sets of variables: the original variables and the extended variables (representing the exponential extreme points). The master constraints consist of the original constraints not included in any slaves plus the convexity constraint(s) and the linking constraints(ensuring that each original variable can be viewed as linear combination of extreme points of the slaves). The solution procedure consists of iteratively solving the reformulated MIP (master) and checking (pricing) if a variable of reduced costs exists, and in which case adding it to the master and solving it again (columns generation), or otherwise stopping the procedure. The advantage of using DWD is that the reformulated relaxation gives bounds stronger than the original LP relaxation, in addition it can be incorporated in a Branch and bound scheme (Branch and Price) in order to solve the problem to optimality. If the computational time for the pricing problem is reasonable this leads in practice to a stronger speed up in the solution time, specially when the convex hull of the slaves is easy to compute, usually because of its special structure. Finally the thesis is thus organized as follows:

Chapter 1 The *Introduction* contains a detailed overview of the thesis.

Chapter 2 We describe the general ideas and methods applying decomposition and reformulations to general problems.

Chapter 3 We introduce the idea of *Dynamic MIP* and we perform our decomposition with this class of problems.

Chapter 4 We investigate the *Multi-load AGV dispatching problem* and we present a new compact and extended formulation.

Chapter 5 We investigate the *2D Cutting Stock Problem* presenting a very effective Column Generation Heuristic.

Contents

Chapter 1

Introduction

Decision makers in industry and administration are often faced with decision problems that are too large and complex for an effective manual solution. To help them in this task, applied mathematicians in operational research have developed modeling and optimization tools. These tools are now widely used in large, medium and small industries. However, there remains a gap between the theoretical developments made on pure models on the one hand and the practical needs for complex application solving. We aim to bridge this gap by developing a software package that will automatically exploit problem structure to take advantage of the Dantzig-Wolfe decomposition principle for solving general, large, practical decision problems formulated as Mixed Integer Programmes (MIP). The principal goal of this thesis is to investigate and to develop a general framework to take advantage of the Dantzig and Wolfe Decomposition technique for solving a generic Mixed Integer Programme. This decomposition concept was introduced in a famous chaper [15] in 1960 by G. B. Dantzig and P. Wolfe and, since then, academic research has focused its attention on developing a specific algorithm called Column Generation (CG). The strength of this method lies in the decomposition of the problem into smaller sub-problems, solving them efficiently and reconstructing a global solution starting from the partial solutions, known technically as columns. The second main advantage is that not all the partial solutions, i.e. the columns, are required to prove the optimality of the entire problem but only a subset, which are generated as necessary by the general procedure. This fact immediately suggests the possibility of using the method to resolve problems of large and complex dimensions. The Dantzig-Wolfe decomposition principle of linear programming extends to integer programming optimization problems. A sub-system of constraints is isolated and the problem is reformulated in terms of the solutions of the subsystem. The resulting integer programming formulation, the so-called master, typically has a large number of columns or variables, one for each feasible solution of the subsystem. The master formulation usually has a strong linear programme relaxation which makes it well suited for a linear programming based branch and bound solution method. To deal with the large number of variables implicitly, a column generation procedure is used at each node of the branch and bound tree. The algorithm that combines column generation and branch and bound is known as branch and price. This approach to solving difficult integer programming problems is a good alternative to other advanced methods such as cutting plane, Lagrange relaxation, or variable definition approaches. Moreover, it is an integrative tool that allows one to take advantage of the latest algorithms developed for general models that appear as sub problems in solving more complex applications. Since 1960 many academic chapers and

books have demonstrated the efficiency of the Decomposition and Reformulation method for MIP, but these have all revealed a common weak point: the fact that in order to obtain an optimal performance the method has to be tailored to the specific problem in question. Our new strand of research fits into this context. Starting off from the ideas developed in the literature, we aim to identify the common points in order to develop a new framework able to apply this concept to a generic MIP problem. The new approach is capable of auto-decomposition and auto-reformulation of the input problem, and of the identification of the optimal solutions in an efficient manner. We thus developed general software for that purpose. This permits us to overcome the challenge of having to rethink a specific column generation approach from scratch for each individual problem, and thus to be able to put forward a standard technique, applicable as a resolving black box algorithm and which works as a complement and alternative to the normal resolving techniques that are currently implemented in all commercial optimization software packages.

In summary, this thesis aims to develop a general framework of Decomposition and Reformulation of generic mixed integer programming problems. To reach this challenging objective it is necessary to confront numerous difficulties. The first step is the analysis of the different aspects of the Reformulation of optimization problems. Many different standard approaches exist in the literature dealing with such problems, and a detailed description of the approaches is provided in [99] by F. Vanderbeck and L.A. Wolsey. Using their words, the aim of reformulation in general is the following:

> Introducing new variables typically permits one to model some combinatorial structure more precisely and to induce integrality through tighter linear constraints linking the variables. One such extended formulation is provided by the classical Minkowski representation of a polyhedron in terms of its extreme points and extreme rays.

The fundamental reformulation techniques can be identified in the following complementary methodologies: the so-called "convexification approach" and the alternative "discretization approach". We concentrated on the identification of the points of strength and weakness of each method to be able to propose a dominant approach in tackling the reformulation of generic problems. Moreover, these approaches can be further subdivided into approaches that make use of the original variables of the problem and approaches that use a projection of the original model in a different vector space that uses new variables instead. It is also necessary to identify the points of strength and weakness of these methods in order to identify the most efficient system. The second large field of research will focus on the decomposition of optimization problems. As far as this aspect is concerned, little has been done in the literature regarding automatic decomposition. As has been partially indicated in the previous paragraph, the algorithms proposed in the Column Generation literature have in fact been developed for specific problems, that is to say problems characterized internally by a particular structure that implicitly suggests the best decomposition to use. An example can be cited to clarify this. When confronting optimization problems which make use of different resources, for example, which are in turn differentiated by specific characteristics, the natural decomposition would consider a sub-problem for each of these resources. The present research project, however, will deal with the decomposition of generic problems in which the nature of the problem is not known a priori. We develop a software optimization package which will automatically identify in general the best decompositions. The principal problem that must be confronted will be the natural trade-off between the decompositions in which the

sub-problems include much of the original problem's information and those that maintain a large part of the original problem to the original problem instead. In general, the former lead to better reformulations but which at the same time present sub-problems that will be much more difficult to resolve. The latter, whilst keeping the resolution of sub-problems simple, lead to less efficient reformulations. We thus devoted ourselves to the definition of the criteria that generally lead to good decompositions. Once these phases of theoretical research were complete, we started the implementation of the optimization framework, which is able to test the efficiency of the new methods proposed. Algorithms are developed using C in order to obtain a benchmark comparing with current state-of-the-art commercial optimization software such as CPLEX. To test the algorithms, the instances referred to are the set of instances called MIPLIP2003, often used by major research in this field. These tests resume the main characteristics of a great variety of optimization problems and also appear to be problems that are difficult to resolve using the current primary commercial software.

To conclude this introduction, we shall give a brief description of the following chapters. In chapter 2 we present the general tool which, given an LP file, automatically detects an exploitable matrix structure, accordingly performs a Dantzig-Wolfe type reformulation of subsets of the constraints (partial convexification), and performs column generation to obtain an optimal LP relaxation. Our results strongly indicate that Dantzig-Wolfe decomposition holds more promise as a general-purpose tool than previously acknowledged by the research community. We then perform an extensive computational study on general MIPLIB2003 instances in order to further validate our method. Chapter 3 concerns the Resource Allocation Problem (RAP), an NP-hard problem which introduces a temporal dimension in classical Knapsack Problems. RAP asks for the most profitable subset of n given tasks that, at no time, exceeds the amount of usable resource. Each task has a determined profit, a start and a finish time and a resource requirement. A classical Integer Linear Programming (ILP) formulation from the literature models the problem by associating a binary variable to each task, in order to define which tasks are selected. We propose a reformulated ILP models in order to strengthen this descriptive model imposing the limit of resource usage by introducing new binary variables, each variable being associated with a feasible subset of tasks which are simultaneously active. These variables, whose number is exponential with respect to the problem size, are efficiently managed by means of a column generation approach. The generation of new variables asks for the solution of a series of knapsack problems. The continuous relaxation of this model is systematically stronger than the relaxation of the descriptive model. Finally, in order to obtain integer solutions, we embed our column generation procedures in a branching scheme, where we branch on the binary variables of the original formulation (i.e., variables associated with tasks). We conclude by presenting computational results on a set of random instances. In chapter 4 we present a column generation approach, specifically conceived for the problem, where the reformulation is very effective. Here we consider the AGV dispatching problem (AGVDP) arising at seaport container terminals where vehicles operate under a multi-load mode and can therefore carry more than one container at a time subject to capacity constraints. We propose two new approaches; the first one is based on a new mixed integer programming (MIP) model and the second one is a column generation procedure based on a set covering formulation of the AGVDP. Both methodologies are implemented and computationally evaluated for different scenarios with respect to minimizing a defined "cost" of dispatching AGVs. The reported computational results demonstrate the effectiveness of the proposed approaches in solving a number of test instances for container terminals, given various layout configurations and levels of workload. Finally in chapter 5, we consider a

Two-Dimensional Cutting Stock Problem where stock of different sizes is available, and a set of rectangular items has to be obtained through two-staged guillotine cuts. We propose a heuristic algorithm, based on column generation, which requires as subproblem the solution of a Two-Dimensional Knapsack Problem with two-staged guillotines cuts. A further contribution of the chaper consists in the definition of a Mixed Integer Linear Programming Model for the solution of this Knapsack Problem, as well as a heuristic procedure based on dynamic programming. Computational experiments show the effectiveness of the proposed approach, which obtains very small optimality gaps and outperforms the heuristic algorithm proposed by Cintra et al. [108].

Chapter 2

Partial Convexification and Dantzig-Wolfe Reformulations of General MIPs

2.1 Introduction

A considerable, if not the major, part of the computational (mixed) integer programming machinery is about outer approximating the convex hull of integer feasible points (or mixed integer sets). The addition of valid inequalities, a.k.a. cutting planes, is the traditional general-purpose device which proved powerful in strengthening the linear programming relaxations. Given that the integer hull is the ultimate object of desire, we ask: Why don't we just work with it? Being fully aware of the boldness of this question, we want to seriously reconsider it by *explicitly* constructing parts of the integer hull via a generic Dantzig-Wolfe type reformulation (decomposition). This extends previous *partial convexification* approaches which only separate a subset of facets from the integer hull.

Dantzig-Wolfe reformulation (DWR) of mixed integer programs (MIPs) became a computationally very successful—sometimes the only applicable—approach to producing high-quality solutions for well-structured combinatorial optimization problems like vehicle routing, cutting stock, p-median, generalized assignment, and many others. Their common structure is the (bordered) block-diagonal form of the coefficient matrix, the traditional realm of Dantzig-Wolfe decomposition. Be aware that so far its use is tailored to the application and far from being a general-purpose tool: It is the *user* who does not only know *that* there is an exploitable structure present but also *what* it looks like, and *how* to exploit it algorithmically. In particular in view of the automatism with which general-purpose cutting planes are separated in all serious MIP solvers, this is an unsatisfactory situation. This raises several research questions of increasing ambition:

- When the MIP contains a *known structure* suitable to Dantzig-Wolfe reformulation, can *an algorithm* detect and exploit it?
- When the contained structure is *unknown* (to be stated more precisely later), can it still be detected and exploited?
- When it is known that the MIP *does not contain* a structure amenable to DWR in the traditional sense, can DWR still be a useful computational tool?

Re-arranging matrices into particular forms is a well-known topic. However, as we will see, when there are several choices, a "good" one may not be obvious to find at all. Besides our work, we are not aware of any attempts to systematically answer the first two questions in a DWR context, and it can be taken as a fact that the research community is very inclined to answer the last, and most interesting question in the negative. In our computational study on several of the hardest MIPLIB2003 instances, we do not only suggest first attempts to accomplish the structure detection in a DWR context. We also support the DWR's potential of becoming a general-purpose method for improving dual bounds by giving surprisingly encouraging computational results. Of course, at the moment, our work is not intended to produce a competitive tool, but to provide a proof-of-concept and demonstrate that the direction is promising. The main findings of our work can be summarized as follows:

- A known or hidden double-bordered block-diagonal (so-called: *arrowhead*) matrix structure can be effectively recovered and prepared for use in DWR by a suitable use of (hyper-)graph partitioning algorithms.
- For some of the hardest MIPLIB2003 instances our reformulations produce stronger dual bounds than `CPLEX` 12.2 with default cutting planes enabled.
- We provide a general-purpose implementation which reads an LP file and performs the detection, the reformulation, and the column generation itself in order to obtain a strong LP relaxation, with only mild user interaction.

The flow of the chapteris as follows: We briefly introduce the concept of partial convexification, and the overall approach. This is then applied to general MIPs. We report on extensive computational experiments and close with a discussion of our results.

2.1.1 Partial Convexification and Dantzig-Wolfe Reformulations

Consider a MIP of the form

$$\max\{c^t x : Ax \leq b,\ Dx \leq e,\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}\ . \tag{2.1}$$

Let $P := \{x \in \mathbb{Q}^n : Dx \leq e\}$ and $P_{IP} := \text{conv}\{P \cap \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}$ denote the LP relaxation and the integer hull with respect to constraints $Dx \leq e$, respectively. We assume for ease of presentation that P is bounded. In the classical *Dantzig-Wolfe reformulation* of constraints $Dx \leq e$ we express $x \in P_{IP}$ as a convex combination of the vertices V of P_{IP}, which leads to

$$\max\{c^t x : Ax \leq b,\ x = \sum_{v \in V} \lambda_v v,\ \sum_{v \in V} \lambda_v = 1,\ \lambda \geq 0,\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}\ . \tag{2.2}$$

It is well-known that the resulting LP relaxation is potentially stronger than that of (2.1) when $P_{IP} \subsetneq P$, which is a main motivation of performing the reformulation in the first place. This *partial convexification* with respect to the constraints $Dx \leq e$ corresponds to adding (implicitly) *all* valid inequalities for P_{IP} to (2.1), which in a sense is the best one can hope for.

The reformulated MIP (2.2) has fewer constraints remaining, the so-called *master constraints* $Ax \leq b$, plus the convexity constraint and the constraints linking the *original* x variables to the *extended* λ variables. On the downside of it, in general MIP (2.2) has an exponential number of λ variables, so its LP relaxation is solved by column generation, where the pricing or *slave MIP* problem to check whether there are variables with positive reduced cost to be added to the current *master LP* problem calls for the optimization of a linear objective

function over P_{IP}. This slave MIP can either be solved by a general-purpose solver or by a tailored algorithm to exploit a specific structure, if known.

In the classical DWR setting, k disjoint sets of constraints are partially convexified, namely when the matrix D has block-diagonal form

$$D = \begin{bmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^k \end{bmatrix},$$

where $D^i \in \mathbb{Q}^{m_i \times n_i}$ for $i = 1, \ldots, k$. In other words, $Dx \leq e$ decomposes into $D^i x^i \leq e^i$ $(i = 1, \ldots, k)$, where $x = (x^1, x^2, \ldots, x^k)$, with x^i being an n_i-vector for $i = 1, \ldots, k$. Every $D^i x^i \leq e^i$ individually is partially convexified in the above spirit. We call k the *number of blocks* of the reformulation. Often enough, constraints are not separable by variable sets as above, and a double-bordered block-diagonal (or *arrowhead*) form is the most specific structure we can hope for, i.e. the constraint matrix of (2.1) looks like this

$$\begin{bmatrix} D^1 & & & & F^1 \\ & D^2 & & & F^2 \\ & & \ddots & & \vdots \\ & & & D^k & F^k \\ A^1 & A^2 & \cdots & A^k & G \end{bmatrix}.$$

The constraints associated with the rows of A^1 are called the *coupling constraints* and the variables associated with the columns of F^1 are called the *linking variables*. One can obtain a form without linking variables (and with additional linking constraints) by replacing each linking variable by one copy for each nonzero entry of the associated column and adding constraints imposing that all these copies be equal (see e.g., [118]). Then we are back to the above traditional setting.

2.1.2 Related Literature

For a general background on Dantzig-Wolfe reformulation of MIPs we refer to the recent survey [88]. The notion of *partial convexification* has been used before. For instance, Sherali et al. [82] choose small subsets of integer variables (and usually only one constraint) to directly separate cutting planes from the partial convexification P_{IP}. Our approach goes far beyond that in terms of number of constraints and variables involved in the reformulation.

There are several implementations which perform a Dantzig-Wolfe reformulation of a general MIP, and handle the resulting column generation subproblems in a generic way, like `BaPCod` [84], `DIP` [86], `G12` [85], and `GCG` [87]. In [87] it is shown that a generic reformulation algorithm performs well when a block-diagonal matrix structure is *known and given* to the algorithm. Tebboth, in his thesis [118], derived some decomposable matrix structures from the problem given in a specific modeling language. A similar approach is taken in the `G12` project. However, we do not know of a code which automatically detects a possible structure just from the matrix, that is well-suited and helpful for a Dantzig-Wolfe reformulation (or creates one by variable splitting), let alone evaluates or exploits it.

Bordered block-diagonal matrices play an important role in, e.g., numerical linear algebra. One typically tries to identify a fixed number of blocks of almost equal size with as few

constraints in the border as possible. The motivation is to prepare a matrix for parallel computation, like for solving linear equation systems, see, e.g., [89], and the references therein. We are not aware of any attempts to evaluate the quality of the re-arranged matrices in terms of suitability for DWR.

We would also like to note the following possible connection to multi-row cuts which recently received some attention. Computational experiments [79] suggest that obtaining valid inequalities from more than one row of the simplex tableau holds some potential. However, in order to use this potential it seems to be imperative to have a criterion for which rows to select. As our choice of which rows to convexify is equally important for similar reasons, the lessons we learn may help there as well.

2.2 Almost Automatic Detection of an Arrowhead Form

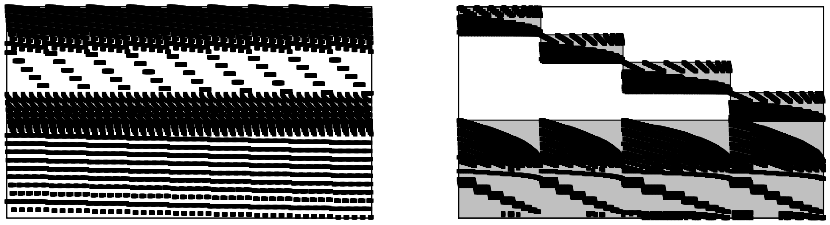
As mentioned, permuting a matrix A into arrowhead form is a common topic in numerical linear algebra. There is a folklore connection between a matrix and an associated (hyper-)graph which is also used in the graph partitioning approach by Ferris and Horn [80]. We first briefly illustrate their algorithm and then adapt it to our purpose of finding tighter LP relaxations of MIPs through DWR.

Assume the number k of blocks is given. Consider the bipartite graph G with one node r_i associated with each row i of A, one node c_j associated with each column j of A, and one edge (r_i, c_j) whenever $a_{ij} \neq 0$. Assuming the number $m + n$ of nodes is a multiple of k, find a *min k-equicut* of G, i.e. partition its node set into k subsets of equal size so that the number of edges in the cut, i.e. that join nodes in different subsets, is minimized. Finally, remove a set of nodes of G so as to ensure, for the remaining graph, that no edge joins nodes in different subsets of the partition. This yields the final arrowhead form: the row nodes removed represent the coupling constraints, the column nodes removed represent the linking variables, and the remaining row and column nodes in the ith subset of the partition define the submatrix D^i. In practice, min k-equicut is solved heuristically by using `Metis` which is an implementation of the multilevel graph partitioning algorithm in [90]. The final removal is done greedily by iteratively removing the node joined to nodes in other subsets by the largest number of edges. Moreover, to eliminate the need to require that the number of nodes be a multiple of k or to insist in having subsets of the same size, dummy nodes disconnected from the rest of the graph are added before doing the partitioning. This imposes only an upper bound on the size of each subset and on the number of subsets (note that subsets with dummy nodes only are irrelevant in the end).

We use a variant of this method because we would like to avoid the final removal phase. Specifically, we define the hypergraph H with a *node* for each nonzero entry of A. There is a *constraint hyperedge* joining all nodes for nonzero entries in the ith row of A. Moreover, there is a *variable hyperedge* joining all nodes for nonzero entries in the jth column of A. To each hyperedge we associate a cost to penalize the use of coupling constraints and linking variables.

Then, we find a heuristic min k-equicut of H, now with the objective of minimizing the sum of the costs of the hyperedges in the cut, i.e. that join nodes in different subsets of the partition. The solution already defines the arrowhead form, as the constraint hyperedges in the cut represent the coupling constraints, the variable hyperedges in the cut represent the linking variables, and the remaining constraint and variable hyperedges joining only nodes in the ith

subset of the partition define the submatrix D^i. Also in this case we use dummy nodes for allowing for unequal block sizes, 20% proved useful.



(a) original *10teams* (b) detected structure

Figure 2.1: (a) Matrix structure directly from the LP file (*10teams*) and (b) with a bordered block-diagonal structure detected by our algorithm

2.3 Computational Results

2.3.1 Notes on the Implementation and Experimental Setup

All experiments were done on Intel i7 quad-core PCs (2.67MHz, 8GB memory) running Linux 2.6.34 (single thread). As MIP and LP solver we used `CPLEX` 12.2 (single thread). Two independent jobs were run in parallel which has an impact on the CPU time of about 3% in our experiments. The min k-equicut problems to determine a reformulation (see Sect. 2.2) are heuristically solved using `Hmetis` [90].

Concerning the implementation, the overall algorithm detects an arrowhead form in the matrix, splits linking variables in order to obtain a bordered block-diagonal form, and then performs a Dantzig-Wolfe reformulation of the resulting blocks. Certain parameters must be given to the algorithm like the number k of blocks or the weights of the hyperedges in the k-equicut problem. We experimented with very few different settings and selected good decompositions instance dependent to demonstrate the concept. In the full version of this chapterwe plan to show that the whole process can be automated without any user interaction at all.

2.3.2 MIPLIB2003

In order to assess the generality of the proposed method we tested the algorithm on MIPLIB2003 instances [83]. We selected a subset of instances, for which (a) the optimum is known, (b) the density is between 0.05% and 5%, (c) the number of non-zeros is not larger than 20,000, and (d) the percentage of discrete variables is at least 20%. We are consistently capable of improving over the LP relaxation in terms of the dual bound. Moreover, on average, our DWR approach closes a larger percentage of the integrality gap than `CPLEX` with default cuts applied, see Table 2.1. We do not report on computation times because the experience is the same as for the RAP (and not the focus here): Often, we need an order of magnitude more time (even though occasionally, we are competitive with `CPLEX`).

Choosing a good decomposition We experimented with a few parameter settings for our algorithm to detect an arrowhead structure. Depending on these settings, different arrowhead forms are produced (for the same input matrix), and these perform differently in the subsequent Dantzig-Wolfe reformulation. Parameters are (a) the number k of blocks, and the penalties for hyperedges which (b) split continuous and (c) discrete variables, as well as (d) penalties for hyperedges which couple constraints. We experimented with $k \in \{2, 3, 4, 5\}$, penalized (throughout all experiments) hyperedges corresponding to continuous and discrete with cost 1 and 2, respectively; and finally we used two different settings for penalizing coupling constraints: mildly (cost 5) and extremely (cost 10^5). Every combination was applied to each instance, and the best result in terms of dual bound is reported. In other words, in this paper, we give only a proof-of-concept *that* a good decomposition can be chosen. *How* to find a good decomposition, and even the meaning of "good," are highly non-trivial issues. E.g., the "right" number k of blocks is far from obvious for instances that do not have a natural (bordered) block-diagonal form. We have preliminary computational experience that a "visually appealing" decomposition performs better than others. We give details on measures for this intuition, and on how to automate the detection and decomposition process in the full version.

Table 2.1 shows that in almost all cases the dual bound found by our DWR approach is much better than that of the continuous relaxation, and often even improves on `CPLEX`'s root node bounds with default cuts applied. The time required to compute our bound is not competitive with the time required by the general-purpose solver to solve the instance, but there remains the possibility that for some instances the significantly stronger dual bound helps in solving the instance to integer optimality.

2.4 Discussion

We have performed the first systematic computational study with an automatic partial convexification by a Dantzig-Wolfe type reformulation of subsets of rows of *arbitrary* mixed integer programs. While it is clear from theory that a partial convexification can improve the dual bound, it has not been considered a generally useful computational tool *in practice*. Thus, the most unexpected outcome of our study is that already a fairly basic implementation, combined with a careful choice of the decomposition, is actually capable of competing with or even beating a state-of-the-art MIP solver in terms of the root node dual bound. Interestingly, to the best of our knowledge for the first time, the "careful choice of the decomposition" is done almost entirely by an algorithm, only mildly helped by the user.

One should not deny that an inner approximation of the integer hull still has considerable disadvantages, as the process is not reversible: we cannot easily get rid of the extended formulation. Also, the choice of the decomposition is final (at present). This "single shot" contrasts cutting plane algorithms which can iteratively increase the number of classes of valid inequalities considered. Therefore, one must carefully decide whether to use a DWR approach or not. A remote goal would be to be able to make this important decision based on the instance only. If in, say, even only a small fraction of "all" MIPs a DWR approach pays, we would have already enriched the generic MIP toolbox.

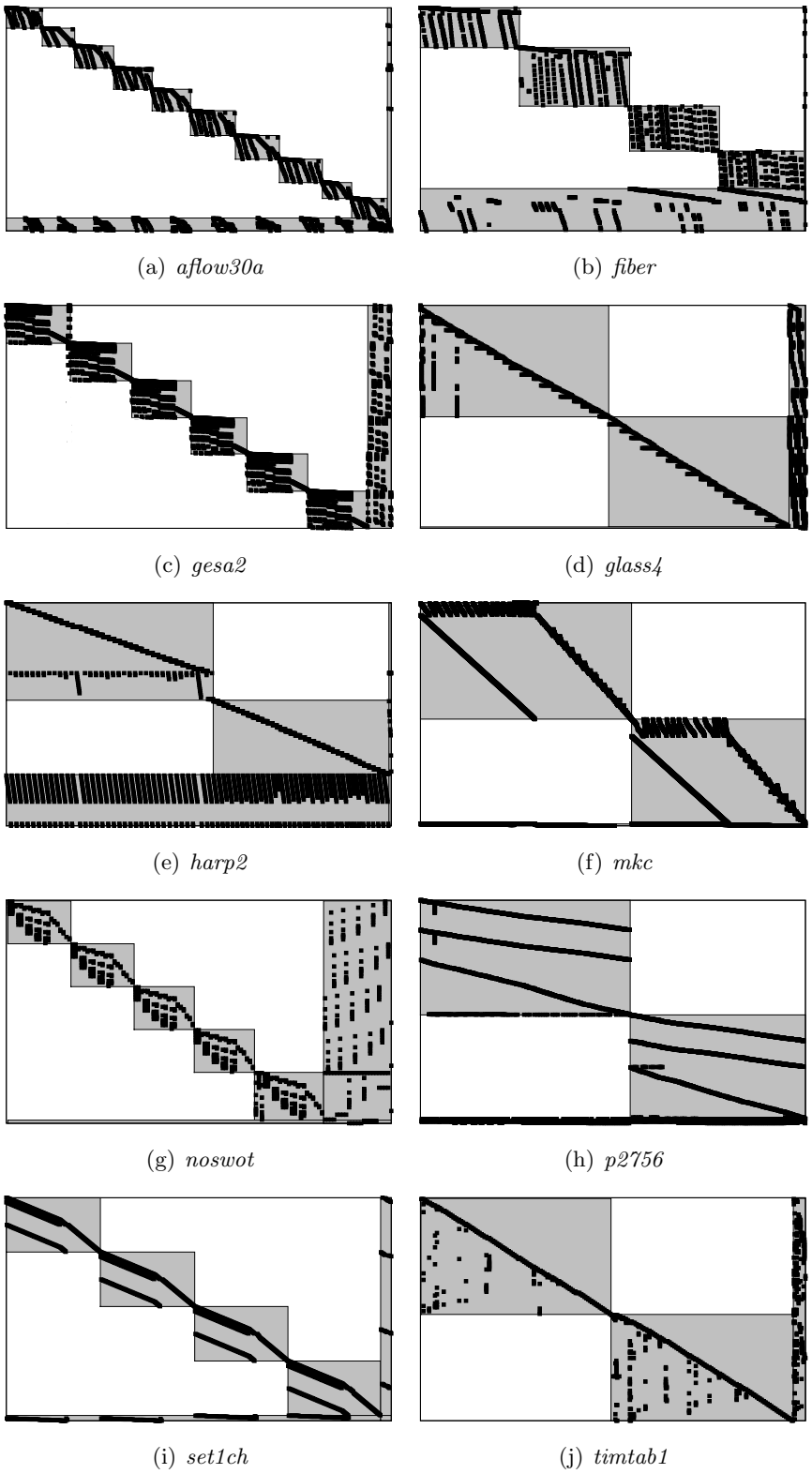
There are some possible immediate extensions concerning the implementation. Even though we have mainly solved the LP relaxation so far, our tool is able to perform a true branch-and-price. Only further experimentation can show whether the advantage in the root node can

be retained throughout the search tree, not only for dynamic MIPs but also for general MIPs (it is also conceivable that an advantage becomes visible *only* further down the tree). If one is only interested in a strong dual bound, the addition of generic cutting planes is a natural next step.

All the questions initially posed in the introduction are computationally and conceptually extremely hard, and at present one cannot hope for conclusive answers to any of them. We therefore think that our work spawns a number of interesting research directions worth pursuing further .

1. The most important task, both from a theoretical and a practical point of view, is to characterize a *good decomposition*. This can also help in quickly deciding whether it is worth trying a reformulation or not.
2. We have seen that the matrix needs not contain any (known) apparent structure in order to make the method perform well. In particular our third question from the introduction needs re-formulation in the light of our results: what does the fact that a model is suitable for application of DWR mean?
3. *Extended formulations* are a topic on its own in combinatorial optimization, mainly used as a theoretical vehicle to obtain stronger formulations. As DWR is a particular kind of extended formulation it is natural to ask: Can an approach like ours turn this into a computational tool?
4. We complained that, once chosen, a decomposition is static. Is there a computationally viable way for dynamically updating an extended formulation, like our DWR?

Taking into account that state-of-the-art solvers make successful use of cutting planes for over 15 years now, it is clear that outer approximations of the integer hull have a prominent headway in experience over inner approximations. We hope to have inspired further research and experimentation on the topic of this chapter.



(a) *aflow30a* (b) *fiber*

(c) *gesa2* (d) *glass4*

(e) *harp2* (f) *mkc*

(g) *noswot* (h) *p2756*

(i) *set1ch* (j) *timtab1*

Figure 2.2: Detected matrix structures for selected MIPLIB2003 instances

						LP	DWR		CPLEX+cuts	
instance	rows	cols	k	ℓ	c	gap	gap	%closed	gap	%closed
10teams	2025	230	4	0	107	0.758	0.000	**100.000**	0.000	**100.000**
aflow30a	842	479	2	0	28	15.098	14.700	2.634	5.353	**64.547**
aflow40b	2728	1442	5	0	39	13.899	13.899	0.000	6.471	**53.441**
fiber	1298	363	2	2	21	61.550	1.067	**98.266**	1.894	96.923
fixnet6	878	478	4	3	14	69.850	18.882	72.967	6.064	**91.318**
gesa2-o	1224	1248	5	65	0	1.177	0.000	**99.986**	0.207	82.379
gesa2	1224	1392	3	65	0	1.177	0.000	**99.986**	0.100	91.507
glass4	322	396	3	16	0	33.334	26.713	**19.862**	33.334	0.000
harp2	2993	112	5	0	39	0.614	0.614	0.000	0.371	**39.599**
manna81	3321	6480	2	78	0	1.010	0.000	**100.000**	0.000	**100.000**
mkc	5325	3411	2	0	29	8.514	0.153	**98.200**	3.778	55.625
modglob	422	291	2	18	3	1.493	0.000	**100.000**	0.142	90.480
noswot	128	182	5	21	3	4.878	0.488	**90.000**	4.878	0.000
opt1217	769	64	4	0	16	25.134	25.134	0.000	0.000	**100.000**
p2756	2756	755	4	39	13	13.932	0.269	**98.070**	7.467	46.407
pp08a	240	136	2	16	0	62.608	2.172	**96.530**	2.525	95.967
pp08aCUTS	240	246	2	16	0	25.434	2.172	**91.459**	3.823	84.968
rout	556	291	5	0	16	8.881	0.681	**92.335**	8.858	0.262
set1ch	712	492	3	20	8	41.311	2.086	94.950	0.923	**97.765**
timtab1	397	171	2	13	0	96.248	14.473	**84.963**	39.050	59.428
timtab2	675	294	4	25	0	92.377	24.426	**73.559**	46.004	50.200
tr12-30	1080	750	3	24	0	89.119	2.713	96.955	0.682	**99.235**
vpm2	378	234	2	7	0	28.078	1.706	**93.924**	6.443	77.053
arithm. mean						30.281	6.624	**74.115**	7.755	**68.570**

Table 2.1: Comparison of the dual bounds provided by our automatic DWR reformulation approach and the general-purpose MIP solver CPLEX for 23 selected instances of MIPLIB2003. Listed are the instance name, the number of constraints and variables, the number k of blocks, the number ℓ of linking variables, and number c of coupling constraints. Under the heading *LP* one finds the relative integrality gap of the LP relaxation (in percent). The relative integrality gaps of DWR and CPLEX with default cuts applied are listed under *DWR* and *CPLEX+cuts*, respectively. The percentage of the LP gap closed is given under *%closed* for both approaches. The last row lists arithmetic means of the columns.

Chapter 3

Application: Temporal Knapsack Problem

3.1 A Class of Structured MIPs without Block-Diagonal Form

We consider a class of problems which has a natural MIP formulation with a coefficient matrix which is very structured, but not completely bordered block-diagonal. A *dynamic MIP* extends a MIP

$$\max\{c^t x : Ax \leq b,\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}$$

by a time horizon $[0, T]$ and a time interval $[o_j, f_j] \subseteq [0, T]$ in which each variable x_j is *active*. In the dynamic MIP all constraints must be satisfied at any instant in the time horizon restricting attention to the variables that are active at that instant (with the other variables fixed to 0). It is simple to observe that the check can be restricted to the instants $I := \{o_1, \ldots, o_n, f_1, \ldots, f_n\}$ in which a variable becomes active or inactive, yielding the following MIP formulation of the dynamic MIP:

$$\max\{c^t x : A^i x \leq b\ (i \in I),\ x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}, \tag{3.1}$$

where A^i is obtained from A by setting to 0 all the entries of columns associated with variables not active at $i \in I$.

A simple example of a dynamic MIP is the *dynamic 0-1 knapsack* problem, in which $q = 0$ and $Ax \leq b$ takes the form $a^t x \leq b,\ 0 \leq x \leq 1$ for some $a \in \mathbb{Q}^n_+$ and $b \in \mathbb{Q}_+$. Namely, we have a set of items each with a size a_j and active for time interval $[o_j, f_j]$ and a container of capacity b and we must select a set of items to pack in the container (for the whole interval in which they are active) so that the total size packed in each instant is at most b. This problem is also known as the *temporal knapsack problem*, *resource allocation* or *unsplittable flow on a line*. The latter name is due to the fact that the time dimension may in fact be a (one-dimensional) spatial dimension, as is the case in the following real-world application that we encountered in railway service design [92]. We are given a railway corridor with m stations $1, \ldots, m$ and n railcars, the jth having weight a_j and to be transported from station o_j to station f_j, gaining profit c_j in this case. Moreover, we have a train that travels from station 1 to station m and can carry at the same time railcars for a total weight b. The problem is to decide which railcars the train carries in order to maximize the profit, and is clearly a dynamic 0-1 knapsack problem.

Although the dynamic 0-1 knapsack problem has been widely studied in the literature, the most basic questions on its complexity are still open, namely it is not known whether it is strongly NP-hard (being trivially weakly NP-hard as it generalizes the 0-1 knapsack problem) or if it has a polynomial-time approximation scheme.

3.2 Problem Definition

The *Temporal Knapsack Problem* (TKP) has a set of tasks which share a limited resource. Each task, if selected, consumes a given amount of the resource during a specified time interval, and produces a given profit. The problem asks for the selection of the subset of tasks which maximizes the profit without exceeding the available resource.

Formally, we are given a set of tasks $j = 1, \ldots, n$, each task has a starting time o_j, an ending time f_j and profit p_j. A quantity w_j of resource is consumed by task j during the interval $[o_j, f_j)$. The total resource available is C. The problem asks for the selection of the subset of tasks which maximizes the profit without exceeding the available resource at any time. All the problem data are positive integers. The TKP is a natural generalization of the *Knapsack Problem*, which arises when a time instant is specified.

Following Calinescu et al. [94], we say that a task z is active at an instant t if $o_z \leq t < f_z$, and for each z we define T_z to be the set of tasks that are active at time t_z. A straightforward ILP model for the TKP, where variable x_j is 1 if task j is selected, is [94]:

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.2}$$

$$\sum_{j \in T_z} w_j x_j \leq C \quad z = 1, \ldots, n \tag{3.3}$$

$$x_i \in \{0, 1\} \quad i = 1, \ldots, n \tag{3.4}$$

where a constraint is defined for every starting instant of a task. However this model can produce redundant dominated constraints, i.e., constraints of the form:

$$\sum_{j \in \overline{K}} w_j x_j \leq C \tag{3.5}$$

when a constraint exists in the model:

$$\sum_{j \in K} w_j x_j \leq C \tag{3.6}$$

with $\overline{K} \subset K$. In order to have only non-dominated constraints, a constraint for an instant s_i must be considered only when the next event which takes place in a subsequent instant is the termination at t_j of a task j. From now on, we will consider ILP formulations of the TKP where the dominated constraints of type (3.5) are removed. In particular, we will have $m < n$ knapsack constraints, i.e.:

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.7}$$

$$\sum_{j \in G_i} w_j x_j \leq C \qquad i = 1, ..., m \tag{3.8}$$

$$x_j \in \{0, 1\} \quad j = 1, ..., n \tag{3.9}$$

where G_i is the index set of the variables which appear in constraint i with a non-zero coefficient. In this chaper we will denote model (3.7) - (3.9) as $TRAD - TKP$, to distinguish it from the reformulations that we are going to propose. This model has multiple knapsack constraints with right-hand-side equal to C and the following structure:

- The weight of a task i is w_i in all the constraints where i appears with a non-zero coefficient;
- the non-zero coefficients of a given column appear in consecutive rows.

Thus, the TKP is a special case of the *Multiple Knapsack Problem*, which is a knapsack problem with more than one constraint and non-negative integer weights and capacity. The continuous relaxation and the dual of the $TRAD - TKP$ read as follows:

Continuous relaxation of the Primal Model:

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.10}$$

$$\sum_{j \in G_i} w_j x_j \leq C \qquad i = 1, \ldots, m \quad (\eta_i) \tag{3.11}$$

$$x_j \leq 1 \qquad j = 1, \ldots, n \quad (\iota_j) \tag{3.12}$$

$$x_j \geq 0 \quad j = 1, \ldots, n \tag{3.13}$$

Dual Model:

$$min \sum_{i=1}^{m} C\eta_i + \sum_{j=1}^{n} \iota_j \tag{3.14}$$

$$\sum_{i: j \in G_i} w_j \eta_i + \iota_j \geq p_j \qquad j = 1, ...n \tag{3.15}$$

$$\eta_i \geq 0 \quad i = 1, ...m \tag{3.16}$$

$$\iota_j \geq 0 \quad j = 1, ...n \tag{3.17}$$

The dual minimizes a total cost given by the constraints selected multiplied by the capacity C plus the dual variables ι_j. The constraints ensure that for each item j the summation of

the η_i variables s.t. $i : j \in G_i$ multiplied by item weight plus the correspondent ι_j is at least equal to the item profit p_j. Algorithm (1) describes the procedure needed to find the Maximal Knapsack Constraints, the complexity is $O(n^2)$.

Algorithm 1 Procedure to find Maximal Knapsack Constraints

Input:
ITEMS= vector of the items arranged by beginning time non decreasing
o_j= beginning time of item j,f_j=finishing time of item j, for j = 1,....,n
Output:
m = number of Maximal Constraints, G_i for i = 1,....,m

```
m = 0,check = -1
for  k = 1 to n do
  for z = 1 to n do
    time= o_z;
    if o_z < f_k then
      time = o_z;
    else
      break;
    end if
  end for
  if time != check then
    I_m = time; m++;
  end if
  check = time;
end for
for k = 1 to m do
  g=0;
  for z = 1 to n do
    if (o_z ≤ I_k) AND (f_z > I_k) then
      G_{k,g}= ITEMS_z; g++;
    end if
  end for
end for
```

The TKP and some closely related problems have been considered in the literature with various names. Computational approaches can be found in Hall and Magazine [97], who consider a generalization of the TKP problem where tasks have a fixed duration but the starting time can vary in a specified interval. This models the scheduling of tasks during a space mission, where each task can take place only during a specified time interval. They propose some heuristic algorithms and upper bounds, which are integrated in an exact approach, and tested on a set of randomly generated instances with 50 to 200 tasks. The proposed exact approach can consistently solve instances with up to 50 tasks.

Barlett et al. [95] consider a generalization of the TKP where the dimension of the Knapsack (i.e., the capacity C), is not fixed but varies over time. The authors propose an algorithm which integrates search tree techniques with cuts generation. The algorithm is experimentally tested on small instances with fixed C (i.e., TKP instances) and compared with the solution

of model (3.2)–(3.4) by means of the general purpose ILP solver CPLEX 8.1.

As far as computational complexity is concerned, the TKP is clearly NP-hard since it generalizes the classical 0-1 Knapsack problem, while the question of whether a PTAS for TKP exists is still open (see [94]).

Arkin and Sivelsberg [96] consider the problem of scheduling jobs with predefined starting and ending times on C identical machines, by maximizing the value of the completed jobs. This corresponds to a TKP where all the weights are identical: the problem is trivially polynomially solvable because the constraint matrix of the ILP model (3.2)–(3.4) is, in this case, totally unimodular; the authors also give a $O(n^2 \log n)$ algorithm. They also prove that the scheduling problem with identical weights and a subset of machines associated with each job (i.e., for each job a subset of machines where it can be scheduled is given) remains NP-hard, and they give a $O(n^{C+1})$ algorithm for the case of a fixed number of machines C.

Chen et al [98] refer to the TKP as *Bandwidth Allocation* and sketch a possible dynamic programming approach which is closely related to the $O(n^{C+1})$ algorithm in [96]. In addition, they propose a polynomial-time approximation algorithm for the special case of TKP where the profit of an item i is proportional to $(t_i - s_i)w_i$.

Calinescu et al [94] show that the TKP, which they call *Resource Allocation Problem*, can be $(1/2 - \epsilon)$-approximated in polynomial time with a randomized algorithm, while recently Darmann et al. [93] discuss some generalizations of the problem, review complexity results for special cases of TKP and prove that the problem remains NP-hard for the case where all the items' profits are identical. In addition, they give a deterministic polynomial time $(1/2 - \epsilon)$-approximated algorithm for the special case where the tasks' start and end times define a proper interval graph (i.e., no interval of activity of a task is contained in another interval of activity).

Example

Table 3.1 contains the data for the Temporal Knapsack Example 1, then figure 3.1 and figure 3.2 show the interval graph representation of that example. An interval graph is the intersection graph of a multiset of intervals on the real line. It has one vertex for each interval in the set, and an edge between every pair of vertices corresponding to intervals that intersect. These graphs are useful in modeling temporal knapsack problems, each interval represents a request for a resource for a specific period of time. In figure 3.1 the ellipses represent the sets T of model (3.2) - (3.4) while in figure 3.2 they represent the sets G of model (3.7) - (3.9). Tables 3.5 and 3.3 show the objective functions and the constraints of models, respectively.

Task	**Profit**	**Weight**	**Start**	**Finish**
$Item_1$	1	2	1	3
$Item_2$	2	2	2	14
$Item_3$	1	3	5	10
$Item_4$	3	2	7	8
$Item_5$	4	1	12	13
C=5				

Table 3.1: Data example 1

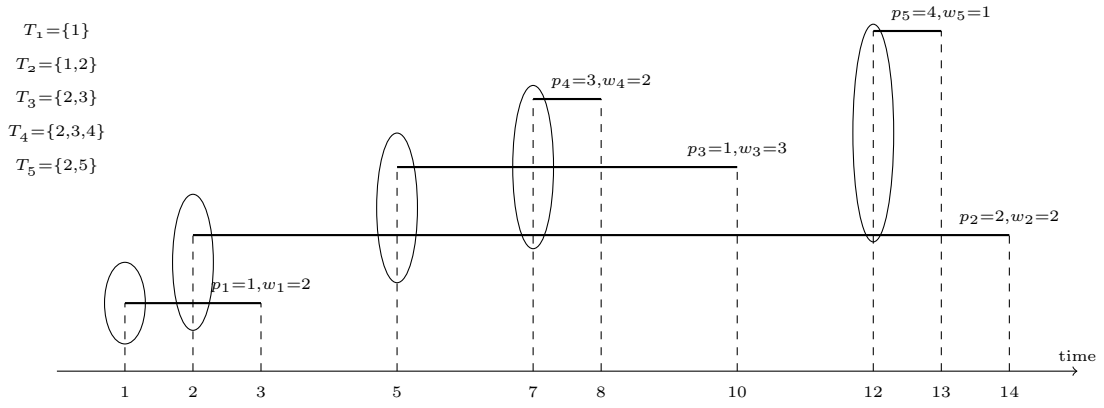


Figure 3.1: Interval Graph Example 1

O.F.	$1x_1$	$+2x_2$	$+1x_3$	$+3x_4$	$+4x_5$	
S.T.	$2x_1$					≤ 5
	$2x_1$	$+2x_2$				≤ 5
		$+2x_2$	$+3x_3$			≤ 5
		$+2x_2$	$+3x_3$	$+2x_4$		≤ 5
		$+2x_2$			$+1x_5$	≤ 5

Table 3.2: Example 1 Model (3.2) - (3.4)

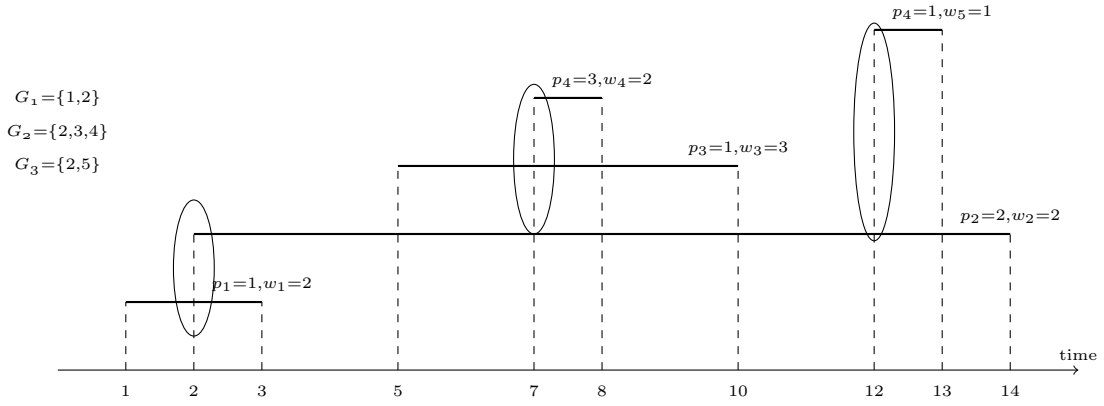


Figure 3.2: Interval Graph Example 1 "Maximal Constraints"

O.F	$1x_1$	$+2x_2$	$+1x_3$	$+3x_4$	$+4x_5$	
S.T.	$2x_1$	$+2x_2$				≤ 5
		$+2x_2$	$+3x_3$	$+2x_4$		≤ 5
		$+2x_2$			$+1x_5$	≤ 5

Table 3.3: Example 1 Model (3.7) - (3.9)

3.3 Dynamic Programming

Dynamic Programming Algorithm 1

The first Dynamic Programming Algorithm is the extension of the dynamic programming for KP01 and it has an overall complexity is $O(n(c+1)^h)$. To initialize the algorithm we need to compute F^g for $g = 0, \ldots, (C+1)^h$ which represent all the possible fillings of the h knapsacks. In our case however we always have n updates, one for each item, but the data structure must be suitable to contain all the possible configurations of h knapsacks of capacity C. Algorithm 2 shows the pseudo code of the procedure.

Algorithm 2 Dynamic Programming Algorithm 1

Input:
C, (p_j, w_j) for $j = 1, \ldots, n$ G_i for $i = 1, \ldots, h$
F^g for $g = 0, \ldots, (C+1)^h$
Output:
J($\{C\}^h$) = optimal subset
$f(\{C\}^h)$ = optimal profit

```
for g = (C + 1)^h − 1 down to 0 do
  f(F^g) = 0;
  J(F^g) = {};
end for
for j = 1 to n do
  for g = (C + 1)^h − 1 down to 0 do
    for i = 1 to h do
      if j ∈ G_i then
        F*_i = F^g_i − w_j;
        if F*_i < 0 then
          break;
        end if
      end if
      if f(F*) + p_j > f(F^g) then
        f(F^g) = f(F*) + p_j;
        J(F^g) = J(F*) ∪ {j};
      end if
    end for
  end for
end for
```

This algorithm is very effective for instances characterized by a low value of capacity C, but the computational complexity grows exponentially in the number of knapsacks h. The number of items simultaneously present in each knapsack does not affect the complexity, while the only important thing is the absolute number of items n.

Dynamic Programming Algorithm 2

Arkin and Silverberg [96] give an $O(n^{C+1})$ algorithm for scheduling jobs on C machines where a job can only be processed by a subset of the available C machines. The algorithm is polynomial when the number of machines C is fixed. The key ingredient of the algorithm is the fact that in every instant at most C jobs can be selected, thus giving $O(n^C)$ possible job configurations (per instant). The idea is used to build a polynomial-size acyclic digraph where paths correspond to feasible selections of items.

The same observation is valid for the TKP with regard to the polynomial number of job combinations which can be selected at a given instant. This is used in [98] to sketch a dynamic programming algorithm and prove that the TKP is polynomially solvable for fixed C.

We follow instead the line of [96] and propose an algorithm which is based on an acyclic digraph representation. Let the amount of resource C be fixed, and consider the formulation $TRAD - TKP$. We define a directed graph $\Gamma = (V, A)$ with a layer for each subset G_i. We now introduce the sets $E_{i-1,i}$ which represent the common items between G_{i-1} and G_i. The first set $E_{0,1}$ is the empty set. Given a layer, there is a node for each feasible combination of tasks that belong to the set $G_i \ / E_{i-1,i}$, i.e. new active items at the constraint associated with the layer. In other words, each node is thus associated with a constraint, a list of active tasks and the corresponding amount of used resource. There are m of such layers, plus an initial layer σ and a final layer τ with only one node, when no task is active.
For each layer, there are at most $2^{|G_i \setminus E_{i-1,i}|}$ subsets of active tasks, so there are at most $O(2^{|G_i \setminus E_{i-1,i}|})$ nodes at layer i, i.e., there are at most $O(2^C)$ nodes per layer (each items uses at least one unit of resource), and $O(m2^C)$ nodes in the whole graph. Now let graph Γ be constructed up to layer $i-1$. We connect the node s of layer i with the nodes $\underline{s}$ of the preceding layer $i-1$ if and only if $s \cup E_{i-1,i}$ is equal to $(\underline{s} \cap E_{i-1,i}) \cup s$. We set the arch profit equal to the profit of the items in s. According to this construction each node has at least one exiting arch and one entering arch, probably more than one. Finally the source node σ is linked to all the nodes of the first layer and all the nodes of the last layer are linked to the sink node τ. It is easy to demonstrate that the number of arches in this graph is $O(m2^{2|G_r/E_{r-1,r}|})$ and a path of maximum length for Γ, from the super source σ to the super sink τ, represents an optimal solution of the associated Temporal Knapsack Problem. It is important to underline that although the graph size is polynomial for fixed C, this can be unnecessarily large in practice. This idea works when the item number simultaneously active is low ($\leq$25) in all the constraints or when we have an high number of items active in many subsets G.

The pseudo code 3 formally describes an efficient way of computing the path of maximum length for Γ, i.e. the optimal solution of the Temporal Knapsack Problem. You update two data structures, f and J, of variable dimension according to the number of feasible subsets of each layer, m times. Each element of the first one, let's say $f_{i,k}$, stores the value of the best path up to each feasible subset k of layer i. Each element of the second one, let's say $J_{i,k}$, stores the optimal item subset respectively. In the final layer m, you can find the optimal solution scanning the entire data structure f_h, i.e. identifying the optimal value $f_{h,opt}$ and the correspondent optimal item subset $J_{h,opt}$.

Algorithm 3 Dynamic Programming Algorithm 2

Input:
C, (p_j, w_j) for j = 1,....,n G_i for i = 1,....,m
$E_{i-1,i}$ for i = 1,....,m
$S_i = \{s \subseteq G_i/E_{i-1,i} : \text{weight}(s) \leq C\}$ for i = 1,....,m
Output:
$J_{h,opt}$ = optimal subset
$f_{h,opt}$ = optimal profit

for g = 1 **to** g = $2^{|S_1|}$ **do**
  $f_{1,g} = profit(s_{1,g})$;
  $J_{1,g} = s_{1,g}$;
end for
for j = 2 **to** m **do**
  for k=1 **to** k=$2^{|S_{j-1}|}$ **do**
    for g=1 to g=$2^{|G_j/E_{j-1,j}|}$ **do**
      $s^* = (s_{j-1,k} \cap E_{j-1,j}\ ) \cup s_{j,g}$;
      if weight(s^*) $\leq$ C **then**
        if $((profit(s_{j,g}) + f_{j-1,k}) \geq f_{j,g})$ **then**
          $J_{j,g} = J(s_{j-1,k}) \cup s_{j,g}$;
          $f_{j,g} = profit(s_{j,g}) + f_{j-1,k}$;
        end if
      end if
    end for
  end for
end for
opt = index of the best subset

The figure 3.3 gives a sketch of the behavior of this algorithm related to the data of example 1. In that case we have 3 subsets G, i.e { 1,2 } { 2,3,4 } { 2,5 }. Each node of the graph is divided in 3 parts: the item subset, the correspondent weight and the the profit of the best path up to the current node. That profit is the profit of the current subset plus the profit of the best father node minus the profits of the common elements between the two nodes. In this figure we enumerated all the subsets of each G to make it simpler to understand while, as shown in the pseudo code of the dynamic programming algorithm 3, it is sufficient to deal with the subset of items which are in G_i and not in G_{i-1}. It is important to notice that subsets with a weight higher than the capacity C are deleted; for instance subset { 2,3,4 } of G_2 which has a weight of 7 has been deleted because the capacity C is only 5. In this example the optimal final node is { 2,5 } The cumulative profits is 10, which represents also the optimal solution value. The optimal path of the node { 2,5 } is { } - { 1,2} - { 2,4 } and finally { 2,5 }. Finally it is important to stress that the implementation of the algorithm requires storing just two layers of subsets each time, in that the optimal path up to the nodes is stored in the profit value.

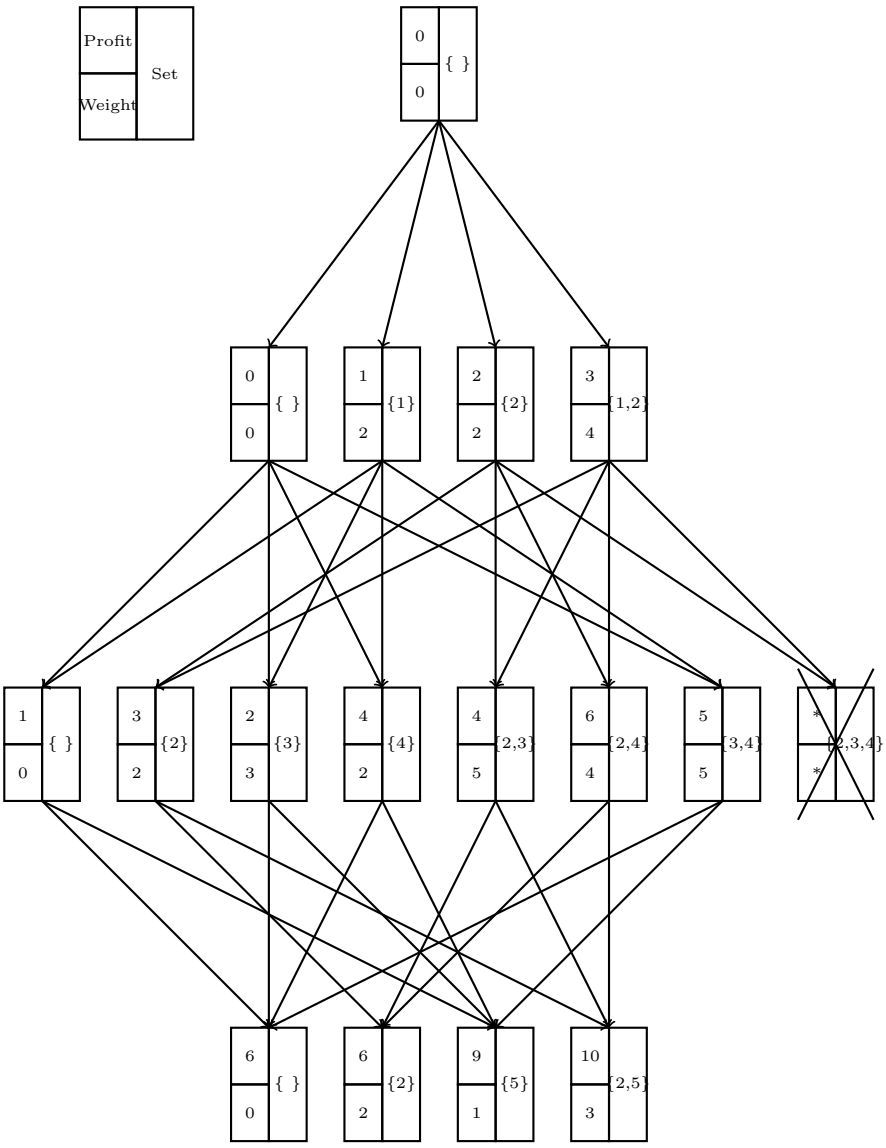


Figure 3.3: Dynamic Programming Algorithm 2 example

3.4 A Pseudo-Compact Reformulation

In this section we give a reformulation of model $TRAD-TKP$ (3.7) - (3.9), whose continuous relaxation produces stronger upper bounds, thus being more suitable for the exact solution of the TKP with a general purpose ILP solver. Although the reformulation is discussed for the case of TKP, it is general enough to be applied to any (mixed) Integer Linear Program. However, the structure of the constraint matrix of $TRAD-TKP$ (discussed in Section 3.2) makes the reformulation particularly effective. In Section 3.4.1 we review the "traditional" procedure for deriving valid cuts for an ILP, and compare it with the proposed reformulation, which is presented in Section 3.4.2. The same reformulation is interpreted in a Dantzig-Wolfe fashion in Section 3.4.3.

3.4.1 Cut Generation

Consider a generic subset of the constraints (3.8), call it h and let $I_h \subset \{1, \ldots, m\}$ be the index set of the constraints in h. We denote by G_h the index set of the variables which appear in at least one of the constraints in h with a non-zero coefficient. We consider the *exact separation* over the convex hull $P_I(h)$ of the vectors, restricted to variables in G_h, satisfying these constraints and the integrality requirements (which is bounded for the TKP):

$$P_I(h) := \text{conv}\{x : \sum_{j \in G_i} w_j x_j \leq C,\ i \in I_h\ x_j \in \{0,1\}, j \in G_h\} \tag{3.18}$$

Let $z_h^1, \ldots, z_h^p$ be the vertices of $P_I(h)$, where $z_{h,j}^k$ is the value of the j-th component in the k-th vertex.

Given a fractional solution x^* to model $TRAD{-}TKP$, the "traditional" procedure for deriving a cut separating x^* and which is valid for $P_I(h)$, i.e. which is valid for all vertices $z_h^1, \ldots, z_h^p$, is by solving the following separating LP, where we restrict the variables not in G_h to be zero:

$$\max \sum_{j \in G_h} x_j^* \alpha_j - \beta \qquad (SEP) \tag{3.19}$$

$$\sum_{j \in G_h} z_{h,j}^k \alpha_j \leq \beta \quad k = 1, \ldots, p \tag{3.20}$$

$$0 \leq \alpha_j \leq 1 \qquad j \in G_h \tag{3.21}$$

$$0 \leq \beta \tag{3.22}$$

where (3.21) is a normalization condition on the values of α. Model SEP has exponentially many constraints, one for each vertex of $P_I(h)$, which can be managed by separation as well. Given an optimal solution α^*, β^* to model SEP where only a subset of the constraints (3.20) are imposed, we look for a violated constraint (if one exists), i.e., for a vertex z_h^k such that:

$$\sum_{j \in G_h} z_{h,j}^k \alpha_j^* > \beta^* \tag{3.23}$$

The vertex can be obtained by introducing binary variables z_j, $j \in G_h$, and solving the following ILP:

$$\max \sum_{j \in G_h} \alpha_j^* z_j \quad sep(SEP) \tag{3.24}$$

$$\sum_{j \in G_i} w_j z_j \leq C \qquad i \in I_h \tag{3.25}$$

$$x_j \in \{0,1\} \qquad j \in G_h \tag{3.26}$$

If a violated constraint exist, it is added to model SEP and the procedure is iterated. Otherwise, model SEP provides a valid inequality for model $TRAD - TKP$ which cuts the current fractional solution x^*.

3.4.2 Reformulation and Column Generation

An alternative to explicit cut generation is to impose directly in model $TRAD - TKP$ that all the valid inequalities for $P_I(h)$ are satisfied. This is equivalent to asking that no valid inequality exists in model SEP.

To enforce this condition, we drop the normalization condition on α_j, $j \in G_h$ in model SEP and obtain a (potentially) unbounded problem. The dual of this model reads:

$$\min\ 0 \qquad D(SEP) \tag{3.27}$$

$$\sum_{k=1}^{p} z_{h,j}^k y_h^k = x_j^* \qquad j \in G_h \tag{3.28}$$

$$\sum_{k=1}^{p} y_h^k = 1 \tag{3.29}$$

$$y_h^k \geq 0 \quad k = 1, \ldots, p \tag{3.30}$$

$$0 \leq \alpha_j \qquad j \in G_h \tag{3.31}$$

$$0 \leq \beta \tag{3.32}$$

Model $D(SEP)$ is feasible if and only if all valid inequalities for $P_I(M)$ are satisfied by x^*, otherwise, when $D(SEP)$ is infeasible, SEP is unbounded, i.e., a valid inequality exists for $P_I(h)$ which is violated by x^*.

Thus, we can impose the feasibility of $D(SEP)$ directly in $TRAD-TKP$ by using the binary variables $y_h^1, \ldots, y_h^p$ associated with the vertices $z_h^1, \ldots, z_h^p$, and obtain a reformulated model where the constraints of subset h can be (eventually) removed:

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.33}$$

$$\sum_{j \in G_i} w_j x_j \leq C \qquad i \notin I_h \tag{3.34}$$

$$\sum_{k=1}^{p} z_{h,j}^k y_h^k = x_j \qquad j \in G_h \tag{3.35}$$

$$\sum_{k=1}^{p} y_h^k = 1 \tag{3.36}$$

$$y_h^k \in \{0,1\} \quad k = 1, \ldots, p \tag{3.37}$$

$$x_j \in \{0,1\} \quad j = 1, \ldots, n \tag{3.38}$$

The same procedure can be applied to more than one subset of constraints, say to g subsets, so as to cover all the original m constraints. In other words, we have $\bigcup_{h=1,\ldots,g} I_h = \{1, \ldots, m\}$, and obtain the following reformulated Temporal Knapsack Problem $REF-TKP$:

$$\max \sum_{j=1}^{n} p_j x_j \qquad REF-TKP, \tag{3.39}$$

$$\sum_{k=1}^{p} z_{h,j}^k y_h^k = x_j \quad h = 1, \ldots, g,\ j \in G_h \tag{3.40}$$

$$\sum_{k=1}^{p} y_h^k = 1 \qquad h = 1, \ldots, g \tag{3.41}$$

$$y_h^k \in \{0,1\} \qquad k = 1, \ldots, p \tag{3.42}$$

$$x_j \in \{0,1\} \qquad j = 1, \ldots, n \tag{3.43}$$

In model $REF-TKP$ there is a binary variable $y_h^1,\dots,y_h^p$ for every vertex $z_h^1,\dots,z_h^p$ of the convex hull $P_I(h)$ defined by every subset of constraints h. Thus, for every subset of constraints h there are exponentially many y_h variables.

In model $REF-TKP$, every x_j variable is linked by a constraint (3.40) to the y_h variables corresponding to a subset of constraints h where x_j appears with a non-zero coefficient (i.e., such that $j \in G_h$). For a general (mixed)-ILP model the number of such constraints can be very large: if $j \in G_h$ for all $h = 1,\dots,g$, we have ng additional constraints, which can make $REF-TKP$ not easily tractable. On the other hand, for problems with a "quasi" block-diagonal structure like the TKP, the reformulation is very effective (see the computational results of Section 3.6) and the number of additional constraints remains limited, because the non-zero coefficients of each column of the constraint matrix appear in consecutive rows.

Branch and Price

In order to manage the exponentially many y_h variables, column generation techniques can be used. Given a restricted master problem $REF-TKP$ which can be initialized by all the x variables and a subset of the exponentially many y_h, $h = 1,\dots,h$ variables, the column generation scheme necessitates the generation of a column y_h associated with constraints of group h and which has positive reduced profit with respect to the current solution of the restricted master. Let us call $\alpha_{h,j}$ the dual variables associated with the constraints (3.40) and β_h the dual variables associated with the constraints 3.41, and α_h^*, β_h^* the current optimal solution of the restricted master problem $REF-TKP$. The column generation problem (slave problem) necessitates finding a subset of items in G_h which satisfies constraints of subset h and has maximum profit with respect to the values $\alpha_{h,j}^*$. This is exactly the same as the $sep(SEP)$ problem of model (3.24)–(3.26) discussed in Section 3.4.1.

Once the optimal solution of the continuous relaxation of $REF-TKP$ is obtained, one may need to branch to obtain an integer solution. Note that in this case it is quite straightforward to embed the column generation scheme in a full Branch-and-Price approach, by performing a binary branching on the x variables.

3.4.3 An alternative view through Dantzig-Wolfe Decomposition

The reformulation discussed in the previous Section can be interpreted in terms of Danzing-Wolfe decomposition (we refer the reader to the recent chapter by Vanderbeck and Wolsey [99] for an introduction to Danzing-Wolfe and to the notation adopted in this Section). Consider an instance of the $TRAD-TKP$ and partition the capacity constraints in g subsets, as previously discussed. Consider two groups of constraints h and l and let j be an item such that $j \in G_h$ and $j \in G_l$ (i.e., j appears in at least one of the constraints of group h (resp. l) with non-zero coefficient). Now introduce two new variables to replace x_j: x_j^h for the constraints of group h and x_j^l for the constraints of group l, and let $\pi_j = \pi_j^h = \pi_j^l = p_j/\rho_j$ be the profits of the new variables, where p_j is the original profit of item j and ρ_j the number of groups of constraints where j appears with non-zero coefficient. Make the substitution and add a linking constraint $x_j^h = x_j^l$ for every j and every two groups of constraints h and l such that $j \in G_h$, $j \in G_l$. Let $\overline{n}$ be the number of variables after the replacement of variables. The obtained (equivalent) model has a set of (new) linking constraints (let $dx = 0$ be the i-th linking-constraints, where d in a vector of size $\overline{n}$ with $d_{j^h} = 1$, $d_{j^l} = -1$, and $d_q = 0, q \neq j^h, q \neq j^l$) plus the original knapsack constraints (with renamed variables). The

capacity constraints now have a block-diagonal structure.

Let z_h^k be the k-th integer solution of the subproblem defined by the constraints of group h, $z_{h,j}^k$ the j-component of z_h^k, and p_h the number of such solutions. We can now apply the Dantzig-Wolfe *discretization approach* for the case of problems with a block-diagonal structure, where a binary variable λ_h^k is associated with z_h^k, the k-th integer solution satisfying the constraints of group h. Thus, any feasible solution x to the $TRAD-TKP$ can be written as $x = \sum_{h=1,\dots,g} \sum_{k=1,\dots,p_h} z_h^k \lambda_h^k$. We get the following reformulation of the linking constraints:

$$\sum_{k=1,\dots,p_h} (dz_h^k)\lambda_h^k + \sum_{k=1,\dots,p_l} (dz_l^k)\lambda_l^k = 0 \tag{3.44}$$

where $Dz_h^k = 1$ if and only if $z_{h,j^h}^k = 1$, i.e., if and only if the solution z_h^k includes item j, and $Dz_l^k = -1$ if and only if $z_{l,j^l}^k = 1$, i.e., if and only if the solution z_l^k includes item j (note that item j is indexed as j^h in block h and j^l in block l).

By substituting the objective function as well we obtain the following reformulated $REF2-TKP$ model:

$$\max \sum_{h=1,\dots,g} \sum_{k=1,\dots,p_h} \lambda_h^k \sum_{j=1,\dots,\overline{n}} z_{h,j}^k \pi_j \tag{3.45}$$

$$\sum_{k=1,\dots,p_h} z_{h,j^h}^k \lambda_h^k = \sum_{k=1,\dots,p_l} z_{l,j^l}^k \lambda_l^k \quad h = 1,\dots,g,\ l = h+1,\dots,g,\ j \in G_h \cap G_l \tag{3.46}$$

$$\sum_{k=1,\dots,p_h} \lambda_k^h \leq 1 \qquad h = 1,\dots,g \tag{3.47}$$

$$\lambda_k^h \in \{0,1\} \qquad h = 1,\dots,g,\ k = 1,\dots,p_h \tag{3.48}$$

Model $REF2-TKP$ is equivalent to model $REF-TKP$ once one observes that constraints (3.46) are the counterpart of the $x-y$ linking constraints (3.40), and (3.47) are the same as (3.41).

3.5 Detailed description of Reformulated Models

In this section we describe in detail the "One Constraint Slave" Reformulated Model and the "Overlapping Constraints Slave" Reformulated Model. Precisely we take into consideration the specific case of Model $REF-TKP$ (3.39) - (3.43) where each slave is composed by one single constraint and several overlapping constraints respectively. In both cases the slaves are generated in such a way that all the constraints of the traditional formulation are covered and reformulated.

3.5.1 "One Constraint Slave" Reformulated Model

In order to define the first model we introduce $S_i = \{s_i \subseteq G_i : \sum_{j \in s_i} w_j \leq C\}$ which represents the set of all the possible feasible item subsets of G_i. As before we use two groups of binary variables. The first one, x_j, is a binary variable indicating whether item j is in the optimal solution. The second one, y_{s_i}, is a binary variables indicating whether items j $\in s_i$ are in the optimal solution. It is important to notice that here the variables y_{s_i} are associated to feasible subsets of items instead of to extreme points; but for the *Temporal Knapsack*

Problem these two entities are exactly the same. For the following models we will also use an extension of the idea of the dual feasible functions used by Fekete and Schepers [100] to get a lower bound for the bin packing problem. To this end we define $\theta_1^i = \lfloor C/max_{j\in G_i}(w_j) \rfloor$ and $\theta_2^i = \lfloor C/min_{j\in G_i}(w_j) \rfloor$ which are respectively the minimum and the maximum number of items you can possible insert in each knapsack constraint i. We are using them to define additional valid inequalities for the model. Finally in order to strengthen the formulation a little bit more we leave also the original constraints of Model $TRAD - TKP$ (3.7) - (3.9), precisely constraints (3.8). As will become clearer in the following paragraphs, these constraints do not affect the column generation procedure and they can be easily inserted as the extended model keeps the original variables.
The model objective function (3.49) maximizes the profit of the item subset chosen. Constraints (3.50) impose that you can select an item in a specific knapsack if and only if at least one feasible subset which contains that specific item is in chosen. Constraints (3.51) are the convexity constraints one for each knapsack i, and they impose that you can select at most one feasible item subset per knapsack.The primal model of the "One Constraint Slave" Reformulated Model reads as follows.

"One Constraint Slave" Primal Model:

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.49}$$

$$\sum_{s_i \in S_i : j \in s_i} y_{s_i} \geq x_j \qquad i = 1, ...m, \forall j \in G_i \tag{3.50}$$

$$\sum_{s_i \in S_i} y_{s_i} \leq 1 \qquad i = 1, ...m \tag{3.51}$$

$$\sum_{j \in G_i} \left\lfloor \frac{w_j}{\frac{C}{\mu+1}} - \epsilon \right\rfloor x_j \leq \mu + 1 \quad i = 1, ..., m, \mu = \theta_1^i, ..., \theta_2^i \tag{3.52}$$

$$\sum_{j \in G_i} w_j x_j \leq C \qquad i = 1, ...m \tag{3.53}$$

$$x_j \in \{0, 1\} \quad j = 1, ..., n \tag{3.54}$$

$$y_{s_i} \in \{0, 1\} \quad i = 1, ...m, \forall s_i \in S_i \tag{3.55}$$

It is important to notice that we replace the equalities with the inequalities in both (3.50) and (3.51), to be more general we put the equalities in $REF - TKP$ model (3.39)-(3.43) while in this model inequalities are sufficient in that we are dealing with a maximization problem with positive profits and weights and just inequalities in the original formulation. This cuts half of the dual space and it works as a speed up in the convergence to the optimization process. Although in general this is a relaxation of the original model, for our problem we do not deteriorate the quality of the upper bound of the continuous relaxation value. Constraints (3.53) are the original constraints of $TRAD - TKP$. Constraints (3.54) and (3.55) impose the binary condition on the variables. It is important to notice that it is possible to replace constraints (3.55) with just the non negativity requirement on y variables thanks to constraints (3.50). In addition, constraints (3.52), which are the extension of the dual

feasible function, impose a maximum number for different items simultaneously present in a knapsack according to the weights. Constant ε is a number smaller than 1, used to decrease by one unit the value associated with items which are inserted into the knapsack and have a weight which is perfectly divisible by a specific percentage of the capacity. Precisely, there are some simultaneous insertion incompatibilities between items in a knapsack i. With those constraints some non integer feasible solutions are cut. For example, when θ is 1, if there are items whose weight is bigger than half of the capacity, only one item of that kind can be chosen.
The following model (3.49)-(3.55) is the continuous relaxation of the previous model, we put the constraints in the classical form to better illustrate the relations with the dual model. In addition for each constraint, we report the associated specific dual variable name.
Then we show the dual of the model (3.56) - (3.62), this is important in that it contains the constraints to be separated during the *Column Generation* procedure, i.e. (3.64). As previously mentioned, the y variables are exponential and you cannot enumerate all of them in advance but they must be generated during the optimization thanks to the dual variable values. Due to the fact that we keep the original variables x, constraints (3.65) are all present from the beginning and they do not need *separation*.

Continuous relaxation of (3.49)-(3.55)

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.56}$$

$$x_j + \sum_{s_i \in S_i : j \in s_i} -y_{s_i} \leq 0 \qquad i = 1, ...m, \forall j \in G_i \quad (\pi_{i,j}) \tag{3.57}$$

$$\sum_{s_i \in S_i} y_{s_i} \leq 1 \qquad\qquad i = 1, ...m \quad (\gamma_i) \tag{3.58}$$

$$\sum_{j \in G_i} \left\lfloor \frac{w_j}{\frac{C}{\mu+1}} - \epsilon \right\rfloor x_j \leq \mu + 1 \quad i = 1, ..., m, \mu = \theta_1^i, ..., \theta_2^i \quad (\beta_{i,\mu}) \tag{3.59}$$

$$\sum_{j \in G_i} w_j x_j \leq C \qquad i = 1, ...m \quad (\eta_i) \tag{3.60}$$

$$x_j \geq 0 \quad j = 1, ..., n \tag{3.61}$$

$$y_{s_i} \geq 0 \quad i = 1, ...m, \forall s_i \in S_i \tag{3.62}$$

Dual Model of (3.56)-(3.62)

$$min \sum_{i=1}^{m} \gamma_i + \sum_{i=1}^{m} \sum_{\mu=\theta_1^i}^{\theta_2^i} (\mu+1)\beta_{i,\mu} + \sum_{i=1}^{m} C\eta_i \tag{3.63}$$

$$\sum_{j \in s_i} -\pi_{i,j} + \gamma_i \geq 0 \qquad i = 1, ...m, \forall s_i \in S_i \tag{3.64}$$

$$\sum_{i:j \in G_i} \pi_{i,j} + \sum_{i=1}^{m} \sum_{\mu=\theta_1^i}^{\theta_2^i} \left\lfloor \frac{w_j}{\frac{C}{\mu+1}} - \epsilon \right\rfloor \beta_{i,\mu} + \sum_{i=1}^{m} w_j \eta_i \geq p_j \quad j = 1, ...n \tag{3.65}$$

$$\pi_{i,j} \geq 0 \quad i = 1, ...m, j = 1, ...n \tag{3.66}$$

$$\beta_{i,\mu} \geq 0 \quad i = 1, ...m, \mu = \theta_1^i, ..., \theta_2^i \tag{3.67}$$

$$\gamma_i \geq 0 \quad i = 1, ...m \tag{3.68}$$

$$\eta_i, \geq 0 \quad i = 1, ...m \tag{3.69}$$

Column Generation Procedure for (3.56)-(3.62)

The continuous relaxation of Model (3.49)-(3.55) must be solved using *Column Generation*, i.e. constraints (3.64) must be separated. Now, since we have m different knapsacks, we set a slave for each of them. The details of the procedure follow. Given i, $\pi_{i,j}^*$,γ_i^* the separation problem is to determine, if it exists, a subset $s_i^* \subseteq \{S_i\}$ such as:

$$\sum_{j \in s_i^*} -\pi_{i,j}^* + \gamma_i^* < 0 \tag{3.70}$$

$$\sum_{j \in s_i^*} \pi_{i,j}^* > \gamma_i^* \tag{3.71}$$

Introducing a new binary variable β_j which has value 1 if $j \in s_i^*$ and 0 otherwise, the *Slave Problem* becomes:

$$\max \sum_{j \in G_i} \pi_{i,j}^* \beta_j \tag{3.72}$$

$$\sum_{j \in G_i} w_j \beta_j \leq C \tag{3.73}$$

$$\beta_j \in \{0, 1\} \qquad j = 1, ...n \tag{3.74}$$

If the optimal solution of Model (3.72) - (3.74) (let's call it δ_i^*) is $\leq \gamma_i^*$ then all the dual constraints defined by i are satisfied by the solution $\pi_{i,j}^*$,γ_i^*, otherwise the subset $s_i^* = \{j \in \{1, ..., n\} : \beta_j^* = 1\}$ defines the most violated constraint to be added and then the procedure is reiterate. It is important to notice that the *Slave Problem* becomes a simple *KP-01*, but at each iteration, instead of the original profits, you optimize using the values of the dual variables.

In addition, at a certain point of the column generation procedure, we can obtain a feasible dual solution, i.e. a valid upper bound for the original problem. Calling ϕ^* the current optimal value of (3.56)-(3.62), the bound is equal to the following formula:

$$\phi^* + \sum_{i:\delta_i^* - \gamma_i^* > 0} \delta_i^* - \gamma_i^*$$

This is due to the fact that all the dual constraints are feasible giving at least that extra value to the variables γ. The objective function increases accordingly from ϕ^* to that bound value.

The table 3.4 reports the Model (3.49)-(3.55) with the complete variable y enumeration. In that example we have 3 subsets G and accordingly here we have 3 convexification constraints (3.51), it is important to notice that item 2 is present in all the subsets G and the link is made by 3 constraints of kind (3.50).

O.F.	$+1x_1$ $+2x_2$ $+1x_3$ $+3x_4$ $+4x_5$			
S.T.	$+x_1$	$-y_{\{1\}1}$ $-y_{\{1,2\}1}$	$\leq$	0
	$+x_2$	$-y_{\{2\}1}$ $-y_{\{1,2\}1}$	$\leq$	0
	$+x_2$	$-y_{\{2\}2}$ $-y_{\{2,3\}2}$ $-y_{\{2,4\}2}$	$\leq$	0
	$+x_3$	$-y_{\{3\}2}$ $-y_{\{2,3\}2}$ $-y_{\{3,4\}2}$	$\leq$	0
	$+x_4$	$-y_{\{4\}2}$ $-y_{\{2,4\}2}$ $-y_{\{3,4\}2}$	$\leq$	0
	$+x_2$	$-y_{\{2\}3}$ $-y_{\{2,5\}3}$	$\leq$	0
	$+x_5$	$-y_{\{5\}3}$ $-y_{\{2,5\}3}$	$\leq$	0
		$+y_{\{1\}1}$ $+y_{\{2\}1}$ $+y_{\{1,2\}1}$	$\leq$	1
		$+y_{\{2\}2}$ $+y_{\{3\}2}$ $+y_{\{2,3\}2}$ $+y_{\{4\}2}$ $+y_{\{2,4\}2}$ $+y_{\{3,4\}2}$	$\leq$	1
		$+y_{\{2\}3}$ $+y_{\{5\}3}$ $+y_{\{2,5\}3}$	$\leq$	1

Table 3.4: Example 1 Model (3.49)-(3.55) (complete variable enumeration)

3.5.2 Projection of "One Constraint Slave" Model on the "y variables" space

Model (3.49)-(3.55) uses both binary variables x and y but it is possible to avoid using the first one, projecting the polyhedra on the y variables only. The idea here is to model the presence of an item in more than one G using equality constraints between overlapping G. To this end we define l_j as the number of maximal knapsack constraints G in which item j is present and N_+ as the set of items which belongs to more than one maximal knapsack constraint G. Using these new entities we can introduce a new version of item subset profit, i.e. $P_{s_i} = \sum_{j \in s_i} (p_j / l_j)$. Each item contributes to the subset profit with a fraction of its profit p_j in that, in the optimal solution, you have to select a feasible subset for each G in which item j is present. In this way you can reconstruct the original value of profit. Projection of "One Constraint Slave" Model on the "y variables" space reads as follows:

Projection of "One Constraint Slave" Model on the "y variables" Primal Model

$$\max \sum_{i=1}^{m} \sum_{s_i \in S_i} P_{s_i} y_{s_i} \tag{3.75}$$

$$\sum_{s_i \in S_i : j \in s_i} y_{s_i} = \sum_{s_{i+1} \in S_{i+1} : j \in s_{i+1}} y_{s_{i+1}} \qquad j \in N_+, \forall i : j \in G_i \cap G_{i+1} \tag{3.76}$$

$$\sum_{s_i \in S_i} y_{s_i} \leq 1 \qquad i = 1, ...m \tag{3.77}$$

$$y_{s_i} \in \{0, 1\} \quad i = 1, ...m, \forall s_i \in S_i \tag{3.78}$$

The new objective function (3.75) models the total profit of the item selected in the optimal solution. Constraints (3.77) are the convexity constraints as in the previous model. Constraints (3.76) make the link for the item $j \in N_+$, i.e. which appear in more than one G. They say that for an overlapping couple of knapsack constraints which share item j, if you select a subset of G_i with item j, you have to select one subset of G_{i+1} with that item as well. Constraints (3.78) impose the binary condition on the variables. To be able to easily write the dual of this model we report the continuous relaxation of Model (3.75)-(3.78) in a slightly different manner. We introduce Θ_j as the set of couples (k, u) of Maximal Knapsack constraints G_i, G_{i+1} which include item j. Moreover we show the dual variable names associated with each group of primal constraints. The relaxed model reads as follows:

Continuous relaxation of (3.75)-(3.78)

$$\max \sum_{i=1}^{m} \sum_{s_i \in S_i} P_{s_i} y_{s_i} \tag{3.79}$$

$$\sum_{s_k \in S_k : j \in s_k} y_{s_k} - \sum_{s_u \in S_u : j \in s_u} y_{s_u} = 0 \qquad j \in N_+, \forall o \in \Theta_j \quad (\pi_{o,j}) \tag{3.80}$$

$$\sum_{s_i \in S_i} y_{s_i} \leq 1 \qquad i = 1, ...m \quad (\gamma_i) \tag{3.81}$$

$$y_{s_i} \geq 0 \quad i = 1, ...m, \forall s_i \in S_i \tag{3.82}$$

Once we have the continuous relaxation written in this way it is simple to write the dual model. This time we only have a set of constraints and the variable π is free in that we have equality constraints in the primal model.

Dual Model of (3.79)-(3.82)

$$min \sum_{i=1}^{m} \gamma_i \tag{3.83}$$

$$\sum_{j \in s_i} \sum_{o \in \Theta_j : i=k, i \neq u} \pi_{o,j} + \sum_{j \in s_i} \sum_{o \in \Theta_j : i=u, i \neq k} -\pi_{o,j} + \gamma_i \geq P_{s_i} \quad i = 1, ...m, \forall s_i \in S_i \tag{3.84}$$

$$\gamma_i \geq 0 \quad i = 1, ...m \tag{3.85}$$

$$\tag{3.86}$$

Column Generation Procedure for (3.79)-(3.82)

As before the continuous relaxation of Model (3.75)-(3.78) must be solved using *Column Generation*, i.e. constraints (3.84) must be separated. Given i, $\pi^*_{o,j}$,γ^*_i determine, if it exists, a subset $s^*_i \subseteq \{S_i\}$ such as:

$$\sum_{j \in s^*_i} \sum_{o \in \Theta_j : i=k, i \neq u} \pi^*_{o,j} + \sum_{j \in s^*_i} \sum_{o \in \Theta_j : i=u, i \neq k} -\pi^*_{o,j} + \gamma^*_i < P_{s_i{}^*} \tag{3.87}$$

$$\sum_{j \in s^*_i} [\sum_{o \in \Theta_j : i=k, i \neq u} \pi^*_{o,j} + \sum_{o \in \Theta_j : i=u, i \neq k} -\pi^*_{o,j}] - P_{s^*_i} < -\gamma^*_i \tag{3.88}$$

$$\sum_{j \in s^*_i} [\sum_{o \in \Theta_j : i=k, i \neq u} -\pi^*_{o,j} + \sum_{o \in \Theta_j : i=u, i \neq k} +\pi^*_{o,j}] + P_{s^*_i} > \gamma^*_i \tag{3.89}$$

Using the binary variable β_j which has value 1 if $j \in s^*_i$ and 0 otherwise, the *Slave Problem* becomes:

$$\max \sum_{j \in G_i} [\sum_{o \in \Theta_j : i=k, i \neq u} -\pi^*_{o,j} + \sum_{o \in \Theta_j : i=u, i \neq k} +\pi^*_{o,j} + p_j/l_j]\beta_j \tag{3.90}$$

$$\sum_{j \in G_i} w_j \beta_j \leq C \tag{3.91}$$

$$\beta_j \in \{0, 1\} \qquad j = 1, ...n \tag{3.92}$$

If the optimal solution of Model (3.90)-(3.92) is $\leq \gamma^*_i$ then all the dual constraints defined by i are satisfied by the solution $\pi^*_{o,j}$,γ^*_i, otherwise the subset $s^*_i = \{j \in \{1, ..., n\} : \beta^*_j = 1\}$ defines the most violated constraint. It is important to notice that the *Slave Problem* is still a simple *KP-01*, but this time the dual variable value is just a part of the new item profit to be optimized.

The table 3.4 reports the Model (3.75)-(3.78) with the complete variable y enumeration. Here we have again the 3 convexity constraints for the 3 maximal knapsack constraints G and 2 constraints modeling the presence of item 2 in every subset G.

O.F.	$+1y_{\{1\}1} +2/3y_{\{2\}1} +5/3y_{\{1,2\}1} +2/3y_{\{2\}2} +1y_{\{3\}2}$ $+3y_{\{4\}2} +5/3y_{\{2,3\}2} +11/3y_{\{2,4\}2} +4y_{\{3,4\}2} +2/3y_{\{2\}3} +4y_{\{5\}3} +14/3\ y_{\{2,5\}3}$		
S.T.	$+y_{\{2\}1} +y_{\{1,2\}1} -y_{\{2\}2} -y_{\{2,3\}2} -y_{\{2,4\}2}$	$=$	0
	$+y_{\{2\}2} +y_{\{2,3\}2} +y_{\{2,4\}2} -y_{\{2\}3} -y_{\{2,5\}3}$	$=$	0
	$+y_{\{1\}1} +y_{\{2\}1} +y_{\{1,2\}1}$	$\leq$	1
	$+y_{\{2\}2} +y_{\{3\}2} +y_{\{2,3\}2} +y_{\{4\}2} +y_{\{2,4\}2} +y_{\{3,4\}2}$	$\leq$	1
	$+y_{\{2\}3} +y_{\{5\}3} +y_{\{2,5\}3}$	$\leq$	1

Table 3.5: Example 1 Model (3.75)-(3.78) (complete variable enumeration)

3.5.3 "Overlapping Constraints Slave" Reformulated Model

In this section we focus our attention on the "Overlapping Constraints Slave" Reformulated Model, here the idea is to create a *Slave Problem* with more than one overlapping original constraint. The goal is to achieve higher value of continuous relaxation, i.e. better bounds. On the other hands, if the *Slave Problem* becomes too big the column generation process starts to be slow and there is a balance between the speed of the overall method, the quality of the bound and number of constraints reformulated for each *Slave Problem*. For simplicity we will describe the case of just 2 overlapping constraints but it can be easily extended to the general case of n constraints. To define the new model we need to introduce $H_{(i,i+1)} = G_i \cup G_{i+1}$ as a subset of items which belongs to an overlapping couple $(i, i+1)$ of maximal knapsack constraints and $S_{(i,i+1)} = \{s_{(i,i+1)} \subseteq H_{(i,i+1)} : \sum_{j \in s_{(i,i+1)} \cap G_i} w_j \leq C, \sum_{j \in s_{(i,i+1)} \cap G_{i+1}} w_j \leq C\}$, which represents the set of all the possible feasible item subsets of $H_{(i,i+1)}$. As before, we use two groups of binary variables. The first one, x_j, is a binary variable indicating whether item j is in the optimal solution. The second one, $y_{s_{(i,i+1)}}$, is a binary variable indicating whether items j $\in s_{i,i+1}$ are in the optimal solution.

"Overlapping Constraints Slave" Primal Model

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.93}$$

$$\sum_{s_{(2i-1,2i)} \in S_{(2i-1,2i)} : j \in s_{(2i-1,2i)}} y_{s_{(2i-1,2i)}} \geq x_j \quad i = 1, .., m/2, \forall j \in H_{(2i-1,2i)} \tag{3.94}$$

$$\sum_{s_{(2i-1,2i)} \in S_{(2i-1,2i)}} y_{s_{(2i-1,2i)}} \leq 1 \qquad i = 1, .., m/2 \tag{3.95}$$

$$x_j \in \{0, 1\} \quad j = 1, ..., n \tag{3.96}$$

$$y_{s_{(i,i+1)}} \in \{0, 1\} \quad \forall s_{(i,i+1)} \in S_{(i,i+1)} \tag{3.97}$$

The model objective function (3.93) maximizes the profit of the subset of items chosen. Constraints (3.94) impose that you can select an item in a specific couple of knapsacks if and only if at least one feasible subset which contains that specific item is chosen. Constraints (3.95) are the convexity constraints, one for each couple of knapsacks i and they impose that you can select at most one feasible item subset. Constraints (3.96) and (3.97) impose the binary condition on the variables. It is important to notice that it is possible to replace constraints

(3.96) with just the non negativity requirement on y variables thanks to constraints (3.97). For simplicity we have omitted to write the additional inequalities but they obviously hold also in this case.

We report the continuous relaxation of the previous model, along with the associated specific dual variable name for each constraint.

Continuous relaxation of (3.93)-(3.97)

$$\max \sum_{j=1}^{n} p_j x_j \tag{3.98}$$

$$\sum_{s_{(2i-1,2i)} \in S_{(2i-1,2i)}: j \in s_{(2i-1,2i)}} y_{s_{(2i-1,2i)}} \geq x_j \quad i = 1,..,m/2, \forall j \in H_{(2i-1,2i)} \; (\pi_{(2i-1,2i),j}) \tag{3.99}$$

$$\sum_{s_{(2i-1,2i)} \in S_{(2i-1,2i)}} y_{s_{(2i-1,2i)}} \leq 1 \qquad i = 1,..,m/2 \qquad (\gamma_{(2i-1,2i)}) \tag{3.100}$$

$$x_j \geq 0 \quad j = 1,....,n \tag{3.101}$$

$$y_{s_{(i,i+1)}} \geq 0 \quad h = 1,...m_c, \forall s_{(i,i+1)} \in S_{(i,i+1)} \tag{3.102}$$

Then we show the dual of the model (3.98) - (3.102), also in this case constraints (3.104) must be separated during the *Column Generation* procedure, in that the y variables are exponential and one cannot enumerate all of them in advance. Accordingly they must be generated during the optimization thanks to the dual variable values.

Dual Model of (3.98)-(3.97)

$$min \sum_{i=1}^{m/2} \gamma_{(2i-1,2i)} \tag{3.103}$$

$$\sum_{j \in s_{(2i-1,2i)}} -\pi_{(2i-1,2i),j} + \gamma_{(2i-1,2i)} \geq 0 \qquad i = 1,..,m/2, \forall s_{(i,i+1)} \in S_{(i,i+1)} \tag{3.104}$$

$$\sum_{(i,i+1): j \in H_{(i,i+1)}} \pi_{(i,i+1),j} \geq p_j \quad \forall j \tag{3.105}$$

$$\gamma_{(2i-1,2i)} \geq 0 \quad i = 1,..,m/2 \tag{3.106}$$

$$\pi_{(2i-1,2i),j} \geq 0 \quad i = 1,..,m/2, j = 1,....n \tag{3.107}$$

Column Generation for (3.98)-(3.97)

The continuous relaxation of Model (3.93)-(3.97) must be solved using *Column Generation*, i.e. constraints (3.104) must be separated. Given (i,i+1), $\pi^*_{(i,i+1),j}$,$\gamma^*_{(i,i+1)}$ determine, if it exists, a subset $s^*_{(i,i+1)} \subseteq \{S_{(i,i+1)}\}$ such as:

$$\sum_{j \in s^*_{(i,i+1)}} -\pi^*_{(i,i+1),j} + \gamma^*_{(i,i+1)} < 0 \tag{3.108}$$

$$\sum_{j \in s^*_{(i,i+1)}} \pi^*_{(i,i+1),j} > \gamma^*_{(i,i+1)} \tag{3.109}$$

Using the binary variable β_j which has value 1 if $j \in s^*_i$ and 0 otherwise, the *Slave Problem* becomes:

$$\max \sum_{j \in H_{(i,i+1)}} \pi^*_{(i,i+1),j}\beta_j \tag{3.110}$$

$$\sum_{j \in G_i} w_j\beta_j \leq C \qquad i, i+1 \tag{3.111}$$

$$\beta_j \in \{0,1\} \qquad j = 1, ...n \tag{3.112}$$

If the optimal solution of model (3.110)-(3.112) $\leq \gamma^*_{(i,i+1)}$ then all the dual constraints defined by (i,i+1) are satisfied by the solution $\pi^*_{(i,i+1),j}$,$\gamma^*_{(i,i+1)}$, otherwise the subset $s^*_{(i,i+1)} = \{j \in \{1, ..., n\} : \beta^*_j = 1\}$ defines the most violated constraint. It is important to notice that the *Slave Problem* is not a *KP-01*, but at each iteration you have to solve a small *Temporal Knapsack Problem* with just 2 constraints using the values of the dual variables as profits.

Table 3.6 contains the data for the Temporal Knapsack Example 2, then figure 3.4 shows the interval graph representation of that example. In the figure the ellipses represent the sets G and H.

Task	**Profit**	**Weight**	**Start**	**Finish**
$Task_1$	3	1	1	3
$Task_2$	5	2	2	14
$Task_3$	4	3	5	10
$Task_4$	3	4	7	8
$Task_5$	2	5	12	19
$Task_6$	1	6	15	19
C=8				

Table 3.6: Data example 2

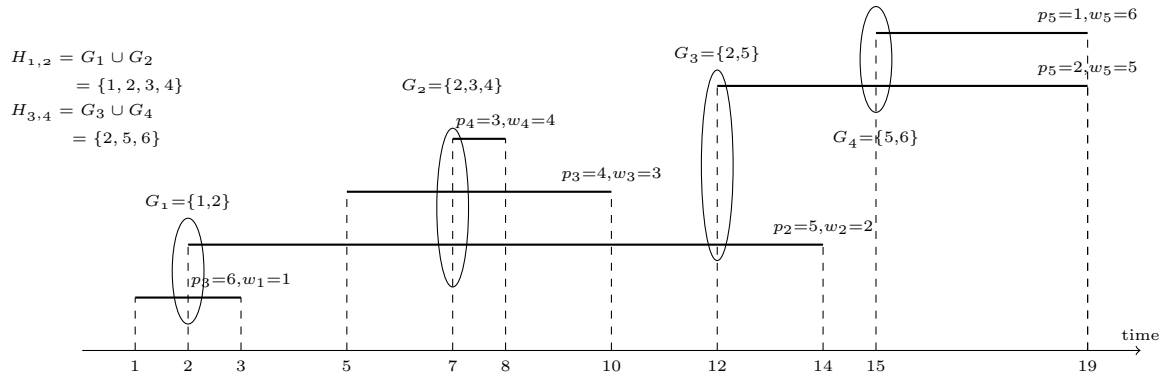


Figure 3.4: Interval Graph Example 2

The table 3.6 reports the Model (3.103) - (3.107) with the complete variables y enumeration for Example 2.

O.F.	$+3x_1 +5x_2 +4x_3$ $+3x_4 +2x_5 +1x_6$			
S.T.	$+x_1$	$-y_{\{1\}1} -y_{\{1,2\}1} -y_{\{1,3\}1} -y_{\{1,2,3\}1} -y_{\{1,4\}1} -y_{\{1,2,4\}1} -y_{\{1,3,4\}1}$	$\leq$	0
	$+x_2$	$-y_{\{2\}1} -y_{\{1,2\}1} -y_{\{2,3\}1} -y_{\{1,2,3\}1} -y_{\{2,4\}1} -y_{\{1,2,4\}1} -y_{\{2,4\}1}$	$\leq$	0
	$+x_3$	$-y_{\{3\}1} -y_{\{1,3\}1} -y_{\{2,3\}1} -y_{\{1,2,3\}1} -y_{\{1,3,4\}1}$	$\leq$	0
	$+x_4$	$-y_{\{4\}1} -y_{\{1,4\}1} -y_{\{2,4\}1} -y_{\{1,2,4\}1} -y_{\{2,4\}1} -y_{\{1,3,4\}1}$	$\leq$	0
	$+x_2$	$-y_{\{2\}2} -y_{\{2,5\}2} -y_{\{2,6\}2}$	$\leq$	0
	$+x_5$	$-y_{\{5\}2} -y_{\{2,5\}2}$	$\leq$	0
	$+x_6$	$-y_{\{6\}2} -y_{\{2,6\}2}$	$\leq$	0
		$+y_{\{1\}1} +y_{\{2\}1} +y_{\{1,2\}1} +y_{\{3\}1} +y_{\{1,3\}1} +y_{\{2,3\}1} +y_{\{1,2,3\}1}$ $+y_{\{4\}1} +y_{\{1,4\}1} +y_{\{2,4\}1} +y_{\{1,2,4\}1} +y_{\{2,4\}1} +y_{\{1,3,4\}1}$	$\leq$	1
		$+y_{\{2\}2} +y_{\{5\}2} +y_{\{2,5\}2} +y_{\{6\}2} +y_{\{2,6\}2}$	$\leq$	1

Table 3.7: Example 2 model (3.103) - (3.107) (complete variable enumeration)

3.6 Computational Results

In this section we report computational experiments on randomly generated instances, so as to evaluate the performance of the algorithms and the reformulations previously presented:

- $TRAD - TKP$ - the model is directly solved through CPLEX12.2 with standard parameters set-up, we also tested a more aggressive cut generation setting without notable improvement. Note that the continuous relaxation of the model provides a valid upper bound for the problem. A better upper bound can be obtained by solving the root node of the $TRAD - TKP$ model with CPLEX12.2, which in this case adds families of general cuts to strengthen the relaxation;
- Cut Generation - according to Section 3.4.1, given a fractional solution to model $TRAD-TKP$ we generate a valid cut for every group of constraints by solving model SEP (whose exponentially many constraints are separated by solving model $sep(SEP)$) with

CPLEX12.2. The constraints of the $TRAD-TKP$ model are partitioned in groups, so as to cover the whole constraint matrix. All generated cuts are added to the model, which is then re-optimized. The procedure is iterated until no cut violated by the current solution can be found. Note that when the procedure is stopped before convergence, the rounded down value of the continue relaxation of model $TRAD-TKP$ (with the added cuts) is a valid upper bound on the value of the optimal integer solution to TKP;

- *$REF-TKP$ Continuous Relaxation* - To solve the continuous relaxation of the $REF-TKP$ model, the constraints of the original $TRAD-TKP$ model are are partitioned in groups, so as to cover the whole constraint matrix, and the restricted master problem is initialized by the x variables (which allow the null solution). New y variables with positive reduced profit, associated with each group of constraints, are then generated by solving slave problems of the form (3.24)–(3.26) (one slave problem is defined for each group of constraints). The generated y variables are added to the restricted master problem which is re-optimized. The column generation scheme is iterated until no column (y variable) with positive reduced profit is generated. The master problem and the slave problems are solved with CPLEX12.2. The continuous relaxation of the $REF-TKP$ model provides a valid upper bound on the optimal solution value of TKP. If the column generation procedure is stopped before convergence, a valid upper bound can be computed by adding to the current solution (of the restricted master problem) the maximal violations of the constraints of the $REF-TKP$ dual, which are equivalent to (3.20) (one for each group of constraints). This bounding procedure is equivalent to computing the value of a feasible solution to the dual of model $REF-TKP$;

- *$REF-TKP$ Integer Solution* - The solution to the continuous relaxation of model $REF-TKP$ can be fractional, thus, in order to obtain a feasible integer solution to TKP, we apply the branching procedure described in Section 3.4.2. Note that we do not take advantage of the sophisticated branching scheme implemented by CPLEX12.2, which cannot be integrated with a column generation procedure;

- Dynamic Programming algorithm - it is directly applied to the formulation $TRAD-TKP$.

All computational tests were conducted by using 1 core of an INTEL Core2 Duo E6550 at 2.33GHz with 8GB of RAM and Cplex 12.2, under LINUX Ubuntu 10.4 operating system.

3.6.1 Random Instance Generator

In this section we describe the way we generated the instances to test the performance of the models described in the previous sections. Our random instance generator produces TKP instances according to the $TRAD-TKP$ formulation. The first parameter is the number m of constraints. The second input parameter is the interval $[dimC_{max}, dimC_{min}]$ we use to define the number of items simultaneously present in one constraint. To be precise, the constraint dimension is a random number generated with a uniform distribution within this interval. Due to the fact one constraint is generated starting from the previous one, the third parameter indicates the percentage of items to be taken from the previous constraint. As before, this percentage is a random number generated with a uniform distribution in the interval $[dimO_{max}, dimO_{min}]$. As far as the profit and the weight of the items are concerned, we generate them as a random number chosen in the intervals $[Pr_{max}, Pr_{min}]$ and $[W_{max}, W_{min}]$,

which are the fourth and fifth parameters. The whole case study is composed of ten groups of ten test instances with similar characteristics. To generate the 10 instances of each group, we fix the value of all the parameters except the number of constraints. For the first 4 groups the number of constraints of the first instance of each group is set to 2688 and the others are obtained adding 128 constraints, each time. The same idea is used for the other groups, but the starting value of constraints is set to 768. The first four groups are specifically conceived to test the performance of the dynamic programming algorithm and the cardinality of the constraints is relatively small. The last six groups are on the other hand conceived to test the $REF-TKP$ models. The cut generation scheme is tested on these groups as well. The main characteristic of these groups is a greater constraint cardinality which makes the dynamic programming algorithm unusable in practice.
The values of the input parameters used to generate the ten groups are presented in Table 3.8. It shows the parameter values, described before, used to generate the 10 random groups, the last column $p = w$ is equal to 1 if we set the item weight equal to the random profit, 0 otherwise.

Table 3.8: Input parameter values used to generate the test instances

	$dimC_{min}$	$dimC_{max}$	$dimO_{min}$	$dimO_{max}$	$dimP_{min}$	$dimP_{max}$	$dimWmin$	$dimW_{max}$	$p = w$
Group I	10	10	95	90	100	10	100	10	0
Group II	15	15	95	90	100	10	100	10	0
Group III	20	20	95	90	100	10	100	10	0
Group IV	25	25	95	90	100	10	100	10	0
Group V	30	30	95	90	100	10	100	10	0
Group VI	30	30	90	70	100	10	100	10	0
Group VII	30	30	95	90	100	10	100	10	1
Group VIII	35	25	95	90	100	10	100	10	0
Group IX	35	25	90	70	100	10	100	10	0
Group X	40	30	95	90	100	10	100	10	0

In table 3.9 we report the specific features of each instances for the groups V-X. These are the most important groups in that we test on those the reformulated models. In details, the first column is the instance name then the number of variables and constraints followed by 3 indices. Index I_1 is the average number of constraints in which each item is present, Index I_2 is the average number of items for each knapsack constraints and finally Index I_3 reports the biggest and the lowest cardinality of subsets G.

	cols	**rows**	*Index* I_1	*Index* I_2	*Index* I_3	**opt**
I41	2071	768	11.1	30.0	30-30	30866
I42	2422	896	11.1	30.0	30-30	35771
I43	2756	1024	11.1	30.0	30-30	40934
I44	3104	1152	11.1	30.0	30-30	46180
I45	3433	1280	11.2	30.0	30-30	50324
I46	3789	1408	11.1	30.0	30-30	55495
I47	4154	1536	11.1	30.0	30-30	59255
I48	4476	1664	11.2	30.0	30-30	65465
I49	4797	1792	11.2	30.0	30-30	69530
I50	5129	1920	11.2	30.0	30-30	75756
I51	4948	768	4.7	30.0	30-30	71998
I52	5769	896	4.6	30.0	30-30	81898
I53	6721	1024	4.7	30.0	30-30	97056
I54	7382	1152	4.7	30.0	30-30	107491
I55	8266	1280	4.6	30.0	30-30	120505
I56	9002	1408	4.7	30.0	30-30	129053
I57	9865	1536	4.7	30.0	30-30	142486
I58	10661	1664	4.7	30.0	30-30	151489
I59	11448	1792	4.7	30.0	30-30	165076
I60	12498	1920	4.6	30.0	30-30	182813
I61	2071	768	11.1	30.0	30-30	22044
I62	2422	896	11.1	30.0	30-30	26115
I63	2763	1024	11.1	30.0	30-30	29110
I64	3103	1152	11.1	30.0	30-30	32692
I65	3434	1280	11.2	30.0	30-30	37016
I66	3774	1408	11.2	30.0	30-30	39593
I67	4164	1536	11.1	30.0	30-30	44735
I68	4488	1664	11.1	30.0	30-30	48182
I69	4786	1792	11.2	30.0	30-30	50559
I70	5142	1920	11.2	30.0	30-30	54842
I71	2916	768	8.0	30.3	35-25	40982
I72	3424	896	7.8	29.8	35-25	47914
I73	3832	1024	8.0	30.1	35-25	52447
I74	4316	1152	8.0	30.0	35-25	59790
I75	4771	1280	8.1	30.1	35-25	66179
I76	5403	1408	7.8	30.0	35-25	75070
I77	5793	1536	8.0	30.0	35-25	81982
I78	6167	1664	8.1	30.0	35-25	85314
I79	6800	1792	7.9	29.9	35-25	95037
I80	7241	1920	7.9	29.9	35-25	100031
I81	5210	768	4.5	30.3	35-25	71426
I82	6057	896	4.4	30.0	35-25	82942
I83	6901	1024	4.4	29.8	35-25	96115
I84	7737	1152	4.5	30.1	35-25	110102
I85	8656	1280	4.5	30.1	35-25	119233
I86	9370	1408	4.5	30.0	35-25	128178
I87	10271	1536	4.5	30.1	35-25	142056
I88	11057	1664	4.5	29.9	35-25	154745
I89	11992	1792	4.5	30.0	35-25	167916
I90	13025	1920	4.4	30.1	35-25	176881
I91	3117	768	8.7	35.3	30-40	42685
I92	3594	896	8.7	35.0	30-40	46526
I93	4176	1024	8.6	34.9	30-40	54437
I94	4671	1152	8.6	34.8	30-40	60719
I95	5209	1280	8.6	34.9	30-40	68432
I96	5628	1408	8.8	35.0	30-40	72337
I97	6215	1536	8.7	35.0	30-40	80122
I98	6730	1664	8.6	35.0	30-40	88460
I99	7172	1792	8.7	34.9	30-40	92380
I100	7709	1920	7.7	34.9	30-40	100915

Table 3.9: Instance Groups V-X features

All instances are publicly available at `http://www.or.deis.unibo.it/research.html`

3.6.2 Global procedure bound comparison

Initially we want to evaluate the performance of the proposed reformulation and compare it with the cut and column generation procedure described in Section 3.4.1 and 3.4.2 . Table 3.10 gives a general idea of the main results. The first column of the table is the instance name, in the second and third column we report the time that CPLEX12.2 needs to solve the continuous relaxation of the $TRAD - TKP$ formulation and the optimality gap of the corresponding upper bound with respect to the optimal solution. Columns four and five report the time spent by CPLEX12.2 to solve the root node of the $TRAD - TKP$ formulation and the optimality gap of the corresponding upper bound with respect to the optimal solution. This means that several families of general cuts are added by CPLEX to obtain a tighter upper bound. The sixth and seventh columns reports the computing time and the gap obtained by solving through column generation the root node of the $REF - TKP$ formulation, where constraints are grouped by 16 at the time. The last 2 columns report the computing time and the gap obtained by solving the root node of the formulation which explicitly adds cuts derived by grouping constraints 16 at the time. According to our computational experience this is the best configuration, which gives on average the best compromise between computing time and quality of the bound. All results were obtained with a time limit of 1 hour of computing time; note that when the time limit is reached (TL in the table), the cut formulation still provides a valid upper bound. It is clearly shown by the results that the explicit cut generation procedure is not competitive for the TKP, while the continuous relaxation of the $REF - TKP$ model provides an improved upper bound on the $TRAD - TKP$ model.

3.6.3 Reformulation and Cut Generation

For the same subset of instances, in Table 3.11 we investigate the performance of the $REF - TKP$ cut generation model for different groupings of the constraints, i.e., we consider different cardinalities for the sets I_h, $h = i, \ldots, g$. For $|I_h| = \{1, 2, 4, 8, 16, 32, 64\}$ we report the time needed to solve the continuous relaxation of the corresponding $REF - TKP$ through cut generation and the optimality gap between the optimal solution and the obtained upper bound. We do not report the time for constraint group of 64 because the procedure always goes in TL. In Table 3.12 we present the comparison of bounds obtained by different block dimensions, more in detail the table shows the absolute difference of bound achievable grouping by 1-2, 2-4, 4-8, 8-16, 16-32 and 32-64. In Table 3.13 we present the same comparison of the previous table but comparing the computing times. These tables show that grouping by 2 not only dominates grouping by 1 in terms of bounds but also in terms of computing time. In Table 3.14 we present the absolute improvement on the $TRAD - TKP$ continuous relaxation. In Table 3.15 we present the improvement on original $TRAD - TKP$ continuous relaxation per second. In Table 3.16 we present the absolute improvement on $TRAD - TKP$ root node. Generally speaking, as expected, by increasing the size of the groups of constraints the obtained optimality gap is reduced. The computing time is smaller for very small groupings (1 or 2 constraints in each group), because in this case during the cut generation there is a large number of (easy) subproblems to be solved but they are easy. By increasing the size of the groups the overall computing time increases, in that there are few subproblems to solve, but very hard ones. For these instances it is not convenient grouping more than 64 overlapping constraints in that even if the potential bounds is better the procedure requires too much time and the bounds at time limits are not strong enough.

	t_{C-T}	gap	t_{R-T}	gap	$t_{COL_{16}}$	gap	$t_{CUT_{16}}$	gap
I41	0.01	14.02	1.48	1.07	11.43	0.07	2055.27	0.07
I42	0.01	13.62	1.97	1.10	15.46	0.15	2405.59	0.15
I43	0.01	13.50	1.74	1.15	14.55	0.14	2923.17	0.14
I44	0.01	13.08	2.18	0.94	22.5	0.20	2875.49	0.20
I45	0.02	13.58	2.72	1.29	22.33	0.20	2994.7	0.20
I46	0.02	13.63	2.77	1.03	19.89	0.09	3036.36	0.09
I47	0.02	14.04	3.04	0.92	21.23	0.13	TL	0.13
I48	0.02	13.25	3.83	1.11	23.08	0.09	TL	0.16
I49	0.03	13.40	4.01	1.27	29.03	0.09	TL	0.16
I50	0.03	13.36	4.73	1.25	30.27	0.05	TL	0.17
I51	0.02	11.56	3.61	0.68	24.08	0.04	TL	2.64
I52	0.02	12.40	3.96	0.54	31.68	0.07	TL	2.85
I53	0.02	11.88	4.13	0.45	43.09	0.02	TL	3.16
I54	0.02	11.71	4.39	0.68	32.48	0.04	TL	3.41
I55	0.03	12.24	4.53	0.63	62.42	0.02	TL	3.36
I56	0.03	12.21	5.79	0.58	54.7	0.09	TL	3.65
I57	0.03	12.31	6.55	0.59	57.17	0.05	TL	3.52
I58	0.04	11.96	7.20	0.82	47.9	0.06	TL	3.55
I59	0.04	12.24	10.08	0.69	57.16	0.08	TL	3.77
I60	0.04	11.44	11.76	0.61	65.14	0.07	TL	3.91
I61	0.01	10.51	1.07	0.09	4.04	0.06	451.18	0.06
I62	0.01	10.20	0.91	0.14	5.37	0.01	412.58	0.01
I63	0.01	9.64	1.13	0.12	5.36	0.00	524.71	0.00
I64	0.01	9.80	1.24	0.15	7.39	0.01	679.41	0.01
I65	0.02	10.06	1.50	0.10	7.22	0.01	657.11	0.01
I66	0.02	10.16	2.36	0.05	11.11	0.05	963.01	0.05
I67	0.02	9.44	1.76	0.09	10.55	0.00	883.5	0.00
I68	0.02	9.68	1.78	0.05	10.5	0.02	1123	0.02
I69	0.02	9.39	1.41	0.14	11.86	0.02	1178.46	0.02
I70	0.03	10.12	2.13	0.12	12.53	0.03	1108.91	0.03
I71	0.01	11.32	1.26	0.75	13.25	0.10	3342.02	0.10
I72	0.02	11.70	1.78	0.88	11.53	0.07	TL	0.07
I73	0.02	12.14	1.97	0.74	19.38	0.06	TL	0.13
I74	0.01	11.69	2.53	0.53	12.71	0.07	TL	0.13
I75	0.02	12.11	2.63	0.79	16.12	0.11	TL	0.35
I76	0.03	11.39	3.08	0.49	19.21	0.13	TL	0.43
I77	0.02	11.32	3.02	0.68	18.85	0.09	TL	0.38
I78	0.02	11.84	4.17	0.69	33.89	0.12	TL	0.40
I79	0.03	11.60	4.89	0.58	26.08	0.08	TL	0.54
I80	0.02	11.53	5.96	0.75	32.82	0.07	TL	0.54
I81	0.02	11.69	2.67	0.54	16.57	0.05	TL	2.15
I82	0.02	11.23	2.99	0.50	28.26	0.03	TL	2.24
I83	0.02	10.84	2.78	0.38	24.09	0.03	TL	2.10
I84	0.02	10.29	3.57	0.43	28.23	0.02	TL	2.49
I85	0.03	11.55	4.53	0.41	65.35	0.05	TL	2.70
I86	0.03	10.99	5.85	0.54	45.76	0.01	TL	2.86
I87	0.03	10.68	7.00	0.64	42.94	0.06	TL	2.86
I88	0.04	10.94	7.35	0.53	51.67	0.08	TL	2.94
I89	0.04	10.75	8.46	0.44	48.62	0.06	TL	3.20
I90	0.04	11.39	11.09	0.66	54.27	0.05	TL	3.62
I91	0.01	11.95	1.82	0.79	10.75	0.07	TL	0.15
I92	0.01	11.15	1.72	0.96	29.76	0.03	TL	0.18
I93	0.02	11.54	2.09	0.79	14.9	0.09	TL	0.28
I94	0.02	12.11	2.50	1.15	25.38	0.08	TL	0.37
I95	0.03	10.96	3.01	0.75	21.02	0.03	TL	0.43
I96	0.03	11.41	5.21	0.91	24.45	0.11	TL	0.54
I97	0.03	11.82	3.78	0.75	33.9	0.07	TL	0.57
I98	0.03	11.97	4.42	0.98	36.59	0.13	TL	0.81
I99	0.03	12.29	5.15	0.85	33.76	0.07	TL	0.75
I100	0.03	11.64	5.22	0.95	41.22	0.13	TL	1.02
	0.02	**11.64**	**3.80**	**0.65**	**27.55**	**0.07**	**1624.38**	**1.18**

Table 3.10: Comparison of the continuous relaxations (without and with cuts) of *TRAD-TKP* and *REF-TKP* with explicit cut and column generations; 1 hour of computing time and groups of 16 overlapping constraints.

	t_{CUT_1}	gap	t_{CUT_2}	gap	t_{CUT_4}	gap	t_{CUT_8}	gap	$t_{CUT_{16}}$	gap	$t_{CUT_{32}}$	gap	$gap_{CUT_{64}}$
I41	116	2.09	128	1.27	168	0.63	477	0.34	2055	0.07	TL	1.37	5.20
I42	226	2.54	190	1.67	237	0.79	581	0.37	2406	0.15	TL	1.98	6.63
I43	307	2.05	234	1.34	308	0.77	711	0.38	2923	0.14	TL	2.03	6.09
I44	301	2.03	223	1.34	295	0.73	662	0.35	2875	0.20	TL	2.18	6.02
I45	373	2.37	290	1.67	341	0.97	764	0.40	2995	0.20	TL	2.31	6.82
I46	408	1.87	312	1.30	342	0.70	828	0.22	3036	0.09	TL	2.20	6.26
I47	460	1.69	349	1.16	388	0.60	824	0.26	TL	0.13	TL	2.42	6.68
I48	544	2.08	401	1.39	467	0.72	951	0.34	TL	0.16	TL	2.61	6.85
I49	610	2.09	432	1.37	502	0.68	1141	0.25	TL	0.16	TL	2.87	7.37
I50	677	2.06	516	1.30	627	0.60	1338	0.24	TL	0.17	TL	2.56	6.46
I51	272	1.32	393	0.51	1123	0.18	TL	0.33	TL	2.64	TL	6.52	9.43
I52	335	1.39	465	0.78	1288	0.29	TL	0.38	TL	2.85	TL	6.55	10.01
I53	387	1.20	517	0.52	1658	0.25	TL	0.57	TL	3.16	TL	7.03	10.00
I54	505	1.35	689	0.65	1789	0.28	TL	0.83	TL	3.41	TL	7.17	10.18
I55	526	1.18	675	0.50	1993	0.19	TL	0.74	TL	3.36	TL	7.29	10.25
I56	593	1.27	750	0.61	2057	0.26	TL	0.95	TL	3.65	TL	7.47	10.38
I57	681	1.24	937	0.56	2404	0.25	TL	0.88	TL	3.52	TL	7.22	10.54
I58	724	1.47	897	0.69	2546	0.33	TL	0.80	TL	3.55	TL	7.33	10.22
I59	767	1.52	983	0.70	2616	0.32	TL	0.75	TL	3.77	TL	7.69	10.47
I60	893	1.38	1066	0.61	3061	0.30	TL	1.11	TL	3.91	TL	7.72	10.18
I61	123	0.21	86	0.16	105	0.13	154	0.06	451	0.06	2326	0.00	1.18
I62	136	0.14	92	0.11	107	0.03	152	0.02	413	0.01	2174	0.00	1.26
I63	170	0.21	128	0.16	135	0.12	195	0.00	525	0.00	3030	0.00	1.21
I64	195	0.20	150	0.14	149	0.11	222	0.01	679	0.01	3420	0.00	1.24
I65	268	0.23	180	0.16	171	0.05	257	0.04	657	0.01	TL	0.02	1.17
I66	294	0.20	216	0.13	225	0.07	359	0.05	963	0.05	TL	0.06	1.55
I67	350	0.22	226	0.13	214	0.03	316	0.00	884	0.00	TL	0.06	1.87
I68	370	0.14	282	0.11	253	0.05	368	0.02	1123	0.02	TL	0.09	1.74
I69	464	0.23	315	0.12	280	0.08	409	0.03	1178	0.02	TL	0.13	2.02
I70	492	0.20	321	0.12	287	0.08	402	0.06	1109	0.03	TL	0.14	1.79
I71	231	1.38	189	0.96	352	0.53	808	0.13	3342	0.10	TL	1.82	5.52
I72	279	1.59	227	1.06	414	0.57	1022	0.22	TL	0.07	TL	2.39	6.00
I73	323	1.28	331	0.82	509	0.38	1027	0.15	TL	0.13	TL	2.36	6.40
I74	356	1.42	309	0.91	456	0.32	988	0.17	TL	0.13	TL	2.47	6.61
I75	393	1.61	361	1.06	577	0.47	1476	0.19	TL	0.35	TL	3.01	6.92
I76	474	1.19	429	0.84	762	0.40	1710	0.22	TL	0.43	TL	3.26	6.89
I77	573	1.43	457	0.97	693	0.43	1629	0.16	TL	0.38	TL	3.06	6.76
I78	628	1.41	518	0.92	745	0.55	1472	0.23	TL	0.40	TL	3.14	6.95
I79	725	1.38	600	0.97	767	0.52	1668	0.21	TL	0.54	TL	3.35	7.06
I80	781	1.43	521	0.93	617	0.48	1801	0.16	TL	0.54	TL	3.17	6.89
I81	317	1.24	298	0.58	791	0.30	TL	0.09	TL	2.15	TL	5.72	8.82
I82	317	1.04	454	0.64	1077	0.30	TL	0.19	TL	2.24	TL	5.75	9.01
I83	393	0.95	543	0.47	1144	0.20	TL	0.15	TL	2.10	TL	5.88	8.66
I84	494	1.03	677	0.48	1426	0.20	TL	0.22	TL	2.49	TL	5.84	8.48
I85	554	0.92	695	0.45	1708	0.19	TL	0.39	TL	2.70	TL	6.32	9.18
I86	628	1.12	837	0.48	1642	0.17	TL	0.32	TL	2.86	TL	6.55	9.28
I87	681	1.04	819	0.56	2059	0.26	TL	0.56	TL	2.86	TL	6.38	9.07
I88	756	1.11	986	0.55	2118	0.29	TL	0.59	TL	2.94	TL	6.72	9.20
I89	845	1.04	1096	0.54	2000	0.28	TL	0.78	TL	3.20	TL	6.67	9.22
I90	1000	1.18	1237	0.60	2756	0.28	TL	0.87	TL	3.62	TL	7.09	9.58
I91	329	1.87	301	1.23	373	0.49	954	0.17	TL	0.15	TL	2.35	5.98
I92	339	1.41	362	0.88	467	0.39	1174	0.12	TL	0.18	TL	2.31	6.16
I93	429	1.34	347	0.98	468	0.52	1210	0.26	TL	0.28	TL	2.64	6.50
I94	500	1.93	390	1.16	597	0.52	1362	0.20	TL	0.37	TL	2.87	6.82
I95	632	1.32	424	0.87	692	0.33	1740	0.09	TL	0.43	TL	3.13	6.80
I96	670	1.39	575	1.07	696	0.62	1644	0.26	TL	0.54	TL	3.13	6.64
I97	656	1.44	622	0.90	764	0.48	1853	0.20	TL	0.57	TL	3.19	7.05
I98	744	1.58	674	1.01	894	0.56	2244	0.30	TL	0.81	TL	3.47	7.07
I99	868	1.53	753	0.98	922	0.53	2262	0.26	TL	0.75	TL	3.57	7.37
I100	973	1.68	475	1.09	721	0.48	2709	0.23	TL	1.02	TL	4.05	7.58
	491	**1.27**	**483**	**0.77**	**922**	**0.38**	**1047**	**0.31**	**1624**	**1.18**	**2738**	**3.61**	**6.76**

Table 3.11: Comparison of the explicit cut generations models with different dimensions; 1 hour of computing time.

$\mathbf{b_{CUT_{1-2}}}$	$\mathbf{b_{CUT_{2-4}}}$	$\mathbf{b_{CUT_{4-8}}}$	$\mathbf{b_{CUT_{8-16}}}$	$\mathbf{b_{CUT_{16-32}}}$	$\mathbf{b_{CUT_{32-64}}}$
262.7	202.6	91.2	80.9	-404.3	-1264.3
323.9	324.9	152.5	78.1	-670.3	-1817.5
298.6	240.8	159.5	100.5	-790.4	-1810.0
327.6	290.5	176.4	67.2	-935.2	-1927.9
368.7	359.9	288.8	105.3	-1091.2	-2491.0
327.7	343.4	266.6	73.8	-1198.9	-2456.0
325.2	339.8	200.1	80.3	-1391.9	-2777.4
471.4	447.9	252.7	116.8	-1651.9	-3061.3
519.6	491.7	298.9	61.1	-1943.0	-3476.1
593.8	543.3	274.0	52.9	-1861.1	-3238.0
595.1	242.6	-110.2	-1710.7	-3073.4	-2476.5
507.7	408.2	-75.8	-2089.1	-3337.3	-3370.0
674.7	270.6	-313.6	-2617.3	-4168.5	-3451.5
762.8	411.1	-602.5	-2895.0	-4502.2	-3885.5
831.3	377.3	-664.1	-3299.4	-5283.4	-4279.6
868.9	456.2	-902.4	-3641.4	-5528.0	-4542.3
976.6	449.3	-904.0	-3930.1	-5894.4	-5695.0
1217.9	549.8	-728.2	-4357.4	-6393.7	-5269.5
1379.0	645.0	-722.8	-5227.7	-7283.8	-5544.7
1420.0	589.6	-1514.6	-5376.4	-7857.6	-5433.5
9.5	7.4	15.5	0.0	13.5	-263.8
9.1	21.7	2.3	2.0	2.5	-333.3
13.4	13.5	34.0	0.0	0.0	-357.4
18.0	9.2	35.4	0.5	1.7	-410.3
24.5	40.1	6.0	7.7	-0.8	-432.2
29.1	22.2	9.2	0.0	-5.6	-600.8
41.4	42.3	14.3	1.0	-26.9	-827.5
16.7	29.8	10.8	3.0	-37.0	-805.7
58.3	20.5	23.6	6.0	-55.5	-976.9
44.6	20.1	14.1	15.8	-62.9	-918.2
175.4	178.8	165.0	13.3	-719.8	-1632.2
260.3	240.5	170.6	68.9	-1139.6	-1882.5
246.2	232.2	122.9	8.6	-1198.3	-2320.1
313.5	355.9	92.9	24.3	-1440.4	-2713.0
372.8	393.2	187.9	-107.9	-1822.9	-2866.7
270.0	330.5	140.1	-159.6	-2208.7	-3024.0
389.1	447.5	223.8	-185.9	-2275.5	-3351.8
423.7	323.9	274.6	-148.2	-2425.3	-3601.4
397.2	438.2	295.6	-321.4	-2776.9	-3919.7
520.1	456.6	316.3	-376.7	-2740.2	-4118.5
479.7	205.4	144.8	-1502.3	-2765.8	-2575.0
339.1	277.8	95.5	-1741.3	-3160.0	-3158.8
463.6	264.8	47.0	-1917.8	-3943.8	-3108.2
611.8	313.4	-29.3	-2569.8	-4014.0	-3380.0
571.1	311.5	-242.6	-2836.3	-4746.9	-4005.2
844.5	390.1	-187.8	-3357.0	-5216.8	-4125.2
681.3	438.0	-436.0	-3378.5	-5499.8	-4484.2
872.7	414.4	-468.8	-3775.8	-6463.8	-4531.6
844.7	446.0	-847.2	-4231.1	-6453.6	-5043.3
1034.0	570.1	-1047.6	-5092.4	-6863.7	-5241.6
281.6	324.7	134.0	8.0	-961.0	-1686.1
252.2	228.9	125.4	-26.1	-1016.9	-1956.9
200.2	254.7	140.2	-11.7	-1324.3	-2304.8
486.3	393.2	196.0	-101.3	-1568.3	-2655.3
319.1	370.3	164.6	-229.3	-1921.1	-2780.7
234.5	330.7	262.9	-198.8	-1948.1	-2803.5
440.4	344.5	223.0	-298.8	-2178.9	-3436.1
519.0	401.4	231.9	-450.3	-2461.6	-3553.4
515.0	427.8	253.6	-465.5	-2722.6	-3922.2
613.9	631.8	247.5	-807.1	-3217.8	-4016.2
454.9	**315.8**	**-53.6**	**-1141.0**	**-2543.8**	**-2873.2**

Table 3.12: Cut generation: comparison of bounds obtained by different block dimensions.

$t_{CUT_{1-2}}$	$t_{CUT_{2-4}}$	$t_{CUT_{4-8}}$	$t_{CUT_{8-16}}$	$t_{CUT_{16-32}}$	$t_{CUT_{32-64}}$
11.5	40.2	308.8	1578.7	*	*
-35.7	47.7	343.7	1824.4	*	*
-73.4	74.4	403.1	2212.1	*	*
-78.6	72.2	367.1	2213.5	*	*
-82.8	50.7	422.9	2231.1	*	*
-95.7	30.6	485.1	2208.8	*	*
-111.0	38.9	436.7	2770.5	*	*
-142.1	65.7	484.0	2649.0	*	*
-178.3	69.9	639.4	2459.9	*	*
-160.5	110.9	710.5	2263.5	*	*
120.8	729.3	*	*	*	*
130.5	822.5	*	*	*	*
130.5	1141.2	*	*	*	*
183.5	1100.2	*	*	*	*
148.5	1318.0	*	*	*	*
157.1	1307.0	*	*	*	*
256.4	1467.2	*	*	*	*
173.6	1648.6	*	*	*	*
215.7	1633.6	*	*	*	*
173.0	1994.5	*	*	*	*
-37.6	19.6	48.6	297.5	1874.4	*
-43.5	15.4	44.9	260.3	1761.9	*
-42.1	7.4	60.1	329.5	2505.8	*
-45.7	-1.0	73.1	457.5	2740.2	*
-87.6	-9.4	86.1	400.5	*	*
-78.2	8.6	133.8	604.4	*	*
-123.9	-12.4	101.7	567.7	*	*
-87.5	-29.4	114.9	755.2	*	*
-149.3	-34.7	128.8	769.6	*	*
-170.8	-34.1	115.1	706.5	*	*
-42.4	163.4	455.8	2533.8	*	*
-52.3	186.7	608.2	2580.0	*	*
8.1	178.1	518.3	2574.9	*	*
-47.1	147.7	531.5	2615.1	*	*
-32.0	215.3	899.0	2125.2	*	*
-44.7	332.9	947.8	1891.0	*	*
-115.4	236.2	935.9	1972.5	*	*
-110.2	227.8	727.1	2130.9	*	*
-125.1	167.3	900.6	1932.8	*	*
-259.8	95.5	1184.0	1801.0	*	*
-18.6	492.5	*	*	*	*
137.1	622.3	*	*	*	*
150.4	600.7	*	*	*	*
183.0	748.7	*	*	*	*
141.3	1013.0	*	*	*	*
208.7	805.4	*	*	*	*
137.5	1240.0	*	*	*	*
230.5	1132.1	*	*	*	*
250.6	904.3	*	*	*	*
237.1	1518.2	*	*	*	*
-27.5	72.3	580.3	2653.4	*	*
22.8	104.7	706.9	2428.2	*	*
-81.9	120.8	741.9	2390.3	*	*
-109.6	206.9	765.5	2241.5	*	*
-208.0	267.3	1048.5	1860.2	*	*
-94.2	121.0	947.3	1957.3	*	*
-33.3	141.8	1088.8	1748.4	*	*
-69.6	220.1	1350.0	1356.0	*	*
-114.4	168.4	1340.4	1338.0	*	*
-498.1	245.7	1988.8	891.1	*	*
-8.3	**439.8**	**975.6**	**1143.5**	**506.4**	*****

Table 3.13: Cut generation: comparison of times obtained by different block dimensions.

$\mathbf{impr_{CUT_1}}$	$\mathbf{impr_{CUT_2}}$	$\mathbf{impr_{CUT_4}}$	$\mathbf{impr_{CUT_8}}$	$\mathbf{impr_{CUT_{16}}}$	$\mathbf{impr_{CUT_{32}}}$	$\mathbf{impr_{CUT_{64}}}$
12.18%	12.92%	13.48%	13.73%	13.96%	12.83%	9.31%
11.37%	12.15%	12.93%	13.30%	13.49%	11.87%	7.48%
11.69%	12.33%	12.83%	13.17%	13.38%	11.71%	7.89%
11.28%	11.89%	12.44%	12.77%	12.90%	11.14%	7.51%
11.49%	12.12%	12.74%	13.23%	13.41%	11.54%	7.26%
11.98%	12.49%	13.02%	13.44%	13.55%	11.68%	7.86%
12.56%	13.03%	13.52%	13.81%	13.93%	11.91%	7.88%
11.41%	12.03%	12.62%	12.96%	13.11%	10.92%	6.87%
11.55%	12.20%	12.81%	13.18%	13.26%	10.84%	6.51%
11.54%	12.22%	12.84%	13.15%	13.21%	11.09%	7.38%
10.37%	11.11%	11.40%	11.27%	9.17%	5.39%	2.35%
11.17%	11.72%	12.15%	12.07%	9.84%	6.27%	2.66%
10.80%	11.42%	11.66%	11.38%	9.00%	5.21%	2.08%
10.50%	11.13%	11.47%	10.97%	8.59%	4.90%	1.70%
11.19%	11.80%	12.07%	11.59%	9.18%	5.34%	2.22%
11.07%	11.67%	11.98%	11.36%	8.89%	5.12%	2.03%
11.21%	11.81%	12.09%	11.53%	9.11%	5.49%	1.98%
10.65%	11.35%	11.67%	11.25%	8.72%	5.00%	1.94%
10.89%	11.62%	11.97%	11.58%	8.80%	4.93%	1.98%
10.20%	10.89%	11.18%	10.44%	7.84%	4.03%	1.40%
10.32%	10.36%	10.39%	10.46%	10.46%	10.51%	9.44%
10.07%	10.10%	10.18%	10.18%	10.19%	10.20%	9.05%
9.45%	9.49%	9.53%	9.64%	9.64%	9.64%	8.53%
9.62%	9.67%	9.69%	9.79%	9.79%	9.80%	8.67%
9.85%	9.91%	10.01%	10.03%	10.04%	10.04%	8.99%
9.98%	10.05%	10.10%	10.12%	10.12%	10.11%	8.74%
9.24%	9.33%	9.41%	9.44%	9.44%	9.39%	7.71%
9.55%	9.58%	9.63%	9.66%	9.66%	9.59%	8.08%
9.18%	9.28%	9.32%	9.36%	9.37%	9.27%	7.52%
9.94%	10.01%	10.04%	10.07%	10.09%	9.99%	8.48%
10.08%	10.46%	10.85%	11.20%	11.23%	9.67%	6.14%
10.28%	10.75%	11.20%	11.51%	11.64%	9.54%	6.07%
11.00%	11.41%	11.80%	12.01%	12.02%	10.01%	6.13%
10.41%	10.87%	11.40%	11.54%	11.57%	9.45%	5.44%
10.68%	11.17%	11.70%	11.95%	11.80%	9.38%	5.58%
10.32%	10.64%	11.03%	11.20%	11.01%	8.40%	4.83%
10.03%	10.46%	10.94%	11.18%	10.98%	8.52%	4.89%
10.58%	11.02%	11.35%	11.64%	11.48%	8.98%	5.25%
10.36%	10.73%	11.14%	11.42%	11.12%	8.53%	4.89%
10.24%	10.70%	11.11%	11.39%	11.06%	8.63%	4.99%
10.58%	11.17%	11.42%	11.60%	9.75%	6.33%	3.14%
10.30%	10.67%	10.96%	11.07%	9.20%	5.82%	2.44%
9.99%	10.42%	10.67%	10.71%	8.93%	5.27%	2.39%
9.36%	9.86%	10.12%	10.09%	8.00%	4.73%	1.98%
10.73%	11.15%	11.38%	11.20%	9.10%	5.58%	2.61%
9.98%	10.56%	10.84%	10.70%	8.37%	4.75%	1.89%
9.75%	10.18%	10.45%	10.18%	8.05%	4.59%	1.78%
9.95%	10.45%	10.69%	10.42%	8.24%	4.52%	1.92%
9.82%	10.27%	10.50%	10.05%	7.80%	4.37%	1.69%
10.33%	10.85%	11.14%	10.61%	8.06%	4.62%	2.00%
10.27%	10.85%	11.52%	11.80%	11.81%	9.83%	6.35%
9.89%	10.37%	10.81%	11.04%	11.00%	9.05%	5.32%
10.34%	10.66%	11.08%	11.31%	11.29%	9.14%	5.39%
10.38%	11.08%	11.65%	11.94%	11.79%	9.52%	5.68%
9.76%	10.18%	10.66%	10.88%	10.58%	8.08%	4.46%
10.16%	10.45%	10.85%	11.17%	10.93%	8.55%	5.11%
10.54%	11.02%	11.40%	11.65%	11.32%	8.92%	5.14%
10.56%	11.07%	11.47%	11.70%	11.26%	8.81%	5.27%
10.93%	11.42%	11.82%	12.06%	11.62%	9.04%	5.31%
10.12%	10.66%	11.22%	11.43%	10.73%	7.91%	4.39%
10.50%	**10.95%**	**11.31%**	**11.36%**	**10.57%**	**8.27%**	**5.17%**

Table 3.14: Cut generation: improvement on the $TRAD - TKP$ continuous relaxation.

$\mathbf{impr_{CUT_1}}$	$\mathbf{impr_{CUT_2}}$	$\mathbf{impr_{CUT_4}}$	$\mathbf{impr_{CUT_8}}$	$\mathbf{impr_{CUT_{16}}}$	$\mathbf{impr_{CUT_{32}}}$	$\mathbf{impr_{CUT_{64}}}$
0.105%	0.101%	0.106%	0.029%	0.007%	0.004%	0.003%
0.050%	0.064%	0.068%	0.023%	0.006%	0.003%	0.002%
0.038%	0.053%	0.055%	0.019%	0.005%	0.003%	0.002%
0.037%	0.053%	0.056%	0.019%	0.004%	0.003%	0.002%
0.031%	0.042%	0.044%	0.017%	0.004%	0.003%	0.002%
0.029%	0.040%	0.042%	0.016%	0.004%	0.003%	0.002%
0.027%	0.037%	0.039%	0.017%	0.004%	0.003%	0.002%
0.021%	0.030%	0.031%	0.014%	0.004%	0.003%	0.002%
0.019%	0.028%	0.030%	0.012%	0.004%	0.003%	0.002%
0.017%	0.024%	0.025%	0.010%	0.004%	0.003%	0.002%
0.038%	0.028%	0.029%	0.003%	0.003%	0.001%	0.001%
0.033%	0.025%	0.026%	0.003%	0.003%	0.002%	0.001%
0.028%	0.022%	0.023%	0.003%	0.002%	0.001%	0.001%
0.021%	0.016%	0.017%	0.003%	0.002%	0.001%	0.000%
0.021%	0.017%	0.018%	0.003%	0.003%	0.001%	0.001%
0.019%	0.016%	0.016%	0.003%	0.002%	0.001%	0.001%
0.016%	0.013%	0.013%	0.003%	0.003%	0.002%	0.001%
0.015%	0.013%	0.013%	0.003%	0.002%	0.001%	0.001%
0.014%	0.012%	0.012%	0.003%	0.002%	0.001%	0.001%
0.011%	0.010%	0.010%	0.003%	0.002%	0.001%	0.000%
0.084%	0.121%	0.122%	0.068%	0.023%	0.005%	0.003%
0.074%	0.110%	0.111%	0.067%	0.025%	0.005%	0.003%
0.056%	0.074%	0.075%	0.049%	0.018%	0.003%	0.002%
0.049%	0.065%	0.065%	0.044%	0.014%	0.003%	0.002%
0.037%	0.055%	0.056%	0.039%	0.015%	0.003%	0.002%
0.034%	0.046%	0.047%	0.028%	0.011%	0.003%	0.002%
0.026%	0.041%	0.042%	0.030%	0.011%	0.003%	0.002%
0.026%	0.034%	0.034%	0.026%	0.009%	0.003%	0.002%
0.020%	0.029%	0.030%	0.023%	0.008%	0.003%	0.002%
0.020%	0.031%	0.031%	0.025%	0.009%	0.003%	0.002%
0.044%	0.055%	0.057%	0.014%	0.003%	0.003%	0.002%
0.037%	0.047%	0.049%	0.011%	0.003%	0.003%	0.002%
0.034%	0.034%	0.036%	0.012%	0.003%	0.003%	0.002%
0.029%	0.035%	0.037%	0.012%	0.003%	0.003%	0.002%
0.027%	0.031%	0.032%	0.008%	0.003%	0.003%	0.002%
0.022%	0.025%	0.026%	0.007%	0.003%	0.002%	0.001%
0.018%	0.023%	0.024%	0.007%	0.003%	0.002%	0.001%
0.017%	0.021%	0.022%	0.008%	0.003%	0.002%	0.001%
0.014%	0.018%	0.019%	0.007%	0.003%	0.002%	0.001%
0.013%	0.021%	0.021%	0.006%	0.003%	0.002%	0.001%
0.033%	0.037%	0.038%	0.003%	0.003%	0.002%	0.001%
0.032%	0.023%	0.024%	0.003%	0.003%	0.002%	0.001%
0.025%	0.019%	0.020%	0.003%	0.002%	0.001%	0.001%
0.019%	0.015%	0.015%	0.003%	0.002%	0.001%	0.001%
0.019%	0.016%	0.016%	0.003%	0.003%	0.002%	0.001%
0.016%	0.013%	0.013%	0.003%	0.002%	0.001%	0.001%
0.014%	0.012%	0.013%	0.003%	0.002%	0.001%	0.000%
0.013%	0.011%	0.011%	0.003%	0.002%	0.001%	0.001%
0.012%	0.009%	0.010%	0.003%	0.002%	0.001%	0.000%
0.010%	0.009%	0.009%	0.003%	0.002%	0.001%	0.001%
0.031%	0.036%	0.038%	0.012%	0.003%	0.003%	0.002%
0.029%	0.029%	0.030%	0.009%	0.003%	0.003%	0.001%
0.024%	0.031%	0.032%	0.009%	0.003%	0.003%	0.001%
0.021%	0.028%	0.030%	0.009%	0.003%	0.003%	0.002%
0.015%	0.024%	0.025%	0.006%	0.003%	0.002%	0.001%
0.015%	0.018%	0.019%	0.007%	0.003%	0.002%	0.001%
0.016%	0.018%	0.018%	0.006%	0.003%	0.002%	0.001%
0.014%	0.016%	0.017%	0.005%	0.003%	0.002%	0.001%
0.013%	0.015%	0.016%	0.005%	0.003%	0.003%	0.001%
0.010%	0.022%	0.024%	0.004%	0.003%	0.002%	0.001%
0.028%	**0.033%**	**0.034%**	**0.013%**	**0.005%**	**0.002%**	**0.001%**

Table 3.15: Cut generation: improvement on original $TRAD - TKP$ continuous relaxation per second.

$\mathbf{impr_{CUT_1}}$	$\mathbf{impr_{CUT_2}}$	$\mathbf{impr_{CUT_4}}$	$\mathbf{impr_{CUT_8}}$	$\mathbf{impr_{CUT_{16}}}$	$\mathbf{impr_{CUT_{32}}}$	$\mathbf{impr_{CUT_{64}}}$
0.00%	0.00%	0.45%	0.74%	1.00%	0.00%	0.00%
0.00%	0.00%	0.31%	0.73%	0.95%	0.00%	0.00%
0.00%	0.00%	0.39%	0.77%	1.01%	0.00%	0.00%
0.00%	0.00%	0.21%	0.59%	0.74%	0.00%	0.00%
0.00%	0.00%	0.32%	0.89%	1.09%	0.00%	0.00%
0.00%	0.00%	0.34%	0.81%	0.94%	0.00%	0.00%
0.00%	0.00%	0.32%	0.66%	0.79%	0.00%	0.00%
0.00%	0.00%	0.39%	0.77%	0.95%	0.00%	0.00%
0.00%	0.00%	0.60%	1.02%	1.11%	0.00%	0.00%
0.00%	0.00%	0.66%	1.01%	1.08%	0.00%	0.00%
0.00%	0.17%	0.50%	0.35%	0.00%	0.00%	0.00%
0.00%	0.00%	0.25%	0.16%	0.00%	0.00%	0.00%
0.00%	0.00%	0.21%	0.00%	0.00%	0.00%	0.00%
0.00%	0.02%	0.40%	0.00%	0.00%	0.00%	0.00%
0.00%	0.13%	0.45%	0.00%	0.00%	0.00%	0.00%
0.00%	0.00%	0.32%	0.00%	0.00%	0.00%	0.00%
0.00%	0.03%	0.34%	0.00%	0.00%	0.00%	0.00%
0.00%	0.13%	0.49%	0.01%	0.00%	0.00%	0.00%
0.00%	0.00%	0.38%	0.00%	0.00%	0.00%	0.00%
0.00%	0.00%	0.31%	0.00%	0.00%	0.00%	0.00%
0.00%	0.00%	0.00%	0.03%	0.03%	0.09%	0.00%
0.00%	0.03%	0.11%	0.12%	0.13%	0.14%	0.00%
0.00%	0.00%	0.00%	0.12%	0.12%	0.12%	0.00%
0.00%	0.01%	0.04%	0.14%	0.14%	0.15%	0.00%
0.00%	0.00%	0.05%	0.06%	0.08%	0.08%	0.00%
0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
0.00%	0.00%	0.06%	0.09%	0.09%	0.03%	0.00%
0.00%	0.00%	0.01%	0.03%	0.04%	0.00%	0.00%
0.00%	0.02%	0.07%	0.11%	0.12%	0.01%	0.00%
0.00%	0.00%	0.04%	0.07%	0.09%	0.00%	0.00%
0.00%	0.00%	0.21%	0.61%	0.65%	0.00%	0.00%
0.00%	0.00%	0.31%	0.67%	0.81%	0.00%	0.00%
0.00%	0.00%	0.35%	0.59%	0.60%	0.00%	0.00%
0.00%	0.00%	0.21%	0.37%	0.41%	0.00%	0.00%
0.00%	0.00%	0.32%	0.60%	0.44%	0.00%	0.00%
0.00%	0.00%	0.09%	0.27%	0.06%	0.00%	0.00%
0.00%	0.00%	0.25%	0.53%	0.30%	0.00%	0.00%
0.00%	0.00%	0.15%	0.47%	0.29%	0.00%	0.00%
0.00%	0.00%	0.07%	0.38%	0.04%	0.00%	0.00%
0.00%	0.00%	0.28%	0.59%	0.22%	0.00%	0.00%
0.00%	0.00%	0.25%	0.45%	0.00%	0.00%	0.00%
0.00%	0.00%	0.20%	0.31%	0.00%	0.00%	0.00%
0.00%	0.00%	0.18%	0.23%	0.00%	0.00%	0.00%
0.00%	0.00%	0.24%	0.21%	0.00%	0.00%	0.00%
0.00%	0.00%	0.22%	0.02%	0.00%	0.00%	0.00%
0.00%	0.06%	0.37%	0.22%	0.00%	0.00%	0.00%
0.00%	0.07%	0.38%	0.08%	0.00%	0.00%	0.00%
0.00%	0.00%	0.24%	0.00%	0.00%	0.00%	0.00%
0.00%	0.00%	0.16%	0.00%	0.00%	0.00%	0.00%
0.00%	0.06%	0.38%	0.00%	0.00%	0.00%	0.00%
0.00%	0.00%	0.31%	0.62%	0.64%	0.00%	0.00%
0.00%	0.09%	0.57%	0.84%	0.78%	0.00%	0.00%
0.00%	0.00%	0.27%	0.53%	0.50%	0.00%	0.00%
0.00%	0.00%	0.63%	0.95%	0.79%	0.00%	0.00%
0.00%	0.00%	0.42%	0.66%	0.33%	0.00%	0.00%
0.00%	0.00%	0.29%	0.65%	0.38%	0.00%	0.00%
0.00%	0.00%	0.27%	0.55%	0.18%	0.00%	0.00%
0.00%	0.00%	0.42%	0.68%	0.17%	0.00%	0.00%
0.00%	0.00%	0.32%	0.59%	0.09%	0.00%	0.00%
0.00%	0.00%	0.47%	0.72%	0.00%	0.00%	0.00%
0.00%	**0.01%**	**0.28%**	**0.38%**	**0.30%**	**0.01%**	**0.00%**

Table 3.16: Cut generation: Improvement on $TRAD - TKP$ root node.

3.6.4 Reformulation and Column Generation

In table 3.17 and 3.18 we investigate the performance of the $REF-TKP$ column generation model for different groupings of the constraints, i.e., we consider different cardinalities for the sets I_h, $h = i, \ldots, g$. For $|I_h| = \{1, 2, 4, 8, 16, 32, 64, 128\}$ we report the time needed to solve the continuous relaxation of the corresponding $REF-TKP$ through column generation and the optimality gap between the optimal solution and the obtained upper bound. As expected, by increasing the size of the groups of constraints the obtained optimality gap is reduced. The computing time is larger for very small groupings (1 or 2 constraints in each group), because in this case during the column generation there is a large number of (easy) subproblems to be solved. By increasing the size of the groups the overall computing time is reduced, and then it increases again, when there are few subproblems to solve, but very hard ones. In Table 3.19 we present the comparison of bounds obtained by different block dimensions, more in detail the table shows the absolute difference of bound achievable grouping by 1-2, 2-4, 4-8, 8-16, 16-32, 32-64 and 64-128. In Table 3.20 we present the same comparison of the previous table but comparing the computing times. These tables show that grouping by 1 2 4 8 are dominated not only in terms of bounds but also in terms of computing time. In Table 3.21 we present the absolute improvement on the $TRAD-TKP$ continuous relaxation. In Table 3.22 we present the improvement on original $TRAD-TKP$ continuous relaxation per second. In Table 3.23 we present the absolute improvement on $TRAD-TKP$ root node.

3.6.5 Reformulation, Column Generation and Branch and Price

In the following we will consider only groupings of 32, 64 and 128 constraints, which represent a good compromise between computing time and quality of the obtained upper bound. As the previous tables have shown the computing time and the gap obtained by solving through column generation the root node of the $REF-TKP$ formulation where constraints are grouped by 32, 64 and 128 at the time, respectively prove the effectiveness of the proposed formulation $REF-TKP$. The reformulation is able to reduce the optimality gap between the optimal solution and the upper bound obtained at the root node of the column generation of around two orders of magnitude with respect to the root node of model $TRAD-TKP$ computed by CPLEX12.2. The cost of this improved bound, in terms of computing time, is between one and two orders of magnitude. A natural question is wether this computationally expensive improved bound pays when one wants to solve the TKP to optimality, i.e., wants to obtain an optimal *integer* solution. The question is investigated through the results reported in Table 3.24 where, for every instance, we report the optimality gap and the time needed for the $TRAD-TKP$ model through CPLEX12.2. When the time limit of 1 h of computing time is reached before optimality (TL), we report the current optimality gap (0 otherwise). In the following columns we report the time needed to solve to optimality the $TKP-REF$ model through Branch and Price for groupings of 32, 64 and 128 constraints, respectively. The table shows that it actually pays to spend some computational effort in obtaining a stronger continuous relaxation: CPLEX12.2 almost always runs into time limit before solving the considered problem, and the remaining gap is such that the solution of the problem in short time is unlikely. The only exceptions are the instances of group VII, which are easily solved also in the $TRAD-TKP$ formulation (in this group the profit of each item equals the cost, thus the problem is in this case a generalization of the Subset-Sum Problem, see Martello and Toth [50]). On the other side, the $REF-TKP$ model can be solved to optimality for the large part of the considered instances, through the straightforward branching scheme described in

Section 3.4.2. Only some instances could not be solved within the available computing time, and the corresponding optimality gap is extremely small. In few words, the advantage of computing a strong continuous relaxation is such that even a very simple Branch-and-Price algorithm can rule over the sophisticated Branch-and-Cut implemented in CPLEX12.2.

3.6.6 *Short Constraints*: Dynamic Programming

In this Section we present the result obtained with the Dynamic Programming algorithm described in Section 3.3. The considered instances (groups I-IV) have the same structure of those analyzed in the previous sections, but have constraints of smaller cardinality (namely 10, 15, 20 and 25 items appear in every constraint) and a larger number of constraints (starting from 2688 and increasing). Having few items per constraint is crucial for the Dynamic Programming algorithm, whose time complexity exponentially grows with the cardinality of the constraints (see Section 3.3). Table 3.25 compares the results from the $TRAD-TKP$ model, the Dynamic Programming Algorithm, and the $REF-TKP$. The maximum computing time for all the tests is 1 hour. The second column shows the optimal value z, which is used to compute all the percentage gaps. The third column reports the time and, in case of time limit, the percentage gap for the $TKP-TRAD$ solved with CPLEX12.1. The forth column shows the computing time t_{DP} of the Dynamic Programming Algorithm. Note that when the Dynamic Programming algorithm runs into time limit, there are no feasible solutions nor upper bounds available. The fifth, sixth and seventh columns show the computing time and, in case of time limit, the percentage gap for the $REF-TKP$ model with groups of 32, 64 and 128 constraints, respectively.

CPLEX12.1 can easily solve instances of the first group with the $TRAD-TKP$ formulation, but quickly gets into troubles as soon as the cardinality of the constraints grows. For instances with cardinality equal to 20 it always runs into time limit. For these instances the $REF-TKP$ model is still effective, in particular when constraints are grouped 128 at the time. Finally, we observe that the Dynamic Programming algorithm is very fast for instances with constraints of small cardinality, it behave similarly to the $REF-TKP$ model for constraints of cardinality 20, while it systematically runs into time limit if the cardinality of the constraints is 25 (or larger).

	t_{COL_1}	gap	t_{COL_2}	gap	t_{COL_4}	gap	t_{COL_8}	gap
I41	153	2.09	59	1.27	24	0.63	13	0.34
I42	241	2.54	72	1.67	28	0.79	23	0.37
I43	355	2.05	107	1.34	48	0.77	25	0.38
I44	353	2.03	106	1.34	41	0.73	26	0.35
I45	415	2.37	143	1.67	50	0.97	33	0.40
I46	497	1.87	149	1.30	47	0.70	37	0.22
I47	461	1.69	200	1.16	60	0.60	41	0.26
I48	588	2.08	189	1.39	75	0.72	48	0.34
I49	964	2.09	246	1.37	63	0.68	53	0.25
I50	761	2.06	328	1.30	83	0.60	81	0.24
I51	155	1.32	52	0.51	21	0.18	22	0.10
I52	129	1.39	57	0.78	24	0.29	22	0.14
I53	176	1.20	76	0.52	32	0.25	30	0.09
I54	205	1.35	101	0.65	45	0.28	33	0.12
I55	272	1.18	111	0.50	45	0.19	36	0.07
I56	273	1.27	116	0.61	39	0.26	55	0.12
I57	307	1.24	118	0.56	47	0.25	72	0.10
I58	452	1.47	133	0.69	50	0.33	48	0.11
I59	344	1.52	171	0.70	53	0.32	49	0.14
I60	403	1.38	174	0.61	76	0.30	80	0.14
I61	85	0.21	38	0.16	12	0.13	8	0.06
I62	67	0.14	31	0.11	14	0.03	12	0.02
I63	72	0.21	37	0.16	14	0.12	11	0.00
I64	112	0.20	45	0.14	21	0.11	13	0.01
I65	143	0.23	68	0.16	22	0.05	19	0.04
I66	221	0.20	82	0.13	38	0.07	20	0.05
I67	148	0.22	61	0.13	29	0.03	25	0.00
I68	247	0.14	84	0.11	33	0.05	20	0.02
I69	214	0.23	96	0.12	35	0.08	20	0.03
I70	331	0.20	106	0.12	40	0.08	25	0.06
I71	112	1.38	43	0.96	19	0.53	15	0.13
I72	159	1.59	78	1.06	26	0.57	19	0.22
I73	239	1.28	68	0.82	29	0.38	25	0.15
I74	178	1.42	77	0.91	36	0.32	20	0.17
I75	227	1.61	95	1.06	42	0.47	27	0.19
I76	322	1.19	111	0.84	46	0.40	34	0.22
I77	281	1.43	114	0.97	41	0.43	36	0.16
I78	349	1.41	148	0.92	49	0.55	45	0.23
I79	347	1.38	146	0.97	46	0.52	38	0.21
I80	436	1.43	184	0.93	66	0.48	39	0.16
I81	76	1.24	43	0.58	19	0.30	19	0.08
I82	106	1.04	49	0.64	21	0.30	23	0.11
I83	112	0.95	55	0.47	30	0.20	29	0.09
I84	148	1.03	81	0.48	35	0.20	31	0.09
I85	167	0.92	74	0.45	33	0.19	37	0.11
I86	195	1.12	114	0.48	36	0.17	37	0.05
I87	280	1.04	105	0.56	54	0.26	38	0.13
I88	246	1.11	108	0.55	57	0.29	52	0.14
I89	271	1.04	133	0.54	54	0.28	58	0.14
I90	336	1.18	160	0.60	58	0.28	60	0.10
I91	137	1.87	61	1.23	21	0.49	19	0.17
I92	177	1.41	85	0.88	31	0.39	30	0.12
I93	197	1.34	91	0.98	32	0.52	24	0.26
I94	328	1.93	166	1.16	40	0.52	41	0.20
I95	346	1.32	132	0.87	45	0.33	42	0.09
I96	334	1.39	148	1.07	47	0.62	36	0.26
I97	313	1.44	160	0.90	52	0.48	43	0.20
I98	541	1.58	201	1.01	59	0.56	75	0.30
I99	540	1.53	229	0.98	75	0.53	52	0.26
I100	586	1.68	240	1.09	89	0.48	58	0.23
	287	**1.27**	**114**	**0.77**	**42**	**0.38**	**35**	**0.16**

Table 3.17: Comparison of the column generations models with different dimensions (from 1 to 8) ; 1 hour of computing time.

	$t_{COL_{16}}$	gap	$t_{COL_{32}}$	gap	$t_{COL_{64}}$	gap	$t_{COL_{128}}$	gap
I41	11	0.07	10	0.06	17	0.01	32	0.01
I42	15	0.15	17	0.10	26	0.03	47	0.03
I43	15	0.14	17	0.07	56	0.02	298	0.02
I44	23	0.20	26	0.09	35	0.02	72	0.00
I45	22	0.20	29	0.03	37	0.02	119	0.02
I46	20	0.09	33	0.04	49	0.04	67	0.04
I47	21	0.13	41	0.03	37	0.00	66	0.00
I48	23	0.09	37	0.02	38	0.01	126	0.00
I49	29	0.09	37	0.06	78	0.00	143	0.00
I50	30	0.05	59	0.02	59	0.01	142	0.00
I51	24	0.04	42	0.03	67	0.00	1132	0.00
I52	32	0.07	50	0.04	88	0.03	218	0.02
I53	43	0.02	56	0.01	214	0.00	302	0.00
I54	32	0.04	42	0.04	113	0.01	1166	0.00
I55	62	0.02	102	0.01	155	0.01	492	0.00
I56	55	0.09	74	0.07	202	0.03	623	0.01
I57	57	0.05	95	0.03	153	0.02	504	0.00
I58	48	0.06	160	0.02	285	0.01	777	0.01
I59	57	0.08	165	0.05	602	0.01	3275	0.01
I60	65	0.07	133	0.02	358	0.00	1666	0.00
I61	4	0.06	3	0.00	3	0.00	4	0.00
I62	5	0.01	6	0.00	5	0.00	5	0.00
I63	5	0.00	5	0.00	7	0.00	7	0.00
I64	7	0.01	7	0.00	9	0.00	9	0.00
I65	7	0.01	8	0.01	11	0.01	8	0.01
I66	11	0.05	8	0.00	13	0.00	13	0.00
I67	11	0.00	12	0.00	10	0.00	12	0.00
I68	11	0.02	12	0.00	8	0.00	20	0.00
I69	12	0.02	16	0.01	13	0.01	12	0.00
I70	13	0.03	15	0.02	14	0.02	17	0.00
I71	13	0.10	14	0.03	21	0.00	42	0.00
I72	12	0.07	25	0.02	35	0.00	97	0.00
I73	19	0.06	16	0.01	40	0.01	35	0.01
I74	13	0.07	23	0.05	25	0.01	58	0.00
I75	16	0.11	38	0.06	35	0.05	126	0.03
I76	19	0.13	35	0.08	75	0.05	94	0.03
I77	19	0.09	44	0.06	45	0.02	66	0.00
I78	34	0.12	31	0.03	58	0.00	112	0.00
I79	26	0.08	33	0.06	89	0.04	93	0.01
I80	33	0.07	42	0.03	91	0.02	110	0.01
I81	17	0.05	42	0.02	46	0.02	203	0.01
I82	28	0.03	68	0.01	71	0.01	176	0.01
I83	24	0.03	30	0.01	40	0.01	68	0.01
I84	28	0.02	50	0.01	84	0.00	171	0.00
I85	65	0.05	69	0.02	78	0.01	211	0.01
I86	46	0.01	88	0.01	135	0.00	399	0.00
I87	43	0.06	69	0.02	128	0.01	317	0.00
I88	52	0.08	130	0.06	153	0.03	441	0.02
I89	49	0.06	90	0.03	157	0.01	430	0.00
I90	54	0.05	126	0.03	228	0.02	689	0.01
I91	11	0.07	18	0.05	30	0.00	75	0.00
I92	30	0.03	26	0.02	39	0.02	69	0.00
I93	15	0.09	21	0.03	36	0.01	77	0.00
I94	25	0.08	40	0.02	73	0.02	204	0.00
I95	21	0.03	33	0.03	54	0.03	210	0.00
I96	24	0.11	33	0.05	72	0.01	419	0.01
I97	34	0.07	53	0.07	65	0.04	143	0.01
I98	37	0.13	63	0.06	134	0.02	837	0.01
I99	34	0.07	45	0.01	93	0.01	167	0.00
I100	41	0.13	47	0.07	132	0.05	370	0.03
	28	**0.07**	**46**	**0.03**	**85**	**0.01**	**298**	**0.01**

Table 3.18: Comparison of the column generations models with different dimensions (from 64 to 128); 1 hour of computing time.

s

$\mathbf{bCOL_{1-2}}$	$\mathbf{bCOL_{2-4}}$	$\mathbf{bCOL_{4-8}}$	$\mathbf{bCOL_{8-16}}$	$\mathbf{bCOL_{16-32}}$	$\mathbf{bCOL_{32-64}}$	$\mathbf{bCOL_{64-128}}$
262.7	202.6	91.2	80.9	6.0	14.0	0.0
323.9	324.9	152.5	78.1	19.1	23.8	0.0
298.6	240.8	159.5	100.5	27.2	20.2	0.0
327.6	290.5	176.4	67.2	51.3	32.6	9.0
368.7	359.9	288.8	105.3	82.7	5.7	0.0
327.7	343.4	266.6	73.8	24.2	0.0	0.0
325.2	339.8	200.1	80.3	56.2	19.0	0.0
471.4	447.9	252.7	160.5	44.8	11.0	4.0
519.6	491.7	298.9	112.6	17.5	43.6	0.0
593.8	543.3	274.0	144.9	20.0	7.5	9.0
595.1	242.6	55.4	42.3	11.0	18.5	0.0
507.7	408.2	124.4	57.4	21.0	8.0	7.0
674.7	270.6	154.7	60.0	9.9	9.8	0.0
762.8	411.1	172.1	85.7	0.0	23.5	15.5
831.3	377.3	142.2	60.3	12.3	5.5	3.5
868.9	456.2	181.5	48.3	18.8	53.3	21.0
976.6	449.3	210.6	72.8	36.9	10.0	28.9
1217.9	549.8	325.4	75.7	68.5	15.2	3.0
1379.0	645.0	288.2	110.4	44.0	59.0	2.5
1420.0	589.6	279.3	125.0	99.2	34.8	2.3
9.5	7.4	15.5	0.0	13.5	0.0	0.0
9.1	21.7	2.3	2.0	2.5	0.0	0.0
13.4	13.5	34.0	0.0	0.0	0.0	0.0
18.0	9.2	35.4	0.5	1.7	0.0	0.0
24.5	40.1	6.0	7.7	0.0	0.0	3.5
29.1	22.2	9.2	0.0	18.3	0.0	0.0
41.4	42.3	14.3	1.0	0.0	0.0	0.0
16.7	29.8	10.8	3.0	6.5	0.5	0.0
58.3	20.5	23.6	6.0	7.3	0.0	3.0
44.6	20.1	14.1	15.8	7.0	0.0	9.0
175.4	178.8	165.0	13.3	29.2	12.3	0.0
260.3	240.5	170.6	69.3	26.6	7.5	0.0
246.2	232.2	122.9	48.1	25.6	0.0	0.0
313.5	355.9	92.9	59.1	13.7	24.0	4.0
372.8	393.2	187.9	52.8	34.1	8.5	10.3
270.0	330.5	140.1	68.4	35.0	20.5	18.2
389.1	447.5	223.8	52.1	24.8	33.0	16.0
423.7	323.9	274.6	94.2	75.3	24.0	0.0
397.2	438.2	295.6	116.3	19.2	25.5	25.0
520.1	456.6	316.3	88.9	43.7	6.8	16.5
479.7	205.4	155.2	21.0	19.0	4.5	6.7
339.1	277.8	158.9	68.4	20.0	1.0	0.0
463.6	264.8	109.9	52.8	15.1	9.0	0.0
611.8	313.4	119.9	69.5	11.8	13.5	0.5
571.1	311.5	93.7	73.9	28.7	13.0	0.0
844.5	390.1	164.3	39.8	10.5	3.5	3.3
681.3	438.0	177.2	98.8	59.0	15.0	18.3
872.7	414.4	233.5	82.5	32.8	47.5	22.8
844.7	446.0	226.6	142.6	53.8	31.5	14.5
1034.0	570.1	327.3	77.1	48.6	13.0	12.0
281.6	324.7	134.0	43.0	9.5	20.2	1.5
252.2	228.9	125.4	41.0	7.0	1.0	6.0
200.2	254.7	140.2	94.0	33.5	7.5	8.0
486.3	393.2	196.0	74.9	36.8	0.0	9.8
319.1	370.3	164.6	45.8	0.0	0.0	17.5
234.5	330.7	262.9	109.9	46.5	24.8	0.0
440.4	344.5	223.0	100.6	3.8	23.0	22.8
519.0	401.4	231.9	152.1	67.2	31.4	8.3
515.0	427.8	253.6	169.2	57.8	4.0	5.5
613.9	631.8	247.5	102.6	66.7	11.3	26.5
454.9	**315.8**	**166.6**	**68.3**	**28.0**	**14.2**	**6.6**

Table 3.19: Column generation: comparison of times obtained by different block dimensions.

$t_{COL_{1-2}}$	$t_{COL_{2-4}}$	$t_{COL_{4-8}}$	$t_{COL_{8-16}}$	$t_{COL_{16-32}}$	$t_{COL_{32-64}}$	$t_{COL_{64-128}}$
-93.8	-35.1	-11.3	-1.5	-1.6	6.9	14.9
-168.8	-44.5	-5.0	-7.4	1.1	9.8	20.3
-248.1	-58.7	-23.3	-10.1	2.8	38.4	241.9
-246.8	-65.4	-14.9	-3.5	3.6	8.5	37.7
-272.6	-92.2	-17.1	-11.0	7.0	7.4	82.2
-347.5	-102.3	-10.0	-16.9	13.0	16.1	18.3
-261.1	-139.8	-19.4	-19.6	19.5	-3.7	28.7
-399.5	-113.7	-27.4	-24.5	13.7	1.6	87.9
-717.9	-183.0	-10.5	-23.9	8.4	40.0	65.0
-433.0	-244.9	-1.9	-51.0	28.6	0.3	82.9
-103.2	-30.7	0.8	2.3	17.6	25.8	1064.5
-72.0	-32.9	-2.8	10.2	17.8	39.0	129.7
-100.2	-44.4	-1.5	13.1	13.2	157.4	87.9
-104.2	-55.4	-12.8	-0.1	9.2	71.0	1053.7
-161.3	-65.6	-9.3	26.8	39.7	52.6	336.9
-156.4	-76.9	15.4	-0.1	18.9	128.1	421.4
-188.7	-71.9	25.2	-14.6	37.6	57.9	351.4
-318.7	-83.6	-1.6	0.0	112.3	124.3	492.9
-172.9	-118.5	-3.8	8.3	107.7	437.5	2672.4
-228.5	-98.8	4.5	-15.0	67.5	225.0	1308.4
-47.4	-26.1	-3.7	-4.2	-0.8	-0.4	1.4
-36.1	-16.6	-2.6	-6.5	0.4	-0.6	0.2
-34.3	-23.2	-2.9	-5.9	0.1	1.8	-0.6
-67.2	-23.8	-8.7	-5.4	-0.5	2.1	0.2
-74.3	-46.6	-2.5	-12.0	0.7	3.0	-3.1
-138.6	-44.2	-18.6	-8.6	-2.9	4.7	-0.4
-86.8	-32.1	-3.7	-14.5	1.5	-1.8	1.7
-162.8	-51.4	-12.3	-10.0	1.0	-3.8	11.8
-118.5	-60.7	-14.9	-8.5	3.9	-2.4	-1.0
-224.7	-65.5	-15.0	-12.9	2.7	-1.6	3.9
-69.1	-24.3	-3.2	-2.1	0.7	6.6	21.2
-80.5	-51.8	-7.5	-7.4	13.9	9.8	61.4
-170.8	-39.2	-3.1	-6.1	-3.8	24.2	-4.7
-101.2	-40.6	-16.3	-7.3	10.6	1.4	33.4
-131.6	-52.7	-15.1	-11.2	21.6	-2.6	90.7
-211.3	-65.0	-12.1	-14.8	16.2	39.3	19.1
-167.0	-72.6	-4.9	-17.5	24.9	1.7	20.7
-201.3	-99.2	-4.1	-10.9	-3.4	27.9	53.2
-201.0	-99.5	-8.1	-11.9	6.4	56.1	4.5
-252.1	-117.8	-27.0	-6.4	8.8	49.5	18.9
-33.3	-23.5	-0.5	-2.3	25.9	3.1	157.1
-57.1	-28.5	2.6	5.1	39.3	3.6	104.4
-57.1	-24.9	-1.0	-4.9	5.5	10.7	27.5
-66.5	-46.6	-3.4	-2.9	22.1	33.7	86.6
-92.4	-40.8	3.8	28.1	3.8	9.2	133.0
-81.4	-77.6	0.3	9.0	42.6	46.5	264.2
-174.9	-50.4	-16.1	4.8	26.3	59.2	188.2
-137.9	-51.6	-4.2	-0.8	78.1	23.0	288.0
-137.7	-79.2	4.2	-9.2	41.2	67.4	272.4
-175.4	-102.8	2.8	-6.0	71.6	102.1	461.2
-76.1	-39.4	-2.1	-8.6	7.4	12.3	44.7
-91.6	-54.1	-0.9	-0.2	-3.4	12.2	30.5
-106.2	-58.1	-8.5	-9.0	6.3	14.5	41.2
-162.3	-126.2	0.8	-15.2	14.4	33.2	131.5
-213.8	-87.3	-2.8	-20.8	12.3	20.5	156.4
-186.1	-101.3	-10.8	-11.6	8.1	39.1	347.0
-152.5	-108.6	-8.3	-9.4	19.1	11.6	78.0
-340.1	-141.7	15.5	-38.3	26.4	71.2	702.7
-311.2	-154.0	-23.3	-18.1	11.0	47.9	74.1
-346.0	-150.9	-30.4	-17.0	5.7	85.1	238.4
-172.9	**-72.6**	**-6.6**	**-7.5**	**18.4**	**39.4**	**212.6**

Table 3.20: Column generation: comparison of times obtained by different block dimensions.

$\mathbf{impr_{COL_1}}$	$\mathbf{impr_{COL_2}}$	$\mathbf{impr_{COL_4}}$	$\mathbf{impr_{COL_8}}$	$\mathbf{impr_{COL_{16}}}$	$\mathbf{impr_{COL_{32}}}$	$\mathbf{impr_{COL_{64}}}$	$\mathbf{impr_{COL_{128}}}$
12.18%	12.92%	13.48%	13.73%	13.96%	13.98%	14.02%	14.02%
11.37%	12.15%	12.93%	13.30%	13.49%	13.54%	13.59%	13.59%
11.69%	12.33%	12.83%	13.17%	13.38%	13.44%	13.48%	13.48%
11.28%	11.89%	12.44%	12.77%	12.90%	12.99%	13.06%	13.07%
11.49%	12.12%	12.74%	13.23%	13.41%	13.56%	13.57%	13.57%
11.98%	12.49%	13.02%	13.44%	13.55%	13.59%	13.59%	13.59%
12.56%	13.03%	13.52%	13.81%	13.93%	14.01%	14.04%	14.04%
11.41%	12.03%	12.62%	12.96%	13.17%	13.23%	13.25%	13.25%
11.55%	12.20%	12.81%	13.18%	13.32%	13.35%	13.40%	13.40%
11.54%	12.22%	12.84%	13.15%	13.32%	13.34%	13.35%	13.36%
10.37%	11.11%	11.40%	11.47%	11.52%	11.54%	11.56%	11.56%
11.17%	11.72%	12.15%	12.28%	12.35%	12.37%	12.38%	12.38%
10.80%	11.42%	11.66%	11.80%	11.86%	11.86%	11.87%	11.87%
10.50%	11.13%	11.47%	11.61%	11.68%	11.68%	11.70%	11.71%
11.19%	11.80%	12.07%	12.17%	12.22%	12.23%	12.23%	12.23%
11.07%	11.67%	11.98%	12.10%	12.13%	12.15%	12.18%	12.20%
11.21%	11.81%	12.09%	12.22%	12.26%	12.29%	12.29%	12.31%
10.65%	11.35%	11.67%	11.86%	11.91%	11.95%	11.96%	11.96%
10.89%	11.62%	11.97%	12.12%	12.18%	12.20%	12.23%	12.23%
10.20%	10.89%	11.18%	11.31%	11.37%	11.42%	11.44%	11.44%
10.32%	10.36%	10.39%	10.46%	10.46%	10.51%	10.51%	10.51%
10.07%	10.10%	10.18%	10.18%	10.19%	10.20%	10.20%	10.20%
9.45%	9.49%	9.53%	9.64%	9.64%	9.64%	9.64%	9.64%
9.62%	9.67%	9.69%	9.79%	9.79%	9.80%	9.80%	9.80%
9.85%	9.91%	10.01%	10.03%	10.04%	10.04%	10.04%	10.05%
9.98%	10.05%	10.10%	10.12%	10.12%	10.16%	10.16%	10.16%
9.24%	9.33%	9.41%	9.44%	9.44%	9.44%	9.44%	9.44%
9.55%	9.58%	9.63%	9.66%	9.66%	9.67%	9.67%	9.67%
9.18%	9.28%	9.32%	9.36%	9.37%	9.39%	9.39%	9.39%
9.94%	10.01%	10.04%	10.07%	10.09%	10.10%	10.10%	10.12%
10.08%	10.46%	10.85%	11.20%	11.23%	11.30%	11.32%	11.32%
10.28%	10.75%	11.20%	11.51%	11.64%	11.69%	11.70%	11.70%
11.00%	11.41%	11.80%	12.01%	12.09%	12.13%	12.13%	12.13%
10.41%	10.87%	11.40%	11.54%	11.62%	11.64%	11.68%	11.69%
10.68%	11.17%	11.70%	11.95%	12.02%	12.06%	12.07%	12.09%
10.32%	10.64%	11.03%	11.20%	11.28%	11.32%	11.34%	11.36%
10.03%	10.46%	10.94%	11.18%	11.24%	11.27%	11.30%	11.32%
10.58%	11.02%	11.35%	11.64%	11.73%	11.81%	11.84%	11.84%
10.36%	10.73%	11.14%	11.42%	11.52%	11.54%	11.57%	11.59%
10.24%	10.70%	11.11%	11.39%	11.47%	11.51%	11.51%	11.53%
10.58%	11.17%	11.42%	11.62%	11.64%	11.67%	11.67%	11.68%
10.30%	10.67%	10.96%	11.13%	11.21%	11.23%	11.23%	11.23%
9.99%	10.42%	10.67%	10.77%	10.82%	10.83%	10.84%	10.84%
9.36%	9.86%	10.12%	10.21%	10.27%	10.28%	10.29%	10.29%
10.73%	11.15%	11.38%	11.45%	11.51%	11.53%	11.54%	11.54%
9.98%	10.56%	10.84%	10.95%	10.98%	10.98%	10.99%	10.99%
9.75%	10.18%	10.45%	10.56%	10.62%	10.66%	10.67%	10.68%
9.95%	10.45%	10.69%	10.82%	10.87%	10.89%	10.91%	10.93%
9.82%	10.27%	10.50%	10.62%	10.70%	10.73%	10.75%	10.75%
10.33%	10.85%	11.14%	11.30%	11.34%	11.36%	11.37%	11.38%
10.27%	10.85%	11.52%	11.80%	11.89%	11.90%	11.95%	11.95%
9.89%	10.37%	10.81%	11.04%	11.12%	11.14%	11.14%	11.15%
10.34%	10.66%	11.08%	11.31%	11.46%	11.51%	11.53%	11.54%
10.38%	11.08%	11.65%	11.94%	12.04%	12.10%	12.10%	12.11%
9.76%	10.18%	10.66%	10.88%	10.93%	10.93%	10.93%	10.96%
10.16%	10.45%	10.85%	11.17%	11.31%	11.37%	11.40%	11.40%
10.54%	11.02%	11.40%	11.65%	11.76%	11.76%	11.79%	11.81%
10.56%	11.07%	11.47%	11.70%	11.86%	11.92%	11.95%	11.96%
10.93%	11.42%	11.82%	12.06%	12.22%	12.28%	12.28%	12.29%
10.12%	10.66%	11.22%	11.43%	11.52%	11.58%	11.59%	11.61%
10.50%	**10.95%**	**11.31%**	**11.50%**	**11.58%**	**11.61%**	**11.63%**	**11.63%**

Table 3.21: Column generation: improvement on the $TRAD - TKP$ continuous relaxation.

$\mathbf{impr_{COL_1}}$	$\mathbf{impr_{COL_2}}$	$\mathbf{impr_{COL_4}}$	$\mathbf{impr_{COL_8}}$	$\mathbf{impr_{COL_{16}}}$	$\mathbf{impr_{COL_{32}}}$	$\mathbf{impr_{COL_{64}}}$	$\mathbf{impr_{COL_{128}}}$
0.080%	0.218%	0.227%	1.059%	1.221%	1.422%	0.839%	0.444%
0.047%	0.168%	0.179%	0.582%	0.873%	0.820%	0.517%	0.292%
0.033%	0.116%	0.120%	0.534%	0.920%	0.776%	0.242%	0.045%
0.032%	0.112%	0.117%	0.491%	0.573%	0.497%	0.377%	0.181%
0.028%	0.085%	0.089%	0.397%	0.601%	0.462%	0.369%	0.114%
0.024%	0.084%	0.087%	0.365%	0.681%	0.413%	0.277%	0.202%
0.027%	0.065%	0.068%	0.338%	0.656%	0.344%	0.380%	0.214%
0.019%	0.064%	0.067%	0.272%	0.571%	0.360%	0.345%	0.105%
0.012%	0.050%	0.052%	0.249%	0.459%	0.356%	0.173%	0.094%
0.015%	0.037%	0.039%	0.162%	0.440%	0.226%	0.225%	0.094%
0.067%	0.215%	0.221%	0.526%	0.479%	0.277%	0.171%	0.010%
0.087%	0.205%	0.213%	0.571%	0.390%	0.250%	0.140%	0.057%
0.061%	0.150%	0.154%	0.394%	0.275%	0.211%	0.056%	0.039%
0.051%	0.110%	0.114%	0.356%	0.360%	0.280%	0.104%	0.010%
0.041%	0.107%	0.109%	0.341%	0.196%	0.120%	0.079%	0.025%
0.041%	0.100%	0.103%	0.221%	0.222%	0.165%	0.060%	0.020%
0.036%	0.100%	0.102%	0.170%	0.215%	0.130%	0.081%	0.024%
0.024%	0.085%	0.088%	0.248%	0.249%	0.075%	0.042%	0.015%
0.032%	0.068%	0.070%	0.248%	0.213%	0.074%	0.020%	0.004%
0.025%	0.062%	0.064%	0.141%	0.175%	0.086%	0.032%	0.007%
0.121%	0.273%	0.274%	1.275%	2.588%	3.234%	3.727%	2.473%
0.150%	0.325%	0.327%	0.861%	1.898%	1.768%	1.992%	1.906%
0.132%	0.255%	0.256%	0.857%	1.798%	1.765%	1.335%	1.467%
0.086%	0.214%	0.214%	0.764%	1.325%	1.414%	1.091%	1.064%
0.069%	0.145%	0.147%	0.523%	1.391%	1.276%	0.927%	1.300%
0.045%	0.122%	0.122%	0.514%	0.911%	1.232%	0.786%	0.810%
0.063%	0.153%	0.155%	0.378%	0.895%	0.783%	0.918%	0.789%
0.039%	0.114%	0.114%	0.471%	0.920%	0.840%	1.245%	0.496%
0.043%	0.097%	0.097%	0.461%	0.790%	0.595%	0.701%	0.757%
0.030%	0.095%	0.095%	0.395%	0.805%	0.665%	0.745%	0.579%
0.090%	0.244%	0.253%	0.729%	0.848%	0.810%	0.551%	0.272%
0.065%	0.138%	0.143%	0.609%	1.010%	0.460%	0.332%	0.121%
0.046%	0.168%	0.174%	0.472%	0.624%	0.777%	0.305%	0.346%
0.059%	0.142%	0.148%	0.577%	0.915%	0.499%	0.473%	0.201%
0.047%	0.118%	0.123%	0.437%	0.745%	0.320%	0.344%	0.096%
0.032%	0.096%	0.099%	0.330%	0.587%	0.320%	0.152%	0.121%
0.036%	0.092%	0.096%	0.308%	0.596%	0.258%	0.249%	0.171%
0.030%	0.074%	0.077%	0.260%	0.346%	0.387%	0.203%	0.106%
0.030%	0.074%	0.077%	0.300%	0.442%	0.355%	0.130%	0.124%
0.023%	0.058%	0.060%	0.290%	0.349%	0.277%	0.126%	0.105%
0.139%	0.261%	0.267%	0.617%	0.703%	0.275%	0.256%	0.058%
0.097%	0.217%	0.223%	0.480%	0.397%	0.166%	0.158%	0.064%
0.089%	0.190%	0.194%	0.371%	0.449%	0.366%	0.269%	0.160%
0.063%	0.122%	0.125%	0.328%	0.364%	0.204%	0.122%	0.060%
0.064%	0.150%	0.153%	0.308%	0.176%	0.167%	0.147%	0.055%
0.051%	0.093%	0.095%	0.298%	0.240%	0.124%	0.081%	0.028%
0.035%	0.097%	0.100%	0.277%	0.247%	0.154%	0.083%	0.034%
0.040%	0.096%	0.099%	0.206%	0.210%	0.084%	0.071%	0.025%
0.036%	0.077%	0.079%	0.184%	0.220%	0.119%	0.068%	0.025%
0.031%	0.068%	0.069%	0.188%	0.209%	0.090%	0.050%	0.017%
0.075%	0.178%	0.189%	0.610%	1.106%	0.657%	0.393%	0.159%
0.056%	0.122%	0.127%	0.368%	0.374%	0.423%	0.289%	0.162%
0.053%	0.118%	0.122%	0.473%	0.769%	0.544%	0.324%	0.150%
0.032%	0.067%	0.070%	0.294%	0.475%	0.304%	0.166%	0.059%
0.028%	0.077%	0.081%	0.260%	0.520%	0.328%	0.203%	0.052%
0.030%	0.071%	0.073%	0.310%	0.463%	0.349%	0.159%	0.027%
0.034%	0.069%	0.071%	0.269%	0.347%	0.222%	0.182%	0.083%
0.020%	0.055%	0.057%	0.156%	0.324%	0.189%	0.089%	0.014%
0.020%	0.050%	0.052%	0.233%	0.362%	0.275%	0.133%	0.074%
0.017%	0.045%	0.047%	0.196%	0.280%	0.247%	0.088%	0.031%
0.050%	**0.124%**	**0.127%**	**0.415%**	**0.630%**	**0.511%**	**0.403%**	**0.277%**

Table 3.22: Column generation: improvement on original $TRAD{-}TKP$ continuous relaxation per second.

$\mathbf{impr_{COL_1}}$	$\mathbf{impr_{COL_2}}$	$\mathbf{impr_{COL_4}}$	$\mathbf{impr_{COL_8}}$	$\mathbf{impr_{COL_{16}}}$	$\mathbf{impr_{COL_{32}}}$	$\mathbf{impr_{COL_{64}}}$	$\mathbf{impr_{COL_{128}}}$
0.00%	0.00%	0.45%	0.74%	1.00%	1.02%	1.06%	1.06%
0.00%	0.00%	0.31%	0.73%	0.95%	1.00%	1.07%	1.07%
0.00%	0.00%	0.39%	0.77%	1.01%	1.08%	1.13%	1.13%
0.00%	0.00%	0.21%	0.59%	0.74%	0.85%	0.92%	0.94%
0.00%	0.00%	0.32%	0.89%	1.09%	1.26%	1.27%	1.27%
0.00%	0.00%	0.34%	0.81%	0.94%	0.99%	0.99%	0.99%
0.00%	0.00%	0.32%	0.66%	0.79%	0.88%	0.92%	0.92%
0.00%	0.00%	0.39%	0.77%	1.02%	1.09%	1.10%	1.11%
0.00%	0.00%	0.60%	1.02%	1.18%	1.21%	1.27%	1.27%
0.00%	0.00%	0.66%	1.01%	1.20%	1.23%	1.24%	1.25%
0.00%	0.17%	0.50%	0.58%	0.64%	0.65%	0.68%	0.68%
0.00%	0.00%	0.25%	0.40%	0.47%	0.50%	0.51%	0.52%
0.00%	0.00%	0.21%	0.37%	0.43%	0.44%	0.45%	0.45%
0.00%	0.02%	0.40%	0.56%	0.64%	0.64%	0.66%	0.68%
0.00%	0.13%	0.45%	0.56%	0.61%	0.62%	0.63%	0.63%
0.00%	0.00%	0.32%	0.46%	0.49%	0.51%	0.55%	0.57%
0.00%	0.03%	0.34%	0.49%	0.54%	0.57%	0.57%	0.59%
0.00%	0.13%	0.49%	0.70%	0.75%	0.80%	0.81%	0.81%
0.00%	0.00%	0.38%	0.55%	0.62%	0.64%	0.68%	0.68%
0.00%	0.00%	0.31%	0.46%	0.53%	0.59%	0.60%	0.61%
0.00%	0.00%	0.00%	0.03%	0.03%	0.09%	0.09%	0.09%
0.00%	0.03%	0.11%	0.12%	0.13%	0.14%	0.14%	0.14%
0.00%	0.00%	0.00%	0.12%	0.12%	0.12%	0.12%	0.12%
0.00%	0.01%	0.04%	0.14%	0.14%	0.15%	0.15%	0.15%
0.00%	0.00%	0.05%	0.06%	0.08%	0.08%	0.08%	0.09%
0.00%	0.00%	0.00%	0.00%	0.00%	0.05%	0.05%	0.05%
0.00%	0.00%	0.06%	0.09%	0.09%	0.09%	0.09%	0.09%
0.00%	0.00%	0.01%	0.03%	0.04%	0.05%	0.05%	0.05%
0.00%	0.02%	0.07%	0.11%	0.12%	0.14%	0.14%	0.14%
0.00%	0.00%	0.04%	0.07%	0.09%	0.11%	0.11%	0.12%
0.00%	0.00%	0.21%	0.61%	0.65%	0.72%	0.75%	0.75%
0.00%	0.00%	0.31%	0.67%	0.81%	0.87%	0.88%	0.88%
0.00%	0.00%	0.35%	0.59%	0.68%	0.73%	0.73%	0.73%
0.00%	0.00%	0.21%	0.37%	0.47%	0.49%	0.53%	0.53%
0.00%	0.00%	0.32%	0.60%	0.68%	0.73%	0.74%	0.76%
0.00%	0.00%	0.09%	0.27%	0.36%	0.41%	0.44%	0.46%
0.00%	0.00%	0.25%	0.53%	0.59%	0.62%	0.66%	0.68%
0.00%	0.00%	0.15%	0.47%	0.58%	0.66%	0.69%	0.69%
0.00%	0.00%	0.07%	0.38%	0.50%	0.52%	0.55%	0.57%
0.00%	0.00%	0.28%	0.59%	0.68%	0.73%	0.73%	0.75%
0.00%	0.00%	0.25%	0.46%	0.49%	0.52%	0.53%	0.54%
0.00%	0.00%	0.20%	0.39%	0.47%	0.50%	0.50%	0.50%
0.00%	0.00%	0.18%	0.30%	0.35%	0.37%	0.37%	0.37%
0.00%	0.00%	0.24%	0.34%	0.41%	0.42%	0.43%	0.43%
0.00%	0.00%	0.22%	0.30%	0.36%	0.39%	0.40%	0.40%
0.00%	0.06%	0.37%	0.49%	0.52%	0.53%	0.53%	0.54%
0.00%	0.07%	0.38%	0.50%	0.57%	0.61%	0.63%	0.64%
0.00%	0.00%	0.24%	0.39%	0.45%	0.47%	0.50%	0.51%
0.00%	0.00%	0.16%	0.29%	0.38%	0.41%	0.43%	0.44%
0.00%	0.06%	0.38%	0.56%	0.60%	0.63%	0.64%	0.64%
0.00%	0.00%	0.31%	0.62%	0.72%	0.74%	0.79%	0.79%
0.00%	0.09%	0.57%	0.84%	0.93%	0.94%	0.94%	0.96%
0.00%	0.00%	0.27%	0.53%	0.70%	0.76%	0.77%	0.79%
0.00%	0.00%	0.63%	0.95%	1.07%	1.13%	1.13%	1.15%
0.00%	0.00%	0.42%	0.66%	0.72%	0.72%	0.72%	0.75%
0.00%	0.00%	0.29%	0.65%	0.80%	0.87%	0.90%	0.90%
0.00%	0.00%	0.27%	0.55%	0.67%	0.68%	0.71%	0.73%
0.00%	0.00%	0.42%	0.68%	0.85%	0.92%	0.96%	0.97%
0.00%	0.00%	0.32%	0.59%	0.78%	0.84%	0.84%	0.85%
0.00%	0.00%	0.47%	0.72%	0.82%	0.88%	0.90%	0.92%
0.00%	**0.01%**	**0.28%**	**0.50%**	**0.59%**	**0.62%**	**0.64%**	**0.65%**

Table 3.23: Column generation: Improvement on $TRAD - TKP$ root node.

Table 3.24: Comparison of *TRAD-TKP*, and *REF-TKP* models; 1 hour of computing time.

	t_{TRAD}	*gap*	$t_{T_{COL_{32}}}$	*gap*	$t_{T_{COL_{64}}}$	*gap*	$t_{T_{COL_{128}}}$	*gap*
I41	682.91	0.000%	44.07	0.000%	19.17	0.000%	61.13	0.000%
I42	731.42	0.000%	264.36	0.000%	165.45	0.000%	246.65	0.000%
I43	TL	0.146%	174.23	0.000%	250.89	0.000%	1823.63	0.000%
I44	TL	0.127%	387.83	0.000%	61.48	0.000%	95.99	0.000%
I45	TL	0.367%	917.26	0.000%	216.19	0.000%	407.14	0.000%
I46	TL	0.039%	145.21	0.000%	363.2	0.000%	1263.9	0.000%
I47	TL	0.012%	109.28	0.000%	36.96	0.000%	65.65	0.000%
I48	TL	0.431%	184.06	0.000%	61.67	0.000%	126.29	0.000%
I49	TL	0.476%	2498.95	0.000%	86.83	0.000%	142.55	0.000%
I50	TL	0.652%	185.32	0.000%	141.72	0.000%	193.85	0.000%
I51	1083.64	0.000%	159.25	0.000%	111.93	0.000%	1555.17	0.000%
I52	3201.23	0.000%	942.02	0.000%	1535.07	0.000%	1171.77	0.000%
I53	3516.77	0.000%	277.24	0.000%	284.75	0.000%	1010.58	0.000%
I54	TL	0.176%	932.89	0.000%	311.35	0.000%	1166.36	0.000%
I55	TL	0.172%	420.11	0.000%	628.44	0.000%	685.45	0.000%
I56	TL	0.120%	TL	0.061%	2028.3	0.000%	1690.62	0.000%
I57	TL	0.173%	800.07	0.000%	371.97	0.000%	504.05	0.000%
I58	TL	0.436%	917.84	0.000%	525.66	0.000%	1574.51	0.000%
I59	TL	0.434%	TL	0.029%	3459.45	0.000%	TL	0.001%
I60	TL	0.335%	2208.74	0.000%	746.49	0.000%	2275.31	0.000%
I61	1.39	0.000%	3.25	0.000%	2.82	0.000%	4.25	0.000%
I62	1.10	0.000%	5.77	0.000%	5.12	0.000%	5.35	0.000%
I63	1.37	0.000%	8.06	0.000%	10	0.000%	9.61	0.000%
I64	4.79	0.000%	6.93	0.000%	8.98	0.000%	9.21	0.000%
I65	2.09	0.000%	25.46	0.000%	42.83	0.000%	9.13	0.000%
I66	2.77	0.000%	8.25	0.000%	12.92	0.000%	12.55	0.000%
I67	7.15	0.000%	12.06	0.000%	10.29	0.000%	11.97	0.000%
I68	2.79	0.000%	21.81	0.000%	11.01	0.000%	22.53	0.000%
I69	8.27	0.000%	25.48	0.000%	24.44	0.000%	12.4	0.000%
I70	12.38	0.000%	16.94	0.000%	15.41	0.000%	17.49	0.000%
I71	96.58	0.000%	61.42	0.000%	20.54	0.000%	41.7	0.000%
I72	1820.70	0.000%	71.31	0.000%	35.24	0.000%	96.62	0.000%
I73	403.50	0.000%	21.25	0.000%	54.55	0.000%	53.42	0.000%
I74	784.95	0.000%	260.15	0.000%	34.04	0.000%	58.08	0.000%
I75	TL	0.055%	304.33	0.000%	286.7	0.000%	410.7	0.000%
I76	1660.95	0.000%	1296.66	0.000%	2020.64	0.000%	124.4	0.000%
I77	TL	0.079%	1328.11	0.000%	184.4	0.000%	156.21	0.000%
I78	TL	0.112%	69.8	0.000%	58.37	0.000%	111.59	0.000%
I79	392.40	0.000%	TL	0.016%	TL	0.000%	207.5	0.000%
I80	TL	0.234%	1990.92	0.000%	702.74	0.000%	194.15	0.000%
I81	1740.23	0.000%	110.43	0.000%	96.6	0.000%	231.85	0.000%
I82	684.61	0.000%	169.25	0.000%	91.28	0.000%	245.51	0.000%
I83	72.11	0.000%	61.35	0.000%	52.39	0.000%	98.01	0.000%
I84	259.00	0.000%	135.7	0.000%	152.15	0.000%	170.64	0.000%
I85	1841.34	0.000%	407.83	0.000%	166.81	0.000%	415.69	0.000%
I86	TL	0.106%	270.03	0.000%	311.85	0.000%	715.33	0.000%
I87	TL	0.125%	1471.97	0.000%	960.29	0.000%	316.71	0.000%
I88	TL	0.101%	TL	0.005%	TL	0.000%	1382.36	0.000%
I89	TL	0.085%	2865.74	0.000%	354.72	0.000%	429.63	0.000%
I90	TL	0.142%	TL	0.006%	TL	0.022%	TL	0.000%
I91	277.08	0.000%	278.67	0.000%	50.77	0.000%	75.06	0.000%
I92	1854.41	0.000%	244.28	0.000%	740.21	0.000%	323.86	0.000%
I93	3165.36	0.000%	167.8	0.000%	170.21	0.000%	76.85	0.000%
I94	TL	0.284%	168.78	0.000%	411.54	0.000%	490.8	0.000%
I95	TL	0.070%	123.84	0.000%	231.89	0.000%	210.2	0.000%
I96	TL	0.233%	2947.23	0.000%	1817.19	0.000%	TL	0.102%
I97	2600.22	0.000%	1256	0.000%	790.09	0.000%	1860.35	0.000%
I98	TL	0.356%	TL	0.000%	399.37	0.000%	1272.27	0.000%
I99	TL	0.351%	154.67	0.000%	283.95	0.000%	166.66	0.000%
I100	TL	0.357%	TL	0.027%	2661.28	0.000%	TL	0.002%
	897.12	0.113%	527.17	0.002%	433.17	0.000%	466.27	0.002%

Table 3.25: Comparison of the Dynamic Programming algorithm, $TRAD-TKP$ and $REF-TKP$ models; 1 hour of computing time allowed.

	z	t_T/gap_T	t_{DP}	$t_{R_{32}}$ / $gap_{R_{32}}$	$t_{R_{64}}$ / $gap_{R_{64}}$	$t_{R_{128}}$
I1-2697-2688-10-10	62524	4.3	0.7	54.2	31.6	8.9
I2-2825-2816-10-10	65046	5.0	0.8	82.3	35.4	32.8
I3-2953-2944-10-10	67558	4.6	0.8	14.1	9.4	9.7
I4-3081-3072-10-10	70316	12.0	0.9	182.5	65.5	59.5
I5-3209-3200-10-10	76634	5.6	0.9	26.9	21.3	15.6
I6-3337-3328-10-10	77204	11.7	1.0	103.2	70.4	12.1
I7-3465-3456-10-10	81690	8.0	1.2	321.1	15.5	8.8
I8-3593-3584-10-10	84581	6.5	1.2	45.3	36.2	25.9
I9-3721-3712-10-10	87297	7.8	1.2	126.2	49.2	22.9
I10-3849-3840-10-10	88889	12.5	1.4	104.7	46.0	21.1
I11-4480-2688-9-15	88574	TL (0.05%)	7.5	255.9	144.5	236.2
I12-4749-2816-9-15	96366	3257.9	8.0	267.9	108.0	50.9
I13-4887-2944-9-15	97987	171.6	8.1	386.6	271.3	273.1
I14-5153-3072-9-15	103747	442.3	9.0	317.1	52.6	72.7
I15-5298-3200-9-15	103498	419.8	8.8	988.4	93.8	71.2
I16-5547-3328-9-15	108686	TL (0.01%)	9.8	198.0	165.8	164.2
I17-5745-3456-9-15	112017	57.7	9.7	89.9	25.3	45.0
I18-5929-3584-9-15	116631	1659.0	11.3	267.8	69.2	35.0
I19-6208-3712-9-15	125346	TL (0.03%)	11.0	340.3	58.5	39.9
I20-6462-3840-9-15	128454	TL (0.11%)	12.1	TL (0.07%)	789.3	378.8
I21-4972-2688-11-20	87259	TL (0.14%)	148.1	TL (0.13%)	127.1	48.2
I22-5161-2816-11-20	89548	TL (0.24%)	154.8	1092.6	229.6	47.3
I23-5423-2944-11-20	96418	TL (0.19%)	163.1	601.2	171.1	76.4
I24-5658-3072-11-20	98019	TL (0.57%)	170.9	200.4	124.2	117.7
I25-5875-3200-11-20	104227	TL (0.41%)	181.0	1656.4	591.1	802.2
I26-6112-3328-11-20	107704	TL (0.61%)	186.3	1271.0	797.3	219.3
I27-6380-3456-11-20	109805	TL (0.23%)	192.8	TL (0.06%)	113.2	76.6
I28-6582-3584-11-20	116248	TL (0.53%)	200.2	786.8	369.7	459.3
I29-6817-3712-11-20	119729	TL (0.56%)	208.8	TL (0.06%)	1159.7	295.7
I30-7026-3840-11-20	123463	TL (0.51%)	215.6	TL (0.07%)	1483.6	961.6
I31-6322-2688-11-25	102424	TL (0.72%)	TL	TL (0.09%)	TL (0.07%)	181.6
I32-6546-2816-11-25	103159	TL (1.14%)	TL	TL (0.06%)	254.7	226.2
I33-6895-2944-11-25	111884	TL (0.92%)	TL	2226.2	1079.8	290.7
I34-7209-3072-11-25	117903	TL (0.88%)	TL	TL (0.05%)	1904.5	287.7
I35-7484-3200-11-25	120668	TL (0.86%)	TL	1585.6	728.4	305.4
I36-7784-3328-11-25	123739	TL (0.91%)	TL	3554.6	588.6	257.5
I37-8089-3456-11-25	130308	TL (0.87%)	TL	TL (0.09%)	3387.6	774.5
I38-8425-3584-11-25	133092	TL (1.11%)	TL	TL (0.05%)	651.3	462.1
I39-8650-3712-11-25	138613	TL (0.93%)	TL	2796.9	791.3	200.6
I40-8977-3840-11-25	144612	TL (0.90%)	TL	TL (0.11%)	587.8	458.7

Chapter 4

Application: Multi-load AGV dispatching in automated seaport container terminals

4.1 Introduction

During recent years, maritime cargo transportation has been established as a predominant means of transportation in international trade. Coupled with the dramatic increase in world trade volumes, terminal operators have been under pressure to enhance container handling performance. Between 1995 and 2008, world container traffic more than tripled in volume, from 137 million twenty feet equivalent units (TEUs) to 387 million TEUs, growing at an average annual rate of about 8% (figure 4.1), despite the 2008 decline in the world's economic activity. As a result, larger vessels have been introduced including the new super post-Panamax 12000+ TEUs vessel, and competition between seaports has increased. The competition is often measured against different performance factors such as the time the ship is in port (transhipment time) as well as low rates for loading and discharging of ships [68]. Therefore, the main objectives for successful container terminals are the rapid turnover of containers and the short berthing of the vessels.

Automation has been recognised as one of the means for improving the overall productivity of the handling and transportation equipment; the demand for designing container terminals with part or full automation has thus increased significantly. The originators of the Automated Container Terminal concept include the developments at Thamesport and the European Container Terminals, Rotterdam. Automated or partially automated operations have since then been introduced at the Container Terminal Altenwerder (CTA) in Hamburg, at Patrick Brisbane, Australia and at the AP Moller Terminal in Virginia, USA [66]. Cranes, previously operated manually, have become automated, while manually driven carts have been replaced by Automated Guided Vehicles (AGVs). For example, at the Euromax Terminal in Rotterdam, (ECT), container vessels are handled using the largest semi-automated quay cranes in the world as well as unmanned (twice as fast than previous generations) AGVs which move the containers between ship and stack. In particular, the ECT uses more than 130 automated stacking cranes and 260 AGVs [67] whereas CTA, operates a fleet of around 70 AGVs. Furthermore, these AGVs are automatically re-fuelled and are capable of carrying more than one container at a time subject to capacity [71]. The multi-load mode of these

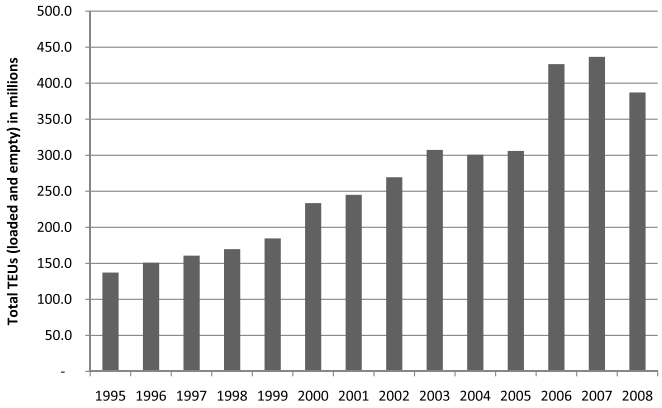


Figure 4.1: World container traffic (total TEUs loaded and empty)

vehicles provides a recent development and innovation in transportation technology.

Container terminals are complex and dynamic systems. Considering the requirement for synchronization of all the interrelated component parts, the design of accurate control software has so far received sufficient attention from the academic community. Comprehensive literature reviews can be found in [68], [72] and [73]. Most of the research efforts on the AGV dispatching problem have targetted specific environments, especially flexible manufacturing systems (FMS), see for example [60] and [74]. Routing approaches include the development of a hybrid collision-free decomposition method presented in [55] for FMS and a distributed collision-free routing method under motion delay disturbance in transportation environments by [62]. A dynamic routing strategy for a generic network of automated material handling systems was also proposed by [75]. Finally, [61] developed an algorithm for automated container terminals producing collision- and deadlock-free routes at the time of route calculation, based on a shortest path algorithm with time-windows .

The dispatching problem in seaport container terminals has not received much attention in the literature apart from some notable contributions where issues related to single-load carriers are considered [69, 70, 76, 77, 78]. For the dual-load AGV dispatching problem, thereafter referred to as AGVDP, [57] propose a flexible priority rule-based approach which is compared to an alternative MIP model. Their computational results, based on two terminal layout configurations and two workload scenarios show an improvement of the overall terminal performance when multi-load carriers are used. This work is extended in [56] where a pattern-based off-line heuristic approach with dual-load AGVs is developed and evaluated using a scalable simulation model which considers stochastic handling times, various terminal configurations and operation modes for AGVs. The results of the detailed simulation study show that the developed heuristic clearly outperforms conventional online heuristics adopted from flexible manufacturing systems. Furthermore, the numerical results suggest that the performance of dispatching strategies is insensitive to the configuration of the terminal.

The complex and highly dynamic environment found in ports, coupled with the requirement for perfect synchronisation of several inter-related components of a container terminal, make

the AGVDP in automated terminals a very hard combinatorial optimisation problem. To this end, the focus on developing effective and efficient real-time approaches for this problem is more than understandable. In this chaper, we focus on providing an optimisation methodology for solving the AGVDP for multi-load carriers that can be used as a benchmark for the evaluation of various real-time heuristics. In particular, we propose a new MIP model for the AGVDP and a column generation approach for a variant of this problem. Furthermore, we present a computational study which verifies that the multi-load AGV dispatching problem can be solved in reasonable computational times while providing good quality solutions.

The remainder of the chaper is organized as follows. Section 4.2 provides a detailed description of the problem under investigation as well as the notation used in the chaper. Section 4.3 and 4.4 describe our proposed new formulations and solution methodologies and Section 4.5 provides an illustrative example. Computational results are given in Section 4.6 and the chaper is concluded in Section 4.7.

4.2 Problem description and notation

Generally, the seaport container terminal system is divided into three parts. Firstly, the berthing area where the quay cranes are located and are used for the loading/unloading of the containers onto/from the vessels. Secondly, the storage yard which includes stacking cranes for storing containers in various blocks. Finally, the AGV area, where the transporting of the containers between the quay cranes and the storage area is realized by AGVs. The general AGV control problem typically consists of three sub-problems: (i) assigning a job to a vehicle (transportation order), (ii) routing the AGVs and (iii) traffic control. The focus of this chaper is on the assignment problem.

The problem considered in this section assumes that the AGVs can have different capacities and that they are operated under a multi-carrier mode. Each job involves the transportation of a container from a pick-up location to a delivery/drop-off location, hence, a job consists of two operations. We denote the process of picking-up/delivering the container as the pick-up/delivery operation of a job.

We are given a set of AGVs, M, a set of cranes, C, and a set of jobs J to be performed within a finite time horizon T_{max}. For each pick-up operation i^+ associated with a specific job $j \in J$ there is a corresponding delivery operation i^-; the set of all such pairs (i^+, i^-) is denoted by $(I^+ : I^-)$. We define set I to include all initial operations $i_m^L \in I^L$, (currently being undertaken by the vehicles), all pick-up and delivery operations I^+ and I^-, respectively, and a dummy end operation i^E. Each operation i is released at time t_i^T (also referred to as target time) and is associated with a specific service location $c \in C$. The AGV load capacity requirement of operation i is denoted by q_i.

Each vehicle m has load capacity Q_m ($Q_m = 2$ for dual-load carriers) and load y_m^0 at time t_m^0; the latter denotes the time that operation i_m^L is completed by AGV m. Fixed travel times, denoted by $d_{i_1 i_2}$, are assumed between pairs of locations where operations i_1, i_2 are performed. Let O_c be the set of all operations assigned to crane $c \in C$. A summary of the above notation is given in Table 4.1.

Given the input data stated above, the problem is to determine the optimal assignment of all operations in set I to vehicles in M so that the total cost associated to the lateness of the operations is minimized, subject to constraints: (i) each operation is assigned to only one

$m \in M$	: AGV vehicles
$c \in C$	: set of service locations (quay and stacking cranes)
$j \in J$	: set of jobs
$i_m^L \in I^L$	: current operation carried out by vehicle m
$i^+ \in I^+$	: pick-up operations
$i^- \in I^-$	: delivery operations
i^E	: dummy end operation
$I = I^L \cup I^+ \cup I^- \cup \{i^E\}$	: the set of all operations
$(i^+, i^-) \in (I^+ : I^-)$	: the set of all pairs of pick-up and delivery operations
t_i^T	: target/release time associated for operation i
t_i^H	: handling time for operation i
t_m^0	: time operation i_m^L is completed
$t \in [0, T_{max}]$	: a specific time within the problem's horizon
q_i	: load capacity requirement of operation i
Q_m	: capacity of vehicle m
y_m^0	: load of vehicle m at time t_m^0
$d_{i_1 i_2}$	: travel time between the locations of operations i_1 and i_2
$\mathcal{O}_c$	: set of operations performed at crane $c \in C$

Table 4.1: AGVDP Notation

AGV, (ii) the pick-up operation for a job precedes its delivery operation and (iii) the load capacity availability of each AGV is respected.

4.3 A new compact MIP formulation of the AGVDP

In this section, we formulate the AGVDP for multi-load carriers over a finite planning horizon as a mixed-integer progrmaming (MIP) model. Using the notation presented in the previous section, the following decision variables are defined:

- Binary variables:

$$x_{i_1,i_2} = \begin{cases} 1, \text{ if op. } i_1 \in I \setminus \{i^E\} \text{ is an immediate predecessor of op. } i_2 \in I \setminus I^L \\ 0, \text{ otherwise} \end{cases}$$

$$z_{mi} = \begin{cases} 1, \text{ if op. } i \in I \text{ is assigned to vehicle } m \in M \\ 0, \text{ otherwise} \end{cases}$$

$$o_{it} = \begin{cases} 1, \text{ if op. } i \in I \text{ has been initiated at time } t \in [0, T_{max}] \\ 0, \text{ otherwise} \end{cases}$$

- Continuous variables with integrality property:

  y_i : load of the AGV after performing operation i

  $t_i^{arr} \backslash t_i^{dep}$: arrival\departure time at\from a service location before\after performing operation $i \in I \backslash \{i^E\}$

  – l_i: lateness in performing operation $i \in I \backslash \{i^E\}$.

The complete mathematical formulation of the problem is given in Model (4.1)-(4.23) .

A new MIP formulation for the multi-load AGVDP in container

Minimize:

$$\alpha \sum_{i \in I} l_i \tag{4.1}$$

subject to:

$$\sum_{i_1 \in I} x_{i_1,i_2} = 1 \qquad \forall i_2 \in I \backslash \{I^L, i^E\} \tag{4.2}$$

$$\sum_{i_2 \in I} x_{i_1,i_2} = 1 \qquad \forall i_1 \in I \backslash \{i^E\} \tag{4.3}$$

$$\sum_{m \in M} z_{mi} = 1 \qquad \forall i \in I \tag{4.4}$$

$$z_{mi^+} - z_{mi^-} = 0 \qquad \forall m \in M, (i^+, i^-) \in (I^+ : I^-) \tag{4.5}$$

$$z_{mi} - x_{i^L_m,i} \geqslant 0 \qquad \forall m \in M, i \in I \tag{4.6}$$

$$z_{mi_2} - z_{mi_1} - x_{i_1,i_2} \geqslant -1 \qquad \forall m \in M, i_1 \in I \backslash \{i^E\}, i_2 \in I \backslash I^L \tag{4.7}$$

$$y_{i^L_m} = y^0_m \qquad \forall m \in M \tag{4.8}$$

$$y_{i_2} - y_{i_1} + (1 - x_{i_1,i_2}) N_1 \geqslant q_{i_2} \qquad \forall i_1 \in I \backslash \{i^E\}, i_2 \in I \backslash I^L \tag{4.9}$$

$$y_i \leqslant \sum_{m \in M} Q_m z_{mi} \qquad \forall i \in I \backslash \{I^L \cup \{i^E\}\} \tag{4.10}$$

$$t^{arr}_{i_2} - t^{dep}_{i_1} + (1 - x_{i_1,i_2}) N_2 \geqslant d_{i_1 i_2} \qquad \forall i_1, i_2 \notin I^E \tag{4.11}$$

$$t^{arr}_{i^-} - t^{dep}_{i^+} \geqslant d_{i^+ i^-} \qquad \forall (i^+, i^-) \in (I^+ : I^-) \tag{4.12}$$

$$t^{dep}_i \geqslant t^T_i + t^H_i \qquad \forall i \in I \tag{4.13}$$

$$t^{dep}_i - t^{arr}_i \geqslant t^H_i \qquad \forall i \in I \tag{4.14}$$

$$t^{dep}_{i^L_m} \geqslant t^0_m \qquad \forall m \in M \tag{4.15}$$

$$l_i - t^{dep}_i \geqslant -t^T_i \qquad \forall i \in I \tag{4.16}$$

$$\sum_t to_{it} \geqslant t^{arr}_i \qquad \forall i \in I \tag{4.17}$$

$$\sum_t to_{it} \geqslant t^T_i \qquad \forall i \in I \tag{4.18}$$

$$\sum_t to_{it} < t^{dep}_i \qquad \forall i \in I \tag{4.19}$$

$$\sum_t o_{it} = 1 \qquad \forall i \in I \tag{4.20}$$

$$\sum_{i \in \mathcal{O}_c} \sum_{\tau = t - t^H_i + 1}^{t} o_{i\tau} = 1 \qquad \forall c \in C \tag{4.21}$$

$$x_{i_1,i_2}, z_{mi}, o_{it} \in \{0, 1\} \qquad \forall i_1, i_2, i \in I, m \in M, t \in [0, T_{max}] \tag{4.22}$$

$$y_i, t^{arr}_i, t^{dep}_i, l_i \in \mathbb{R}^+ \qquad \forall i \in I \tag{4.23}$$

The objective function (4.1) aims at minimizing the total lateness over all operations.

Assigning operations to AGVs Constraints (4.2) and (4.3) ensure that there is only one immediate predecessor and one immediate successor, respectively, for each operation. Each operation must be assigned to exactly one AGV (constraints (4.4)). Both the pick-up and delivery operations of a given job must be assigned to the same vehicle (constraints (4.5)). If an operation is an immediate successor of the first operation performed by a vehicle, then both operations must be assigned to the same vehicle (constraints (4.6)); the same applies when two operations have a precedence relation (constraints (4.7)).

Balancing vehicle loads The vehicle load after performing an initial operation is updated by constraints (4.8). If operation i_2 assigned to a given AGV is an immediate successor of operation i_1, then the vehicle load after performing i_2 is given by the vehicle load after performing i_1 adjusted by the changes resulting from the completion of operation i_2 (constraints (4.9)); this is a *big-M* constraint where N_1 is a constant number large enough to render the constraint redundant if $x_{i_1,i_2} = 0$. The load of the vehicle after performing an operation is constrained above by the vehicle capacity (constraints (4.10)).

Balancing travel times Constraints (4.11) enforce the requirement that, if operation i_2 immediately succeeds operation i_1, the arrival time of i_2 must be at least greater than the departure time of i_1 plus the travel time from the service location of i_1 to that of i_2; this is a *big-M* constraint where N_2 is a constant number large enough to render the constraint redundant if $x_{i_1,i_2} = 0$. Constraints (4.12) ensure that no cycles are created since the arrival time of the delivery operation of a given job is set to be at least greater than the departure time of the pick-up operation plus the corresponding travel time. To ensure feasibility, constraints (4.13) and (4.14) forbid the departure time of a pick-up operation to be less than its release plus handling time, or less than its arrival time plus handling time, respectively. The departure time of the first operation of each vehicle is given by constraints (4.15).

Lateness of operations The lateness in completing an operation is defined as the difference between its departure time and target time (constraints (4.16)).

Occupancy of service locations The service starting time of any operation is set to be greater than or equal to the arrival time at its service location (constraints (4.17)), greater than or equal to the target time of the operation (constraints (4.18)) and less than the departure time from the service location (constraints (4.19)). Each operation can only start at a given time (constraints (4.20)) and must be serviced by one crane at any point in time (constraints (4.21)).

Variable values Finally, constraints (4.22) and (4.23) define the allowable values for the decision variables.

4.4 A new column generation-based procedure for solving a variant of the AGVDP

In this section, we consider a variant of the AGVDP described in Section 4.2, in which we impose the additional constraint that each AGV should not serve more than a fixed number of jobs. This requirement contributes to the balancing of the workload of AGVs and, hence, provides better coordination between the AGV operations and other types of handling equipment within the logistics chain at automated container terminals.

4.4.1 Set Covering Formulation

We develop a new set covering formulation of the multi-load AGVDP defined over a finite time horizon in which an upper bound (say γ) on the number of jobs that can be performed by each AGV is imposed.

Let s denote a sequence of operations of size $1 + 2\gamma$ for a given AGV; the sequence includes an initial operation plus the pick-up and delivery operations of at most γ jobs. A sequence s is said to be *feasible* if (i) each pick-up operation of a given job precedes the corresponding delivery operation in the sequence and (ii) the load requirement after performing an operation at a position of the sequence does not exceed the unused capacity of the corresponding vehicle. Furthermore, we introduce S to denote the set of all feasible sequences s, $\overline{S}_{c,t} \subseteq S$, the set of all feasible sequences $s \in S$ served by crane c at time t and δ_s, the total lateness associated to the sequence s. A summary of the above notation is given in Table 4.2.

S	: set of all feasible sequences
$s \in S$	: a sequence of pick-up and delivery operations
δ_s	: lateness associated to sequence s
γ	: maximum number of jobs allowed in a sequence s
$k \in K = \{1, \ldots, 1 + 2\gamma\}$	: set of possible positions of the sequence s
$\overline{S}_{ct}$	: set of sequences s occupying crane c at time t
α	: cost coefficient per unit of lateness

Table 4.2: Notation: set covering formulation

To develop a set covering formulation of the AGVDP, we define a binary decision variable ϑ_s that is equal to 1 if and only if sequence $s \in S$ belongs to the optimal solution. The resulting set covering formulation of the problem, called SC, is given in Model (4.24)-(4.28).

Set covering formulation (SC)

Minimize

$$Z(SC) = \alpha \sum_{s \in S} \delta_s \vartheta_s \tag{4.24}$$

subject to:

$$\sum_{s \in S: i \in s} \vartheta_s \geq 1 \qquad \forall i \in I \tag{4.25}$$

$$\sum_{s \in S} \vartheta_s \leq |M| \tag{4.26}$$

$$\sum_{s \in \overline{S}_{c,t}} \vartheta_s \leq 1 \qquad \forall c \in C, t \in [0, T_{max}] \tag{4.27}$$

$$\vartheta_s \in \{0, 1\} \qquad \forall s \in S \tag{4.28}$$

Objective (4.24) minimizes the total delay associated with the sequences which are present in the optimal solution. Constraints (4.25) impose that each operation $i \in I$ is assigned to at least one sequence. Inequality (4.26) forces the solution to contain no more than $|M|$ sequences. Finally, constraints (4.27) specify the feasibility of crane occupancy and constraints (4.28) define the allowable ranges for the decision variables.

4.4.2 Column generation procedure

Problem SC cannot be solved directly since the number of columns may be enormous even for AGVDP instances of moderate size. In order to solve Model (4.24)-(4.28), we initially omit the crane occupancy constraints (4.27) and then define the dual problem, called DSC, of the linear relaxation of the resulting SC, which can be used to generate a valid lower bound to the AGVDP. We propose a column generation procedure (referred to as Procedure CG) whereby the DSC is solved at each iteration in order to find the best set of feasible sequences that satisfy constraints (4.25)-(4.26).

Let π_i, $i \in I$ be the dual variables associated with constraints (4.25) and β denote the dual variable of constraint (4.26). The column generation is performed using the following steps.

Procedure CG

Step 1. Define set $S' \leftarrow SEQ$ (initial set of feasible sequences obtained using the heuristic described in Section 4.4.4 or any random set of sequences).
Step 2. Solve DSC assuming $S = S'$ to get the optimal dual values $\pi_i^* \forall i \in I$ and β^*.
Step 3. Using the optimal dual values resulting from Step 2, find sequence s^* that achieves maximum violation of the dual constraint given by:

$$\sum_{i \in I: i \in s} \pi_i - \beta \leq \alpha \delta_s \tag{4.29}$$

Sequence s^* is obtained by solving the sub-problem, given by Model 4.4.3, described in Section 4.4.3.
Step 4. If no such sequence exists, exit procedure. Otherwise, append s^* to S' and return to Step 2.

At the end of Step 4, a set of sequences $S' \subseteq S$ is found by Procedure CG. A feasible solution to Model (4.24)-(4.28) is then obtained using set S' and a Branch-and-Bound algorithm; the value of the resulting solution is a valid lower bound to the AGVDP.

4.4.3 Sub-problem: finding the sequence of maximum profit

In this section, we formulate and solve the problem of finding the sequence of operations which achieves maximum violation of the dual constraint (4.29), namely, the sequence s^* of maximum profit. The main idea is to construct a feasible sequence for an AGV initially consisting of $1 + 2\gamma$ positions to which operations can be assigned. In order to develop a mathematical formulation of this problem, the following notation has to be defined. Let $k \in K = \{1, \ldots, 1 + 2\gamma\}$ be the set of possible positions in a sequence of operations. We define the following sets of decision variables:

- Binary variable:
  $\zeta_{ik} = \begin{cases} 1, & \text{if operation } i \in I \text{ is assigned to position } k \in K \\ 0, & \text{otherwise} \end{cases}$

- Continuous variables with integrality property:
  μ_k: load of the AGV after performing the operation assigned to position $k \in K$ of a sequence
  $\tau_k^{arr} \backslash \tau_k^{dep}$: the arrival\departure time at\from the service location before\after performing the operation assigned to position $k \in K$ of a sequence

The mathematical formulation of the sub-problem is given in Model 4.4.3.

Sub-problem

Maximize:

$$\sum_{k \in [1,|I|]} \sum_{i \in I} \pi_i^* \zeta_{ik} - \alpha \sum_{k \in [1,|I|]} [t_k^{dep} - \sum_{i \in I} t_i^T \zeta_{ik}] \tag{4.30}$$

subject to:

$$\sum_{k \in K} \zeta_{ik} \leq 1 \qquad \forall i \in I \tag{4.31}$$

$$\sum_{i \in I} \zeta_{ik} \leq 1 \qquad \forall k \in K \tag{4.32}$$

$$\sum_{i \in I} \zeta_{ik} - \sum_{i \in I} \zeta_{ik-1} \leq 0 \qquad \forall k \in K \setminus \{1\} \tag{4.33}$$

$$\zeta_{i^+k} - \sum_{k'=k+1,\ldots,|I|} \zeta_{i^-k'} \leq 0 \qquad \forall (i^+, i^-) \in (I^+, I^-), \; k \in K \setminus \{1\} \tag{4.34}$$

$$\sum_{i \in I^L} \zeta_{i1} = 1 \tag{4.35}$$

$$\sum_{i \in I^L} \zeta_{ik} = 0 \qquad \forall k \in K \setminus \{1\} \tag{4.36}$$

$$\tau_k^{dep} - \tau_k^{arr} - \sum_{i \in I} t_i^H \zeta_{ik} = 0 \qquad \forall k \in K \tag{4.37}$$

$$\tau_k^{arr} - \tau_{k-1}^{dep} - d_{i_1,i_2}(\zeta_{i_1,k-1} + \zeta_{i_2,k} - 1) - N_3(\sum_{i \in I} \zeta_{ik} - 1) \geq 0 \qquad \forall i_1 \in I, i_2 \in I, \; k \in K \setminus \{1\} \tag{4.38}$$

$$\tau_k^{dep} - \sum_{i \in I} (t_i^T + t_i^H)\zeta_{ik} \geq 0 \qquad \forall k \in K \setminus \{1\} \tag{4.39}$$

$$\mu_k - \mu_{k-1} - \sum_{i \in I} q_i \zeta_{ik} = 0 \qquad \forall k \in K \setminus \{1\} \tag{4.40}$$

$$\mu_k \leq \sum_{i \in I^L} \zeta_{i1} Q_i \qquad \forall k \in K \tag{4.41}$$

$$\zeta_{ik} \in \{0, 1\} \qquad \forall i \in I, k \in K \tag{4.42}$$

$$\tau_k^{dep}, \mu_k \in \mathbb{Z}^+ \qquad \forall k \in K \tag{4.43}$$

The objective function (4.30) aims at maximizing the profit of the operations inserted in the sequences using as profit the dual variables π_i^* minus the total delay caused by the generated sequences.

Assigning operations to sequences Constraints (4.31) and (4.32), respectively, ensure that each operation must be assigned to at most one position in a sequence and in each position there must be at most one operation. Constraints (4.33) allow an operation to be inserted at a position in the sequence only if the previous position is occupied. Constraints (4.34) force the pick-up operation of a job to precede its corresponding delivery operation. Constraints (4.35) and constraints (4.36), respectively, guarantee that the first position of a sequence is assigned one of the initial AGV operations whereas operations linked to vehicles cannot be assigned from the second position onwards.

Balancing travel times Constraints (4.37) impose that the departure time from a specific position in the sequence is greater than the arrival time at the same position plus the handling

time of the assigned operation. Constraints (4.38) specify that, if position k of a sequence is occupied, then the arrival time at this position must be greater than the departure time from the previous position $k-1$ plus the travel time between the locations of the operations assigned to the two positions; this is a *big-M* constraint where N_3 is a constant number large enough to render the constraint redundant if $\sum_{i \in I} \zeta_{ik} = 0$. Finally, no departure from a position k is possible before the target/release time plus the handling time of the assigned operation has elapsed (constraints (4.39)).

Balancing vehicle loads Constraints (4.40) ensure that the vehicle load at a certain position must be equal to the load of the previous position plus the load capacity requirement of the assigned operation; furthermore an upper bound is imposed on the vehicle load capacity by constraints (4.41).

Variable values Constraints (4.42) and (4.43) define the allowable values for the decision variables.

Similarly to problem SC (Model (4.24)-(4.28)), it is impractical to solve the Sub-problem (Model 4.4.3) since the computational time required to generate a solution depends on the allowable number of jobs (γ) in a given sequence. Hence, in our computational experimentation presented in Section 4.6, we limited the number of jobs performed by each AGV to four ($\gamma = 4$); this implementation, of course, leads to a valid lower bound to the optimal solution of Model (4.24)-(4.28). Furthermore, due to the large number of constraints (4.38), Model 4.4.3 is initially solved by completely relaxing this set of constraints and subsequently adding, in an iterative fashion, those that are violated by the current solution to the ILP model. Detailed computational experimentation has indicated that the best results were obtained by inserting a restricted number of such violated constraints (< 10) at each iteration.

4.4.4 A heuristic randomized approach for constructing a set of feasible sequences

This section outlines a heuristic approach that can be used in the initialization step of Procedure CG described in Section 4.4.2 for solving Model (4.24)-(4.28). The proposed procedure constructs an initial set of feasible sequences to Model (4.24)-(4.28), referred to as SEQ, with the view to achieving computational savings in the implementation of the column generation procedure. Set SEQ includes all the sequences consisting of up to three jobs, generated using complete enumeration. A subset of sequences with at least four jobs, of size Lim, is also included in SEQ, constructed using a randomized heuristic approach.

The heuristic approach is described as follows. Let $\mathcal{F}_m$ be the feasible operations for each vehicle $m \in M$; these include all the pick-up operations of the problem, i.e. $\mathcal{F}_m = I^+$. We construct a number of sequences equal to the number of vehicles in the following way. Each sequence seq_m is initialized to $seq_m = \{i_m^L\}$ and the current operation of the vehicle is set to $i_m^c = i_m^L$. For each $m \in M$ we assign operation $i \in \mathcal{F}_m$ such that $t_i^T + d_{i_m^c i}$ is minimum. If $i \in I^-$, we update sets $\mathcal{F}_{m'} = \mathcal{F}_{m'} \setminus \{i\}$ for $m' \in M$. If $i \in I^+$ then $\mathcal{F}_m = \mathcal{F}_m \cup \{i^-\} \setminus \{i\}$ for $i^- \in I^-$ such that $(i, i-) \in (I^+ : I^-)$. We also set $i_m^c = i$. We repeat this step until $\cup_{m \in M} \mathcal{F}_m = \emptyset$.

The second stage of the procedure constructs a new sequence seq' from any pre-constructed sequence $seq \in SEQ$ in the following way. We set $seq' = seq$. Note that sequence seq' is

associated to a vehicle m. For each $i^+ \in seq'$, a random number $r \in [0,1]$ is drawn. If the random number is greater than $prob_d$ (the probability of deletion), the pick-up operation and its corresponding delivery operation are removed from seq'. Otherwise, if $prob_r \leqslant r < prob_d$, where $prob_r$ is the probability of replacement, i^+ is replaced with $i' = \arg\min\{i \in I^+ \setminus \{seq'\}$ s.t. $q_i = q_{i^+}\}$. The corresponding delivery operations are also replaced. Finally, if $r < prob_r$, operation $i' = \arg\min\{i \in I^+ \setminus \{seq'\}$ s.t. $q_i \leq Q_m\}$ is added at the end of seq', followed by the corresponding delivery operation. If seq' contains at least four jobs and if the limit sequences is not exceeded, seq' is appended to SEQ. This process is repeated while $|SEQ| \leqslant Lim$.

4.5 An illustrative example

In order to illustrate Procedure CG described in Section 4.4, we use an example. We consider a container terminal with $|C| = 4$ cranes (two quay cranes, denoted by QC1 and QC2, and two stacking cranes, denoted by SC1 and SC2), $|M| = 2$ AGVs and $|J| = 2$ jobs. Both AGVs are initially located at $QC1$ and each has a capacity of $Q_1 = Q_2 = 2$ units. The pick-up operation of job 1 is located at crane $SC1$ and its delivery operation at $SC2$, while the pick-up/delivery operations for job 2 are located at $SC2/SC1$, respectively. Both jobs require a capacity of 1 unit. The travel time required by the AGVs between each pair of cranes is equal to 1 time unit. We define set of operations $I = \{i_1^L, i_2^L, p1, p2, d1, d2\}$ where $p1$, $p2$ refer to the pick-up operations of jobs 1,2, respectively, and $d1$, $d2$ to the delivery operations of job 1,2, respectively.

Table 4.3 shows the relaxed problem SC (Section 4.4.1) consisting of an initial set S' of three sequences. Let θ_s be the decision variable associated to sequence $s \in S'$, with $s = 1$ being the sequence including all the operations, i.e. $\{i_1^L, i_2^L, p1, p2, d1, d2\} = I$, $s = 2$ the trivial sequence $\{i_1^L\}$ and $s = 3$ the trivial sequence $\{i_2^L\}$. Constraints (4.25) of Model (4.24)-(4.28) are given by rows $c1 - c6$ of this table and constraints (4.26) are represented by row $c7$. Let $\pi_1 - \pi_6$ and β denote the corresponding dual variables for each row.

Minimize									
	obj $\overline{z} =$	$1000\theta_1$							
subject to:									
	c1:	θ_1	$+$	θ_2			$\geq$	1	(π_1)
	c2:	θ_1			$+$	θ_3	$\geq$	1	(π_2)
	c3:	θ_1					$\geq$	1	(π_3)
	c4:	θ_1					$\geq$	1	(π_4)
	c5:	θ_1					$\geq$	1	(π_5)
	c6:	θ_1					$\geq$	1	(π_6)
	c7:	θ_1	$+$	θ_2	$+$	θ_3	$\leq$	2	(β)
End									

Table 4.3: Illustrative example: problem DSC (initialization step)

Table 4.4 shows the output of problem DSC obtained for five consecutive iterations of Procedure CG (Section 4.4.2); each row of the table corresponds to one iteration. Columns identified by $\theta_1 - \theta_7$ report the set of optimal values of the decision variables in set S'; the procedure

starts by considering the first three sequences described above, adding one new sequence at each iteration. The remaining columns of Table 4.4 show $\overline{z}$, the current optimal solution value of problem DSP and the set of values $(\pi_1 - \pi_6, \beta)$ of the dual variables associated to the primal constraints (4.25) and (4.26), respectively. The new sequence s^*, the corresponding value of lateness, δ, and the value $\rho = \sum_{i \in I: i \in s^*} \pi_i - \beta - \alpha\delta$ are also given in Table 4.4 for each iteration. Note that Procedure CG is terminated after 5 iterations since no sequence which violates constraint (4.29) can be found. Table 4.5 shows problem DSC solved at the end of the column generation procedure (root node) producing a lower bound value of $\overline{z} = 12$.

The illustrative example is solved to optimality by adding constraints (4.27) (shown in rows $c8$ and $c9$ of Table 4.5) and (4.28) and using a Branch-and Bound procedure. The resulting optimal solution has a value $Z(SC) = 12$.

θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	$\overline{z}$	π_1	π_2	π_3	π_4	π_5	π_6	β	sequence s^*	δ	ρ
1.0	1.0	0.0					1000	0	0	1000	0	0	0	0	i_1^L-p1-d1	5	995
1.0	1.0	0.0	0.0				1000	0	0	5	995	0	0	0	i_1^L-p2-d2	7	988
0.5	0.0	0.5	0.5	0.5			506	0	494	499	501	0	0	494	i_2^L-p1-d1-p2-d2	16	984
0.0	0.0	0.5	0.5	0.5	0.5		14	0	2	7	9	0	0	2	i_2^L-p1-d1	5	2
0.0	0.0	0.0	0.0	1.0	0.0	1.0	12	0	0	5	7	0	0	0		-	0

Table 4.4: Illustrative example: column generation procedure (5 iterations)

Minimize		
	obj $\overline{z} =$	$1000\theta_1 + 5\theta_4 + 7\theta_5 + 16\theta_6 + 5\theta_7$
Subject To		
	c1:	$\theta_1 + \theta_2 + \theta_4 + \theta_5 \geq 1$
	c2:	$\theta_1 + \theta_3 + \theta_6 + \theta_7 \geq 1$
	c3:	$\theta_1 + \theta_4 + \theta_6 + \theta_7 \geq 1$
	c4:	$\theta_1 + \theta_5 + \theta_6 \geq 1$
	c5:	$\theta_1 + \theta_4 + \theta_6 + \theta_7 \geq 1$
	c6:	$\theta_1 + \theta_5 + \theta_6 \geq 1$
	c7:	$\theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5 + \theta_6 + \theta_7 \leq 2$
End		
"crane occupancy"	c8:	$\theta_4 + \theta_6 + \theta_7 \leq 1$
constraints	c9:	$\theta_4 + \theta_6 + \theta_7 \leq 1$

Table 4.5: Illustrative example: solution to Model (4.24)-(4.28) (root node)

4.6 Computational experiments

The aim of this section is to analyze the computational performance of the proposed MIP model (Model (4.1)-(4.23)) and column generation approach using Model (4.24)-(4.28) for the multi-load AGVDP. Both methodologies were coded in Microsoft Visual Studio C++ and run on an Intel Core 2 Processor (2.5GHz with 3.5GB of RAM, Windows Operating System) using IBM ILOG CPLEX v12.1. We describe our test set generation procedure based on four

different scenarios. Computational results for a total of 40 test instances are reported for both methodologies. Note that in all our computational experiments, the cost coefficient α is set to 1.

4.6.1 Test problem generation

As a test bed, we generate a set of test instances based on four scenarios, following a similar approach to the one adopted by [57]. In particular, two layouts are defined as follows. The first one includes six cranes served by a fleet of three AGVs. The three cranes are lined up in a row along the berth, while the remaining cranes are located in a row in the berthing area, a typical design in highly automated container ports. All travel times required by the AGVs are set to 1 time unit for a trip between two adjacent cranes and all distances are assumed to be symmetrical. The design of the second layout is similar, with the number of cranes and AGVs doubled. For both layouts we investigate two different workloads specified in terms of the number of jobs that have to be processed: 8 and 12 jobs for the first layout and 14 and 18 jobs for the second layout. The target/release time for each pick-up operation is drawn from the uniform distribution $U[1, 20]$ (the corresponding delivery operation has the same target time). The crane associated to each operation is randomly generated with equal likelihood given to all possible connections between quay and stacking cranes. The handling time of each pick-up and delivery operation is set to 1 time unit. For all remaining operations the handling time is 0. The share of 40/45ft containers is set to 30%. For each of the scenarios 10 problem instances were randomly generated resulting in a total of 40 test instances. Table 4.6 displays the settings used for the layout of the guide-path, the number of jobs and number of AGVs for each of the four scenarios.

Scenario	#Cranes	#Jobs	#AGVs
1	6	8	3
2	6	12	3
3	12	14	6
4	12	18	6

Table 4.6: Scenario settings for generating test instances

4.6.2 Implementation and numerical results

The performance of the multi-load AGVDP was evaluated for the following proposed methods:

1. MIP Model (4.1)-(4.23)
2. Relaxed MIP Model (4.1)-(4.23) in which the "crane occupancy" constraints (4.17)-(4.21) are relaxed
3. Same as (i) including the additional constraint for limiting the number of jobs that can be processed by each AGV by a fixed number
4. Same as (ii) including the additional constraint for limiting the number of jobs that can be processed by each AGV by a fixed number
5. Set covering formulation Model (4.24)-(4.28)

6. Relaxed Model (4.24)-(4.28) in which the "crane occupancy" constraints (4.27) are relaxed

Note that in the implementation of methods (iii) and (iv) we add the following constraint to the corresponding mathematical formulation:

$$\sum_{i \in I+} z_{mi} \leq \gamma \qquad \forall m \in M \tag{4.44}$$

All test instances were solved using any of the above methods terminated when either the optimal solution was found or a time limit of 1000 or 10000 CPU seconds elapsed for Model (4.1)-(4.23) or Model (4.24)-(4.28), respectively; a higher time limit was set for Model (4.24)-(4.28) to allow the computation of a better quality bound for large-size instances.

The results obtained for each test instance using any of the solution methods (i) to (vi) are displayed in Tables 7 to 10. Each instance is uniquely identified by $I\_|M|\_|J|\_|C|_k$ consisting of $|M|$ vehicles, $|J|$ jobs and $|C|$ service locations. Index $k \in \{1, \ldots, 10\}$ denotes the k^{th} instance within a given scenario. The columns of these tables are defined as follows.

- UB, UB1, UB2: Cost of the optimal AGVDP solution found by the model proposed in [57] or Model (4.1)-(4.23) or Model (4.24)-(4.28) (methods (i)-(vi)), respectively, (or cost of the best solution found).
- LB, LB1, LB2: Lower bound computed at the root node of the Branch-and-Bound (B&B) algorithm using the model proposed in [57] or Model (4.1)-(4.23) (obtained using CPLEX 12.1) or the continuous relaxation of Model (4.24)-(4.28), respectively.
- $\%e_1, \%e_2$: Lower bound LB1 or LB2 as a percentage of the best upper bound found (minimum of UB1 and UB2), for a given instance,
  $\%e_1 = 100\frac{LB1}{\min\{UB1,UB2\}}$
  $\%e_2 = 100\frac{LB2}{\min\{UB1,UB2\}}$
- t_{root}: Computational time (in CPU seconds) for finding LB, LB1 or LB2
- $\%gap, \%gap_1$: Percentage deviation from UB, UB1 of the best lower bound LB*, LB1* found by the B&B algorithm at the end of the time limit using the model proposed in [57] or Model (4.1)-(4.23) , i.e.
  $\%gap = 100\frac{UB-LB^*}{UB1}$
  $\%gap_1 = 100\frac{UB1-LB1^*}{UB1}$
- $\%gap_2$: Percentage deviation from UB2 of LB2, i.e.
  $\%gap_2 = 100\frac{UB2-LB2}{UB2}$
- t_{tot}: Total computational time (in CPU seconds) for finding UB, UB1 or UB2
- $|S'|$: Number of columns (in thousands) generated by Procedure CG used by either method (v) or (vi)

For comparison purposes, we also implemented the MIP model proposed in [57], here referred to as GRUN04, and solved all instances on the same computer; the results obtained are also reported in Table 4.7.

Table 4.7 demonstrates the performance of the proposed MIP Model (4.1)-(4.23) for the general AGVDP in which no restriction on the number of jobs assigned to each AGV is placed. The results displayed for the Relaxed Model (4.1)-(4.23) are compared to GRUN04 for all instances in the four scenarios. The table shows that the Relaxed Model (4.1)-(4.23) clearly outperforms GRUN04 for all largest and hardest instances; our model solved 65% of all instances to optimality, in an average CPU time of about 7.5 minutes compared to only 33% of these problems being solved by GRUN04 with an average time of 12.2 minutes. More specifically, the instances in scenarios 3 and 4 were beyond the capabilities of GRUN04 given the CPU time limit; the average optimality gap for those instances in all scenarios not solved to optimality was 37%. On the other hand, the corresponding average optimality gap computed by the Relaxed Model (4.1)-(4.23) is only 11.45%.

Table 4.8 reports the behaviour of Model (4.1)-(4.23) for a fixed number of jobs performed by an AGV. In particular, results for $\gamma = 3, 4, 5$ jobs per sequence are shown. Note that no results are reported for instances in scenario 2 and $\gamma = 3$ since there was no feasible solution found by Model (4.1)-(4.23) or Relaxed Model (4.1)-(4.23) . From Tables 7 and 8, we observe that the behaviour of Model (4.1)-(4.23) or Relaxed Model (4.1)-(4.23) is not significantly affected by these additional constraints in terms of computational time or percentage of instances solved to optimality in each scenario or the value of the best upper bound found. For example, we note that without adding the crane occupancy constraints (Relaxed Model (4.1)-(4.23)), the percentage of instances solved to optimality for $\gamma = 4$ are 100%, 70%, 60% and 20% for scenario 1, 2, 3 and 4, respectively, and 63% over all instances. When the crane occupancy constraints are added, the corresponding percentages are 100%, 80%, 70% and 10% for scenario 1, 2, 3 and 4, respectively, and 65% over all scenarios. However, for a given value of γ, the inclusion of crane occupancy constraints in the AGVDP impacts the quality of solution as reflected in the corresponding values of the average optimality gap (over all instances obtained by Model (4.1)-(4.23) and Relaxed Model (4.1)-(4.23) , respectively).

Table 4.9 shows the computational results derived from the column generation approach. We report results for $\gamma = 3, 4$ for methods (v) Model (4.24)-(4.28) and (vi) Relaxed Model (4.24)-(4.28). As in Table 4.8, no results are reported for scenario 2 instances and $\gamma = 3$ as no feasible solution was found for these instances. The table shows that the feasible solutions obtained with the column generation approach node are of good quality. For the Relaxed Model (4.24)-(4.28), the column generation procedure is capable of closing the optimality gap in 97% of the instances for $\gamma = 3$ and 75% of the instances for $\gamma = 4$ (all the instances in scenario 4 were found to be beyond the capabilities of Procedure CG). The results corresponding to method (v) Model (4.24)-(4.28) are also generally of good quality; the optimal solution is found for 46% of the instances over $\gamma = 3, 4$.

Finally, Table 4.10 compares lower bounds LB1 and LB2. The table shows that LB2 systematically outperforms LB1. In particular, $\%e_2$ is 100% over all instances for $\gamma = 3$, compared to 89% for $\%e_1$. For $\gamma = 4$, LB2 is found to be equal to the optimal value for all instances in scenarios 1-3 ($\%e_2 = 100\%$) while $\%e_1$ is equal to 87%, 76% and 93% for scenarios 1,2 and 3, respectively.

4.7 Conclusions

The multi-load AGV dispatching problem in automated container terminals is a complex and dynamic problem and solutions to this problem need to be capable of adapting to rapid

changes. For this reason, most developments in the literature focus on priority rules or heuristics which can be applied in real-time. The main contribution of this chaper is the development of effective solution approaches for finding optimal and feasible solutions to the multi-load AGV dispatching problem. Our focus is on AGVs which are operated in a dual carrier mode, i.e. capable of carrying either a 40/45-ft container or two 20-ft containers. The solution approaches proposed in this chaper are capable of providing a platform for benchmarking heuristic methodologies.

Our numerical results reveal that the new complete MIP formulation we are proposing is capable of solving efficiently many of the test instances in the literature; in some cases, however, the solver is not able to close the optimality gap within the time limit of 1000 CPU seconds. In order to obtain stronger lower bounds we have developed a set covering formulation of the problem. Its continuous relaxation is solved using a new column generation approach. This approach shows that if the number of jobs assigned to each vehicle is fixed, strong bounds can be derived. Moreover feasible solutions of good quality are obtained for most instances. In many cases it is also possible to even close the optimality gap within a reasonable amount of time.

	GRUN04					(i) Model 1					(ii) Relaxed Model 1				
Instance	**UB**	**LB**	$\mathbf{t}_{root}$	**%gap**	$\mathbf{t}_{tot}$	**UB1**	**LB1**	$\mathbf{t}_{root}$	$\mathbf{\%gap}_1$	$\mathbf{t}_{tot}$	**UB1**	**LB1**	$\mathbf{t}_{root}$	$\mathbf{\%gap}_1$	$\mathbf{t}_{tot}$
I_3_8_6$_1$	47	24.6	0.8	0.00	4.4	47	41.0	1.4	0.00	4.4	47	41.0	0.9	0.00	2.2
I_3_8_6$_2$	45	23.4	0.5	0.00	2.7	45	42.0	1.3	0.00	1.9	45	42.0	0.8	0.00	2.1
I_3_8_6$_3$	45	26.8	0.4	0.00	4.0	45	37.0	1.1	0.00	2.5	45	37.0	0.9	0.00	3.3
I_3_8_6$_4$	46	24.4	0.5	0.00	3.7	46	42.0	1.3	0.00	1.8	46	42.0	0.9	0.00	1.1
I_3_8_6$_5$	48	22.2	0.5	0.00	7.3	48	40.0	1.1	0.00	8.2	48	40.0	0.8	0.00	3.7
I_3_8_6$_6$	57	27.7	0.6	0.00	42.4	57	45.0	1.6	0.00	67.6	57	42.0	0.9	0.00	135.5
I_3_8_6$_7$	44	28.0	0.6	0.00	2.9	44	42.0	1.1	0.00	1.5	44	40.0	1.2	0.00	1.3
I_3_8_6$_8$	49	24.0	0.4	0.00	5.6	49	41.0	1.1	0.00	3.7	49	41.0	0.8	0.00	3.0
I_3_8_6$_9$	44	24.5	0.5	0.00	3.9	44	41.0	1.1	0.00	2.8	44	40.0	0.9	0.00	1.7
I_3_8_6$_{10}$	45	25.4	0.4	0.00	4.6	49	43.0	1.6	0.00	2.0	45	41.0	0.9	0.00	2.2
Scenario 1- Averages:				**0.00**	**8.2**				**0.00**	**9.6**				**0.00**	**15.6**
I_3_12_6$_1$	72	36.4	1.6	0.00	595.1	77	61.0	2.9	0.00	71.2	72	58.0	3.5	0.00	36.6
I_3_12_6$_2$	72	37.3	2.9	0.00	778.9	77	59.0	3.7	0.00	67.5	72	56.0	2.2	0.00	35.4
I_3_12_6$_3$	81	34.1	1.9	27.15	1000.0	82	61.0	3.0	9.25	1000.0	84	55.0	2.0	22.49	1000.0
I_3_12_6$_4$	75	37.0	2.8	0.00	855.1	75	58.0	2.5	0.00	214.0	75	57.0	2.2	0.00	66.9
I_3_12_6$_5$	77	35.4	1.5	15.43	1000.0	77	57.0	2.5	0.00	154.6	77	57.0	2.3	1.33	1000.0
I_3_12_6$_6$	88	40.4	2.1	14.98	1000.0	91	65.0	3.5	6.44	1000.0	88	60.0	2.4	0.00	564.7
I_3_12_6$_7$	77	38.1	2.4	11.39	1000.0	70	61.0	4.2	0.00	30.7	69	59.0	2.5	0.00	20.4
I_3_12_6$_8$	77	36.6	2.4	11.42	1000.0	78	58.0	3.4	3.97	1000.0	77	57.0	2.7	0.00	832.3
I_3_12_6$_9$	83	33.8	2.9	24.44	1000.0	83	58.0	2.9	3.27	1000.0	83	58.0	2.2	19.29	1000.0
I_3_12_6$_{10}$	67	33.3	3.0	6.68	1000.0	68	58.0	3.5	0.00	113.3	67	55.0	3.5	0.00	129.4
Scenario 2- Averages:				**11.15**	**922.9**				**2.29**	**465.1**				**4.31**	**468.6**
I_6_14_12$_1$	78	34.9	3.1	15.26	1000.0	81	81.0	11.3	0.00	19.6	78	78.0	7.6	0.00	17.9
I_6_14_12$_2$	86	28.0	1.7	28.75	1000.0	86	80.0	9.8	0.00	128.4	86	78.0	4.4	0.00	245.9
I_6_14_12$_3$	106	34.0	3.5	37.42	1000.0	105	96.0	9.6	6.02	1000.0	102	90.0	7.1	7.43	1000.0
I_6_14_12$_4$	115	28.0	1.8	36.03	1000.0	111	99.0	11.7	5.78	1000.0	111	98.0	5.8	8.65	1000.0
I_6_14_12$_5$	84	28.0	2.2	24.13	1000.0	87	85.0	18.3	0.00	20.2	84	82.0	9.7	0.00	16.7
I_6_14_12$_6$	96	31.3	3.3	24.17	1000.0	96	95.0	9.3	0.00	69.2	96	94.0	5.9	0.00	73.7
I_6_14_12$_7$	109	33.5	4.0	37.93	1000.0	102	97.0	8.2	0.00	446.1	98	91.0	5.7	0.00	752.0
I_6_14_12$_8$	91	28.0	2.2	27.21	1000.0	87	86.0	5.9	0.00	126.2	87	85.0	10.1	0.00	142.7
I_6_14_12$_9$	107	28.0	1.8	42.04	1000.0	104	89.0	10.9	7.78	1000.0	103	87.0	8.4	6.80	1000.0
I_6_14_12$_{10}$	96	37.0	3.4	16.58	1000.0	97	94.0	8.5	0.00	58.0	96	93.0	7.7	0.00	50.5
Scenario 3- Averages:				**28.95**	**1000.0**				**1.96**	**386.8**				**2.29**	**429.9**
I_6_18_12$_1$	162	42.3	103.8	58.62	1000.0	122	104.0	31.1	10.66	1000.0	115	103.0	26.6	8.71	1000.0
I_6_18_12$_2$	143	52.3	7.6	38.16	1000.0	131	116.0	42.1	6.49	1000.0	126	113.0	12.6	5.54	1000.0
I_6_18_12$_3$	187	45.2	352.6	57.88	1000.0	-	115.0	40.9	-	1000.0	123	110.0	13.2	7.27	1000.0
I_6_18_12$_4$	146	51.3	8.1	39.52	1000.0	157	104.0	22.0	27.68	1000.0	130	102.0	16.3	16.72	1000.0
I_6_18_12$_5$	179	36.0	5.0	63.23	1000.0	-	110.2	27.2	-	1000.0	138	108.0	13.8	21.73	1000.0
I_6_18_12$_6$	194	43.8	222.6	59.82	1000.0	-	107.0	40.3	-	1000.0	-	107.0	19.9	-	1000.0
I_6_18_12$_7$	118	47.8	9.6	27.63	1000.0	109	107.0	52.3	0.00	129.3	106	104.0	5.8	0.00	232.9
I_6_18_12$_8$	123	52.8	7.0	29.12	1000.0	120	115.0	23.2	0.00	429.4	117	109.0	27.8	0.00	678.6
I_6_18_12$_9$	266	42.8	289.2	72.88	1000.0	-	114.0	45.9	-	1000.0	-	110.0	18.1	-	1000.0
I_6_18_12$_{10}$	164	36.0	5.3	61.46	1000.0	-	109.0	38.9	-	1000.0	-	106.0	24.1	-	1000.0
Scenario 4- Averages:				**50.83**	**1000.0**				**8.96**	**855.9**				**8.57**	**891.2**
All instances - Averages:				**22.73**	**732.8**				**2.50**	**429.4**				**3.40**	**451.3**
%Instances solved to optimality:				**33%**					**60%**					**65%**	

Table 4.7: Computational performance of MIP Model (4.1)-(4.23)

Table 4.8: Computational performance of MIP Model (4.1)-(4.23) with fixed number of jobs

	(iii) Model 1									(iv) Relaxed Model 1								
	$\gamma = 3$			$\gamma = 4$			$\gamma = 5$			$\gamma = 3$			$\gamma = 4$			$\gamma = 5$		
Inst.	**UB1**	**%gap$_1$**	**t$_{tot}$**	**UB1**	**%gap$_1$**	**t$_{tot}$**	**UB1**	**%gap$_1$**	**t$_{tot}$**	**UB1**	**%gap$_1$**	**t$_{tot}$**	**UB1**	**%gap$_1$**	**t$_{tot}$**	**UB1**	**%gap$_1$**	**t$_{tot}$**
I_3_8_6$_1$	47	0.00	3.8	47	0.00	7.6	47	0.00	4.8	47	0.00	2.3	47	0.00	2.6	47	0.00	2.2
I_3_8_6$_2$	45	0.00	2.1	45	0.00	3.3	45	0.00	2.7	45	0.00	2.8	45	0.00	2.3	45	0.00	2.1
I_3_8_6$_3$	45	0.00	4.0	45	0.00	3.9	45	0.00	4.5	45	0.00	3.0	45	0.00	3.5	45	0.00	2.3
I_3_8_6$_4$	46	0.00	3.1	46	0.00	2.4	46	0.00	1.6	46	0.00	1.0	46	0.00	1.5	46	0.00	1.0
I_3_8_6$_5$	48	0.00	5.4	48	0.00	9.7	48	0.00	10.4	48	0.00	4.2	48	0.00	4.9	48	0.00	5.5
I_3_8_6$_6$	57	0.00	41.3	57	0.00	67.1	57	0.00	65.3	57	0.00	124.5	57	0.00	172.1	57	0.00	98.4
I_3_8_6$_7$	44	0.00	2.0	44	0.00	2.8	44	0.00	1.9	44	0.00	1.3	44	0.00	1.8	44	0.00	1.7
I_3_8_6$_8$	49	0.00	3.8	49	0.00	4.3	49	0.00	9.1	49	0.00	2.8	49	0.00	3.7	49	0.00	3.5
I_3_8_6$_9$	44	0.00	4.5	44	0.00	3.6	44	0.00	6.1	44	0.00	2.1	44	0.00	2.0	44	0.00	3.2
I_3_8_6$_{10}$	49	0.00	3.4	49	0.00	3.2	49	0.00	4.6	45	0.00	1.8	45	0.00	1.6	45	0.00	4.1
Sc. 1 - Avg:		**0.00**	**7.3**		**0.00**	**10.8**		**0.00**	**11.1**		**0.00**	**14.6**		**0.00**	**19.6**		**0.00**	**12.4**
I_3_12_6$_1$				77	0.00	112.3	77	0.00	108.2				72	0.00	58.6	72	0.00	20.4
I_3_12_6$_2$				77	0.00	114.3	77	0.00	111.4				72	0.00	58.1	72	0.00	27.5
I_3_12_6$_3$				86	16.00	0.0	82	8.07	1000.0				79	14.32	1000.0	79	10.48	1000.0
I_3_12_6$_4$				75	0.00	122.3	75	0.00	62.6				75	0.00	179.6	75	0.00	49.4
I_3_12_6$_5$				77	0.00	200.3	77	0.00	197.6				77	13.50	1000.0	77	0.00	726.5
I_3_12_6$_6$				91	10.02	1000.0	93	13.44	1000.0				88	0.00	652.4	88	8.15	1000.0
I_3_12_6$_7$				72	0.00	153.6	70	0.00	80.8				71	0.00	156.5	69	0.00	141.3
I_3_12_6$_8$				78	0.00	925.7	78	0.01	1000.0				77	0.00	979.4	77	0.00	681.1
I_3_12_6$_9$				83	0.00	986.0	83	5.78	1000.0				83	15.83	1000.0	83	11.94	1000.0
I_3_12_6$_{10}$				68	0.00	145.6	68	0.00	126.4				67	0.00	141.6	67	0.00	116.7
Sc. 2 - Avg:					**2.60**	**376.0**		**2.73**	**468.7**					**4.37**	**522.6**		**3.06**	**476.3**
I_6_14_12$_1$	81	0.00	9.5	81	0.00	9.6	81	0.00	26.5	78	0.00	8.5	78	0.00	8.9	78	0.00	1000.0
I_6_14_12$_2$	86	0.00	210.3	86	0.00	240.5	86	0.00	137.9	86	0.00	164.6	86	0.00	293.7	86	0.00	372.6
I_6_14_12$_3$	107	8.13	1000.0	105	6.32	1000.0	105	6.67	1000.0	102	6.84	1000.0	102	8.11	1000.0	102	6.64	1000.0
I_6_14_12$_4$	115	11.05	1000.0	111	7.10	1000.0	113	8.85	1000.0	111	8.56	1000.0	111	8.65	1000.0	111	8.36	1000.0
I_6_14_12$_5$	87	0.00	41.1	87	0.00	46.9	87	0.00	34.9	84	0.00	37.4	84	0.00	19.7	84	0.00	19.6
I_6_14_12$_6$	96	0.00	130.0	96	0.00	100.5	96	0.00	89.5	96	0.00	28.8	96	0.00	76.3	96	0.00	82.1
I_6_14_12$_7$	102	3.29	1000.0	102	0.00	688.3	102	2.50	1000.0	98	0.00	666.4	98	2.04	1000.0	98	0.00	569.5
I_6_14_12$_8$	87	0.00	53.1	87	0.00	79.7	87	0.00	171.7	87	0.00	179.0	87	0.00	254.0	87	0.00	143.3
I_6_14_12$_9$	104	7.88	1000.0	105	8.91	1000.0	104	8.58	1000.0	103	1.51	1000.0	103	7.38	1000.0	103	7.43	1000.0
I_6_14_12$_{10}$	97	0.00	115.0	97	0.00	106.6	97	0.00	98.5	96	0.00	152.1	96	0.00	86.5	96	0.00	184.0
Sc. 3 - Avg:		**3.04**	**455.9**		**2.23**	**427.2**		**2.66**	**455.9**		**1.69**	**423.7**		**2.62**	**473.9**		**2.24**	**537.1**
I_6_18_12$_1$	200	47.00	1000.0	114	5.26	1000.0	334	68.26	1000.0	122	14.95	1000.0	115	9.17	1000.0	121	14.47	1000.0
I_6_18_12$_2$	147	19.73	1000.0	376	68.62	1000.0	212	44.13	1000.0	127	6.47	1000.0	126	5.56	1000.0	126	6.46	1000.0
I_6_18_12$_3$	139	15.83	1000.0	148	20.95	1000.0	131	10.69	1000.0	128	11.66	1000.0	123	7.56	1000.0	123	7.57	1000.0
I_6_18_12$_4$	190	41.89	1000.0	323	65.79	1000.0	-	-	1000.0	164	36.23	1000.0	134	21.44	1000.0	128	16.84	1000.0
I_6_18_12$_5$	295	62.04	1000.0	216	48.42	1000.0	-	-	1000.0	176	38.64	1000.0	-	-	1000.0	136	20.17	1000.0
I_6_18_12$_6$	179	37.93	1000.0	375	70.32	1000.0	419	73.44	1000.0	119	5.04	1000.0	119	4.62	1000.0	119	3.99	1000.0
I_6_18_12$_7$	112	2.68	1000.0	109	0.00	396.7	109	0.00	476.6	110	1.33	1000.0	106	0.00	116.2	106	0.00	137.6
I_6_18_12$_8$	128	8.20	1000.0	120	0.47	1000.0	120	0.00	721.9	121	4.96	1000.0	117	0.00	870.8	117	0.00	682.7
I_6_18_12$_9$	674	82.64	1000.0	288	59.28	1000.0	-	-	1000.0	141	20.23	1000.0	139	19.42	1000.0	-	-	1000.0
I_6_18_12$_{10}$	311	64.81	1000.0	175	34.83	1000.0	183	39.89	1000.0	128	16.48	1000.0	128	16.16	1000.0	125	14.64	1000.0
Sc. 4 - Avg:		**38.28**	**1000.0**		**37.39**	**939.7**		**33.77**	**919.9**		**15.60**	**1000.0**		**9.32**	**898.7**		**9.35**	**882.0**
All inst. - Avg:		**13.77**	**487.75**		**10.56**	**438.42**		**7.85**	**463.88**		**5.76**	**479.43**		**3.94**	**478.71**		**3.52**	**476.96**
%Inst. solved:		**53%**			**65%**			**60%**			**57%**			**63%**			**65%**	

	(v) Model 2		**(vi) Relaxed Model 2**							
	$\gamma = 3$	$\gamma = 4$	$\gamma = 3$				$\gamma = 4$			
Inst.	**UB2**	**UB2**	**UB2**	**%gap$_2$**	**t$_{tot}$**	**|S'|**	**UB2**	**%gap$_2$**	**t$_{tot}$**	**|S'|**
I_3_8_6$_1$	47	47	47	0.00	0.3	5.0	47	0.00	52.8	5.0
I_3_8_6$_2$	45	45	45	0.00	0.5	7.0	45	0.00	26.2	7.0
I_3_8_6$_3$	45	45	45	0.00	0.4	7.0	45	0.00	37.6	7.0
I_3_8_6$_4$	46	46	46	0.00	0.1	1.7	46	0.00	11.5	1.7
I_3_8_6$_5$	48	48	48	0.00	0.3	5.0	48	0.00	113.6	5.0
I_3_8_6$_6$	57	57	57	0.00	0.3	5.0	57	0.00	181.7	5.0
I_3_8_6$_7$	44	44	44	0.00	0.3	3.5	44	0.00	24.8	3.5
I_3_8_6$_8$	49	49	49	0.00	0.3	5.0	49	0.00	38.4	5.0
I_3_8_6$_9$	44	44	44	0.00	0.4	7.0	44	0.00	74.4	7.0
I_3_8_6$_{10}$	49	49	45	0.00	0.2	2.4	45	0.00	25.6	2.4
Sc. 1 - Avg:				**0.00**	**0.3**	**4.9**		**0.00**	**58.7**	**4.9**
I_3_12_6$_1$		89					72	0.00	272.8	60.3
I_3_12_6$_2$		105					72	0.00	402.7	60.3
I_3_12_6$_3$		85					79	0.00	308.1	60.3
I_3_12_6$_4$		75					75	0.00	116.3	60.3
I_3_12_6$_5$		77					77	0.00	201.3	60.3
I_3_12_6$_6$		94					88	0.00	273.2	60.3
I_3_12_6$_7$		83					71	0.00	331.5	60.2
I_3_12_6$_8$		81					77	0.00	347.6	60.3
I_3_12_6$_9$		83					83	0.00	315.3	60.3
I_3_12_6$_{10}$		73					67	0.00	207.7	60.3
Sc. 2 - Avg:								**0.00**	**277.7**	**60.3**
I_6_14_12$_1$	84	84	78	0.00	7.9	48.0	78	0.00	236.3	72.0
I_6_14_12$_2$	86	86	86	0.00	8.4	48.0	86	0.00	1793.2	72.0
I_6_14_12$_3$	107	107	102	0.00	12.5	58.6	102	0.00	3233.9	82.6
I_6_14_12$_4$	111	111	111	0.00	11.6	71.0	111	0.00	2131.6	95.0
I_6_14_12$_5$	90	90	84	0.00	7.8	39.2	84	0.00	239.8	63.2
I_6_14_12$_6$	96	96	96	0.00	15.8	85.6	96	0.00	998.5	109.6
I_6_14_12$_7$	103	103	98	0.00	11.2	58.6	98	0.00	3628.7	82.6
I_6_14_12$_8$	87	87	87	0.00	6.4	39.2	87	0.00	570.8	63.2
I_6_14_12$_9$	104	104	103	0.00	4.3	31.9	103	0.00	1459.2	55.9
I_6_14_12$_{10}$	98	98	96	0.00	14.6	85.6	96	0.00	1243.9	109.6
Sc. 3 - Avg:				**0.00**	**10.1**	**56.6**		**0.00**	**1553.6**	**80.6**
I_6_18_12$_1$	121		117	0.00	19.9	134.3				
I_6_18_12$_2$	132		126	0.00	15.0	98.6				
I_6_18_12$_3$	126		123	0.00	19.1	115.3				
I_6_18_12$_4$	128		128	0.00	26.4	155.8				
I_6_18_12$_5$	136		131	0.00	39.7	180.0				
I_6_18_12$_6$	119		119	0.00	28.6	155.8				
I_6_18_12$_7$	112		110	1.82	19.8	84.2				
I_6_18_12$_8$	122		117	0.00	27.0	155.8				
I_6_18_12$_9$	136		124	0.00	47.0	180.0				
I_6_18_12$_{10}$	126		125	0.00	34.9	134.3				
Scenario 4 - Avg:				**0.18**	**27.7**	**139.4**				
All inst. - Avg:				**0.06**	**12.7**	**66.9**		**0.00**	**630.0**	**42.7**
%Inst. solved to opt.:	**53%**	**40%**		**97%**				**100%**		

Table 4.9: Computational performance of column generation approach (Model (4.24)-(4.28))

	(iv) Relaxed Model 1 vs (vi) Relaxed Model 2							
	$\gamma = 3$				$\gamma = 4$			
Inst.	**LB1**	**%e$_1$**	**LB2**	**%e$_2$**	**LB1**	**%e$_1$**	**LB2**	**%e$_2$**
I_3_8_6$_1$	41	87.23	47	100.00	41	87.23	47	100.00
I_3_8_6$_2$	42	93.33	45	100.00	42	93.33	45	100.00
I_3_8_6$_3$	37	82.22	45	100.00	37	82.22	45	100.00
I_3_8_6$_4$	42	91.30	46	100.00	42	91.30	46	100.00
I_3_8_6$_5$	40	83.33	48	100.00	40	83.33	48	100.00
I_3_8_6$_6$	42	73.68	57	100.00	42	73.68	57	100.00
I_3_8_6$_7$	40	90.91	44	100.00	40	90.91	44	100.00
I_3_8_6$_8$	41	83.67	49	100.00	41	83.67	49	100.00
I_3_8_6$_9$	40	90.91	44	100.00	40	90.91	44	100.00
I_3_8_6$_{10}$	41	91.11	45	100.00	41	91.11	45	100.00
Sc. 1 - Avg:		**86.77**		**100.00**		**86.77**		**100.00**
I_3_12_6$_1$					58	80.56	72.0	100.00
I_3_12_6$_2$					56	77.78	72.0	100.00
I_3_12_6$_3$					55	69.62	79.0	100.00
I_3_12_6$_4$					57	76.00	75.0	100.00
I_3_12_6$_5$					57	74.03	77.0	100.00
I_3_12_6$_6$					60	68.18	88.0	100.00
I_3_12_6$_7$					59	83.10	71.0	100.00
I_3_12_6$_8$					57	74.03	77.0	100.00
I_3_12_6$_9$					58	69.88	83.0	100.00
I_3_12_6$_{10}$					55	82.09	67.0	100.00
Scenario 2 - Averages:						**75.53**		**100.00**
I_6_14_12$_1$	78	100.00	78	100.00	78	100.00	78	100.00
I_6_14_12$_2$	78	90.70	86	100.00	78	90.70	86	100.00
I_6_14_12$_3$	90	88.24	102	100.00	90	88.24	102	100.00
I_6_14_12$_4$	98	88.29	111	100.00	98	88.29	111	100.00
I_6_14_12$_5$	82	97.62	84	100.00	82	97.62	84	100.00
I_6_14_12$_6$	94	97.92	96	100.00	94	97.92	96	100.00
I_6_14_12$_7$	91	92.86	98	100.00	91	92.86	98	100.00
I_6_14_12$_8$	85	97.70	87	100.00	85	97.70	87	100.00
I_6_14_12$_9$	87	84.47	103	100.00	87	84.47	103	100.00
I_6_14_12$_{10}$	93	96.88	96	100.00	93	96.88	96	100.00
Sc. 3 - Avg:		**93.47**		**100.00**		**93.47**		**100.00**
I_6_18_12$_1$	103	88.03	117	100.00	103	89.57		
I_6_18_12$_2$	113	89.68	126	100.00	113	89.68		
I_6_18_12$_3$	110	89.43	123	100.00	110	89.43		
I_6_18_12$_4$	102	79.69	128	100.00	102	76.12		
I_6_18_12$_5$	108	82.44	131	100.00	108	-		
I_6_18_12$_6$	107	89.92	119	100.00	107	89.92		
I_6_18_12$_7$	104	94.55	110	98.18	104	98.11		
I_6_18_12$_8$	109	93.16	117	100.00	109	93.16		
I_6_18_12$_9$	110	88.71	124	100.00	110	79.14		
I_6_18_12$_{10}$	106	84.80	125	100.00	106	82.81		
Sc. 4 - Avg:		**88.04**		**99.82**		**87.55**		
All ins. - Avg:		**89.43**		**99.94**		**85.78**		**100.00**

Table 4.10: Computational results: Comparison of Model 4.3 and Model (4.24)-(4.28)

Chapter 5

Application: A Column Generation Heuristic for the 2D Cutting Stock Problem

Given a set of rectangular items and infinitely many identical rectangular bins, the *Two-Dimensional Cutting Stock Problem* (2DCSP) asks to cut all the items by using the minimum number of bins or, equivalently, by minimizing the global area of the used bins. We consider the Two-Dimensional *Two-Staged Guillotine Cutting Stock Problem with Multiple Stock Size* (MS2DCSP), that is, a 2DCSP where: i) All the items must be obtained through guillotine cuts, i.e., cuts that are parallel to the sides of the bin and cross the bin from one side to the other; ii) All the items must be obtained with two-staged cuts, where each stage consists in a set of parallel guillotine cuts performed on the shape obtained in the previous stage (we allow trimming, i.e., a third stage cut can be used to separate a rectangle from a waste area); and iii) Bins of different sizes are available. We consider the two cases where the items can or cannot be orthogonally rotated. In the typology introduced by Wäscher et al. [113], the problem is the constrained version of the *Two-Dimensional Rectangular Multiple Stock Size Cutting Stock Problem*.

Formally, we are given m classes of rectangular items, each item class i ($i = 1, \ldots, m$) having a demand d_i, and n classes of rectangular bins, each bin class j ($j = 1, \ldots, n$) having an infinite number of copies. Each item of class i ($i = 1, \ldots, m$) has dimensions (l_i, w_i), where l_i and w_i are, respectively, the length and the width of the item. Each bin of class j ($j = 1, \ldots, n$) has dimensions (L_j, W_j), where L_j and W_j are, respectively, the length and the width of the bin. All the input data are assumed to have integer positive values. We have to define the cutting patterns (simply denoted as patterns in the following) so as to obtain all the requested items in the specified demand, by minimizing the global area of the used bins. The problem

is NP-hard and arises in the industry whenever a sheet of chaper, wood, glass, or metal has to be cut.

2DCSP was introduced by Gilmore and Gomory [105], who as well introduced the $k-staged$ version of the problem and considered bins of different sizes. Alvarez-Valdes et al. [106] developed and compared several heuristic methods, based on column generation, for solving the general Two-Dimensional Cutting Stock Problem. For solving the subproblems, in addition to dynamic programming, they developed some heuristics. Riehme et al. [107] considered the two-staged version of the problem in the case where bins of different sizes are available and the demands of the item classes differ in a large range. Cintra et al. [108] considered several Two-Dimensional Guillotine Cutting Stock Problems and their variants in which orthogonal rotations are allowed. They presented dynamic programming algorithms for the Rectangular Knapsack Problem, which are then used to generate patterns in approaches for the Cutting Stock Problems based on Column Generation. In particular, they considered the two-dimensional two-staged guillotine version of the Cutting Stock Problem where bins of different sizes are available and rotation of the items is allowed. Their algorithm generates columns to optimally solve the continuous relaxation of the Cutting Stock Model (see Section 5.1), which is then iteratively rounded-down so as to obtain an integer feasible solution. The algorithm by Cintra et al. [108] is the only algorithm in the literature reporting detailed computational results which allow a complete comparison. A survey on several Two-Dimensional Packing Problems can be found in Lodi et al. [110].

The chaper is organized as follows: in Section 5.1 we review a classical Cutting Stock model, in Section 5.2 we describe a model and a heuristic algorithm for the generation of cutting patterns, and in Section 5.3 we present the overall Column Generation Heuristic. Computational results are presented and discussed in Section 5.4, and Section 5.5 concludes the presentation.

5.1 Cutting Stock Model

Our approach is based on column generation and follows the seminal work by Gilmore and Gomory [103, 104] for the One-Dimensional Cutting Stock Problem. Let S_j $(j = 1, \ldots, n)$ be the family of all the feasible cutting patterns for bin class j, that is, cutting patterns which do not exceed the size of bin class j and can be obtained by two-staged guillotine cuts. Following Cintra et al. [108], and without loss of generality, we only allow patterns where the first cuts are horizontal. In addition, we consider only patterns which do not contain more items of an item class i than specified by the corresponding request d_i.

MS2DCSP can be modelled by using an integer variable x_{s_j} associated with each feasible cutting pattern $s_j \in S_j$ for the bin class j $(j = 1, ..., n)$. The value of this variable denotes the number of times the pattern is used in the solution. The objective function minimizes the total area of the used bins. By introducing a parameter $C^i_{s_j}$ to represent the number of items of item class i $(i = 1, ..., m)$ in the cutting pattern s_j, and by defining $\kappa_j = L_j W_j$ as the area of a bin of bin class j, a classical model for MS2DCSP reads:

$$z := min \quad \sum_{j=1}^{n} \sum_{s_j \in S_j} \kappa_j x_{s_j} \tag{5.1}$$

$$subject \quad to \quad \sum_{j=1}^{n} \sum_{s_j \in S_j} C^i_{s_j} x_{s_j} \geq d_i \quad i = 1, \ldots, m \tag{5.2}$$

$$x_{s_j} \in \mathbb{Z}^+, \quad s_j \in S_j \qquad j = 1, \ldots, n, \tag{5.3}$$

By dropping the integrality requirement for the variables, imposed by constraints (5.3), we obtain the Linear Programming (LP) Relaxation of model (1)-(3), defines by (1), (2) and:

$$x_{s_j} \geq 0, \quad s_j \in S_j, \quad j = 1, \ldots, n \tag{5.4}$$

Model (5.1),(5.2) and (5.4), called *Master Problem* (MP), needs column generation techniques to be efficiently managed. Actually, it has exponentially many variables, which cannot be explicitly generated for large-size instances. Instead of the original Master Problem, a *Restricted Master Problem* (RMP) is considered which is initialized and solved with a subset of the exponentially many variables $x_{s_j}, s_j \in S_j$, $j = 1, \ldots, n$. In particular, RMP is initialized with the variables (columns) corresponding to a feasible solution computed through the heuristic algorithm described in Section 5.2.1. Given a solution to RMP, new variables, needed to optimally solve MP, are obtained by separating the following dual constraints:

$$\sum_{i=1}^{m} C_{s_j}^i \pi_i^* \leq \kappa_j \quad s_j \in S_j, \quad j = 1, ..., n \tag{5.5}$$

where π_i^* $(i = 1, ..., m)$ represents the optimal dual variable associated with the i-th constraint of type (5.2).

Accordingly, the so-called *Slave Problem* (SP) is to determine if, for some bin class j, a feasible pattern $s_j^* \in S_j$ exists, such that:

$$\sum_{i=1}^{m} C_{s_j^*}^i \pi_i^* > \kappa_j \tag{5.6}$$

While the satisfaction of the demand associated with the item classes is imposed in MP, the SPs (one is defined for each bin class) are devoted to the generation of feasible cutting patterns satisfying constraints (5.6). SPs (see Section 5.2.2) consist in a series of Two-Dimensional Two-Staged Knapsack Problems with Guillotine Cuts (2DKP), where one looks for the pattern of maximum profit according to the current value of the optimal dual variables (profits) π_i^* $(i = 1, \ldots, m)$. If a feasible pattern s_j^* with maximum profit greater than κ_j is found, the column corresponding to s_j^* is added to the current RMP.

When no feasible pattern with maximum profit greater than the corresponding κ_j is found, the rounded-up value of the optimal solution to MP is a valid lower bound for the original MS2DCSP.

5.2 Cutting pattern generation

A first approach to 2DKP was proposed by Gilmore and Gomory [105] , and is based on two nested unbounded Knapsack Problems (KP). This is an exact method that generates cutting patterns which can be characterized by overproduction of items. Overproduction affects the value of the optimal solution of the continuous relaxation of the model, which can produce a weaker lower bound on the optimal solution value of the problem.

The best known compact optimization model to solve 2DKP was proposed by Lodi and Monaci [109]. The idea of this model (called M2 in the following) is to define a potential strip for

each item to be cut, and to implicitly order the strips according to non-increasing widths of the associated items. A strip is initialized and can be used only if it contains the item whose width defines the strip width. This idea removes much of the symmetry which affects descriptive 2DKP models. Lodi and Monaci [109] proposed as well a model, called M1, more suited for Two-Dimensional Bin Packing Problems, where each item class contains a single item, i.e. $d_i = 1$ $(i = 1, \ldots, m)$.

5.2.1 Initial Heuristic

A heuristic algorithm for MS2DCSP is used to provide an initial feasible solution to the problem in a very short computing time. In order to take into account rotation, a copy of each item class is created by swapping length and width; then the new set of item classes is sorted according to non-increasing widths. In case more item classes have the same width, they are ordered by non increasing length. The algorithm associates each item class i with a bin class, say $b(i)$. $b(i)$ is defined as the bin class of minimum area such that a single bin of this class can fit the whole demand d_i of item class i. When no such bin class exists, $b(i)$ is defined as the bin class of minimum area such that a single bin of this class can fit the largest number of items of class i.

The items are iteratively inserted into the bins by following the predefined order. The insertion is performed by strips: each strip determines a guillotine cut in the bin; the width of the strip equals the width of the first inserted item. Strips are filled from left to right, as explained in the following. Given an item of class i that has to be cut, we initialize a new strip, which is inserted in a partially cut bin or in a new bin of class $b(i)$ if such a bin does not exist. Consequently its demand d_i and the demand of the corresponding rotated item class are updated. When the strip cannot fit any additional item of class i, a new strip is initialized in the same bin or in a new bin of class $b(i)$ and the item is inserted in the new strip. Before proceeding with the new strip, a greedy procedure looks iteratively for items of the following item classes (with smaller width) that can fill the remaining space in the first strip.If such item class exists, the maximum number of items that can be cut according to the remaining demand d_i and to the remaining space are inserted. Consequently its demand d_i and the demand of the corresponding rotated item class are updated. The greedy procedure continues to fill the strip until no item can be inserted. The new strip is placed over the first strip in the current bin whenever it is possible. Otherwise, a new bin of class $b(i)$ is initialized and the greedy procedure explained before is used to fill the empty space over the strip. When the demand of the current item class i is satisfied, the algorithm continues with the next item class, in the order previous defined, that has a remaining demand. If such item class does not exist, the algorithm terminates.

5.2.2 A Mixed Integer Linear Model for the 2DKP

Given a bin class j ($j = 1,...,n$) and a feasible dual solution π_i^* $(i = 1, \ldots, m)$ to RMP, in order to generate new columns with negative reduced cost or to prove the optimality of the current master solution, we have to solve a 2DKP. In this section we propose a Mixed Integer Linear Programming (MILP) Model, called M_0 in the following. In the computational experiments (Section 5.4), M_0 will be compared with model M_2 proposed by Lodi and Monaci in [109]. We present the case where items can be rotated, which is more general; the case of no rotation trivially follows.

For a specified bin class with dimensions (L, W), define $I = \{i : (w_i \leq W \text{ and } l_i \leq L) \text{ or } (l_i \leq W \text{ and } w_i \leq L)\}$ as the index set of the item classes which can fit in the bin, and let ϱ be the maximum number of strips which can fit in the bin. This value can be computed as the minimum between the global number of items to be cut and the maximum number of times the item with the smallest width or length can be cut from that bin:

$$\varrho = min(\sum_{i \in I} d_i, max_{i \in I}(max(\left\lfloor \frac{W}{w_i} \right\rfloor, \left\lfloor \frac{W}{l_i} \right\rfloor)) \tag{5.7}$$

We introduce three groups of variables. The first one contains two binary variables $\varphi_{i,r}$ and $\overline{\varphi_{i,r}}$, indicating whether at least one item of class i, respectively rotated item class i, $(i = 1, \ldots, m)$, is in strip r $(r = 1, \ldots, \varrho)$. The second group of variables contains two integer variables $\gamma_{i,r}$ and $\overline{\gamma_{i,r}}$, which denote the number of items of class i, respectively rotated item class i $(i = 1, \ldots, m)$, to be cut from strip r $(r = 1, \ldots, \varrho)$. The third group contains continuous variables y_r representing the width of strip r $(r = 1, \ldots, \varrho)$. In addition, we introduce two coefficients θ_{min} and θ_{max} denoting, respectively, the minimum and maximum number of items which could be cut from a strip of a bin.

$$\theta_{min} = \left\lfloor \frac{L}{max_{i \in I}\{l_i, w_i\}} \right\rfloor \tag{5.8}$$

$$\theta_{max} = \left\lfloor \frac{L}{min_{i \in I}\{l_i, w_i\}} \right\rfloor \tag{5.9}$$

A possible MILP model for 2DKP reads:

$$\tau := max \sum_{i \in I} \pi_i^* \sum_{r=1}^{\varrho} (\gamma_{i,r} + \overline{\gamma_{i,r}}) \tag{5.10}$$

$$subject \quad to \quad \sum_{i \in I} (l_i w_i)(\gamma_{i,r} + \overline{\gamma_{i,r}}) \leq y_r L \qquad r = 1, \ldots, \varrho \tag{5.11}$$

$$\sum_{i \in I} (l_i \gamma_{i,r} + w_i \overline{\gamma_{i,r}}) \leq L \qquad r = 1, \ldots, \varrho \tag{5.12}$$

$$\sum_{r=1}^{\varrho} (\gamma_{i,r} + \overline{\gamma_{i,r}}) \leq \tilde{d}_i \qquad i \in I \tag{5.13}$$

$$\sum_{r=1}^{\varrho} y_r \leq W \tag{5.14}$$

$$w_i \varphi_{i,r} \leq y_r \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.15}$$

$$l_i \overline{\varphi_{i,r}} \leq y_r \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.16}$$

$$\gamma_{i,r} \leq \hat{d}_i \varphi_{i,r} \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.17}$$

$$\overline{\gamma_{i,r}} \leq \breve{d}_i \overline{\varphi_{i,r}} \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.18}$$

$$y_r \leq y_{r+1} \qquad r = 1, \ldots, \varrho - 1 \tag{5.19}$$

$$\sum_{i \in I} \lfloor l_i / (L/(\theta+1)) - \varepsilon \rfloor \, \gamma_{i,r} \leq \theta \qquad r = 1, \ldots, \varrho,\ \theta = \theta_{min}, \ldots, \theta_{max} \tag{5.20}$$

$$\sum_{i \in I} \lfloor w_i / (L/(\theta+1)) - \varepsilon \rfloor \, \overline{\gamma_{i,r}} \leq \theta \qquad r = 1, \ldots, \varrho,\ \theta = \theta_{min}, \ldots, \theta_{max} \tag{5.21}$$

$$\gamma_{i,r} \in \mathbb{Z}^+ \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.22}$$

$$\overline{\gamma_{i,r}} \in \mathbb{Z}^+ \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.23}$$

$$\varphi_{q,r} \in \{0, 1\} \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.24}$$

$$\overline{\varphi_{q,r}} \in \{0, 1\} \qquad i \in I,\ r = 1, \ldots, \varrho \tag{5.25}$$

$$y_r \geq 0 \qquad r = 1, \ldots, \varrho \tag{5.26}$$

where, for $i \in I$:

$$\tilde{d}_i = min\{d_i, \frac{(L\ W)}{(l_i\ w_i)}\} \tag{5.27}$$

$$\hat{d}_i = min\{d_i, \lfloor \frac{L}{l_i} \rfloor\} \tag{5.28}$$

$$\breve{d}_i = min\{d_i, \lfloor \frac{L}{w_i} \rfloor\} \tag{5.29}$$

$$\tag{5.30}$$

Objective function (5.10) maximizes the profit of the selected items. Geometric constraints (5.11) ensure that the global area of the items in a strip does not exceed the area of the strip. Constraints (5.12) ensure that the length of each strip does not exceed the length of the bin. For rotated items, the length corresponds to the width. Constraints (5.13) ensure that the number of items of class i does not exceed the maximum number $\tilde{d}_i$ of items in the bin. The width feasibility of the bin is provided by constraint (5.14). Constraints (5.15)

and (5.16) ensure that the width y_r of strip r is not smaller than the width of any item and the length of any rotated item, respectively, in the strip. Constraints (5.17) and (5.18) impose the activation of binary variables $\varphi_{i,r}$ and $\overline{\varphi_{i,r}}$ whenever at least one item of class i and, respectively, of its corresponding rotated class, is used in strip r. Parameters $\hat{d}_i$ and $\breve{d}_i$ denote the maximum number of items of class i, and, respectively, of its corresponding rotated class, in the strip. To reduce the symmetry of the model, constraints (5.19) impose that strips have non decreasing widths. In addition, constraints (5.20) and (5.21) impose a maximum number of different item classes simultaneously present in a strip according to its length. These constraints are an extension of the dual feasible functions used by Fekete and Schepers [100] to obtain a lower bound for the Bin Packing Problem. Parameter ε is a very small number, used to decrease of one unit the value associated with item classes which are inserted an integer number of times into the bin. In particular, these constraints impose some simultaneous insertion incompatibilities between item classes present in a strip r, and eliminate some non integer feasible solutions. For example, when θ is 1, if there are item classes whose length is larger than half of the length of the bin, only one item of those item classes can be cut in a single strip.

5.2.3 Heuristic Pattern Generation Scheme

In this section we discuss a heuristic procedure for solving 2DKP, i.e. for generating patterns, without resorting to the solution of a MILP, which can be a time consuming task. The procedure, which does not guarantee optimality of the solution, is based on the solution of two nested (NP-Hard) Bounded Knapsack Problems (BKP). The inner one combines items together into feasible strips, while the outer one deals with groups of strips. Let define ω as the number of possible different strip widths in a bin with dimensions (L, W). This coefficient is equal to the number of different item widths (and lengths, when rotation is allowed) which are not larger than the bin width W. For each strip r ($r = 1, ..., \omega$) let w_r be the width of the strip and define G_r as the set which includes the item classes $i \in I$ with $w_i \leq w_r$ (or $l_i \leq w_r$ for rotation) and $l_i \leq L$ (or $w_i \leq L$ for rotation). Using variables $\gamma_{i,r}$ and $\overline{\gamma_{i,r}}$ defined in Section 5.2.2, the inner BKP, which maximizes the profit of the items cut in a strip r for the given bin, reads:

$$\Pi_r := max \quad \sum_{i \in G_r} \pi_i^* (\gamma_{i,r} + \overline{\gamma_{i,r}}) \tag{5.31}$$

$$subject \quad to \quad \sum_{i \in G_r} (l_i \gamma_{i,r} + w_i \overline{\gamma_{i,r}}) \leq L \tag{5.32}$$

$$0 \leq \gamma_{i,r} \leq \hat{d}_i, \quad \gamma_{i,r} \in \mathbb{Z}^+ \quad i \in G_r \tag{5.33}$$

$$0 \leq \overline{\gamma_{i,r}} \leq \breve{d}_i, \quad \overline{\gamma_{i,r}} \in \mathbb{Z}^+ \quad i \in G_r \tag{5.34}$$

Given the optimal solution $\gamma_{i,r}^*$ and $\overline{\gamma_{i,r}^*}$ of model (5.31)-(5.34), its corresponding value Π_r^* is set as profit of strip r in the outer BKP, and $\gamma_{i,r}^*$, $\overline{\gamma_{i,r}^*}$ are interpreted as parameters. We introduce an integer variable β_r, which represents the number of strips of type r (i.e. having width equal to w_r) in the cutting pattern. The outer BKP, for the given bin, reads:

$$\delta := max \quad \sum_{r=1}^{\omega} \Pi_r^* \beta_r \tag{5.35}$$

$$subject \quad to \quad \sum_{r=1}^{\omega} w_r \beta_r \leq W \tag{5.36}$$

$$0 \leq \beta_r \leq min(\lfloor W/\omega_r \rfloor, min_{i \in G_r}(\left\lfloor \tilde{d}_i/(\gamma_{i,r} + \overline{\gamma_{i,r}}) \right\rfloor)), \quad \beta_r \in \mathbb{Z}^+ \quad r = 1, ...\omega \tag{5.37}$$

Optimally solving in sequence the two BKPs, clearly does not ensure the global optimality, because solving to optimum the inner BKP may lead to sub optimal solutions for the outer one. In addition, some patterns could be characterized by a number of items of class i greater than the corresponding demand d_i. This fact deteriorates the lower bound provided by the optimal solution of MP. We face this problem by introducing a further phase for removing the exceeding item quantities from the cutting patterns once the procedure is concluded.

Both BKP models are solved using the dynamic programming scheme presented in Algorithm 4, where $\overline{c}$ represents the capacity of the knapsack, $\overline{n}$ the number of items, $\overline{p}_j$, $\overline{w}_j$ and $\overline{d}_i$, respectively, the profit, the weight and the demand of item j ($j = 1, \ldots, \overline{n}$). As data structures we use a vector f of size $\overline{c}+1$, whose component $f_{[s]}$ denotes the optimal profit corresponding to the knapsack problem with capacity s; and a matrix A of size $(\overline{c}+1) \times \overline{n}$, whose element $a_{[s][j]}$ denotes the number of copies of item j corresponding to capacity s. The algorithm complexity is $O \sum_{j=1,\ldots,\overline{n}}(\overline{d_j}))$. The algorithm is an extension of the dynamic programming approach for the 0-1 Knapsack Problem (see Martello and Toth [50]).

5.3 Column Generation Heuristic

To solve to optimality MP given by model (5.1), (5.2) and (5.4), a classical column generation scheme is applied. In detail, RMP is initialized with the columns corresponding to a heuristic solution. Then the current RMP is solved to optimality, thus obtaining a vector of dual variables π^*. A *slave problem* SP is defined for every bin class j ($j = 1, \ldots, n$). The SPs are heuristically solved with the heuristic pattern generation scheme (see Section 5.2.3) and new columns with negative reduced cost are added to the current RMP. If no column with negative reduced cost is found, the SPs are tackled with the exact model described in Section 5.2.2, which can find columns to be added to the current RMP or prove the optimality of the current fractional solution. In detail, we solve for each bin class the 2DKP exact model, using the dual variables π^* as profits; if the solution of the model has, for at least one bin class, negative reduced cost the corresponding column is added to the current RMP and RMP is re-optimized. Otherwise, i.e. if no negative reduced cost column is found, the current (possibly fractional) solution of RMP is optimal for the MP.

In order to obtain an integer solution, we tested three different heuristic strategies:

1. The first strategy consists in applying an Integer Linear Programming (ILP) solver (namely, CPLEX 12.1 [116]) to model (5.1)–(5.3), by considering only the variables used to optimally solve its continuous relaxation, i.e., the MP (5.1), (5.2) and (5.4). This strategy takes full advantage of the advanced routines embedded in the ILP solver.

2. The second strategy is a so-called *diving* heuristic, and consists in iteratively solving MP by generating columns, and then rounding-up the lower bound of a fractional variable to the closest integer, until an integer solution is obtained. In detail, after having solved to optimality MP, at each iteration we set the lower bound of every integer variable to the current value of the variable, and for the largest fractional variable, we set the lower bound to its rounded-up value. Then we call the column generation scheme to solve the current MP with the modified lower bounds associated with the variables. Note that MP is optimally solved only at the root node, so as to obtain a valid lower bound. At the

Algorithm 4 KP-Bounded Dynamic Programming

Input:
$\overline{c}$, $\overline{n}$, ($\overline{p_j}$, $\overline{w_j}$, $\overline{d_j}$) for j = 1,...,$\overline{n}$
Data structures:
A: $(\overline{c}+1) \times \overline{n}$ integer matrix
f: $\overline{c}+1$ integer vector
Output:
$f_{[\overline{c}]}$: optimal profit;
x_j: copies of item j in the optimal solution;

$f_{[s]} = 0$ $s = 0, 1, \ldots, \overline{c}$
$a_{[s][j]} = 0$ $s = 0, 1, \ldots, \overline{c}$ $j = 1, \ldots, \overline{n}$

for $j = 1, \ldots, \overline{n}$ **do**
  for $s = \overline{c}, \ldots, 1$ **do**
    for $i = 1, \ldots, \overline{d_j}$ **do**
      if $(s - (i \cdot w_j) \geq 0)$ **then**
        if $f_{[s-i \cdot w_j]} + i \cdot p_j > f_{[s]}$ **then**
          $f_{[s]} = f_{[s-i \cdot w_j]} + i \cdot p_j$;
          $a_{[s][j]} = i$;
        end if
      else
        break;
      end if
    end for
  end for
end for
$s = \overline{c}$;
for $j = \overline{n}, \ldots, 1$ **do**
  $x_{[j]} = a_{[s][j]}$;
  $s = s - w_j \cdot x_{[j]}$;
end for

instance	m	*total*
gcut1d	10	669
gcut2d	20	982
gcut3d	30	1489
gcut4d	50	2751
gcut5d	10	645
gcut6d	20	1064
gcut7d	30	1626
gcut8d	50	2363
gcut9d	10	590
gcut10d	20	830
gcut11d	30	1298
gcut12d	50	2081

Table 5.1: Instances

subsequent iterations, the columns are generated by applying the heuristic procedure described in Section 5.2.3. This method is equivalent to a Branch-and-Price algorithm, based on a depth-first strategy, and stopped when the first integer solution is found. The advantage of this method is that the column generation is not stopped at the root node of the Branch-and-Bound tree.

3. The third strategy combines the previous two: we execute the diving heuristic, and then we apply the ILP solver to problem (5.1)–(5.3) where we consider all the variables generated during the diving heuristic.

A time limit of 15 seconds is imposed when solving problem (5.1)–(5.3) in strategies 1 and 3.

5.4 Computational Experiments

All the algorithms described in the previous sections where implemented in C and run on one core of an Intel Core 2 Quad 2.50 GHz, with 7.7 GB shared memory, under Linux Ubuntu 9.04 operating system. All the Linear Programs and the Mixed-Integer Linear Programs where solved with CPLEX 12.1 [116].

We considered the 12 instances proposed by Cintra et al. [108] and publicly available from the web [**?**]. These instances are obtained by adding a random integer demand in the range $[1, \ldots, 100]$ to the 2DKP instances $gcut1, \ldots, gcut12$ from the OR library [114], and by defining three bin classes for each instance: the first bin class is the original one of dimensions (L, W), the other two classes have dimensions $(1.2L, 0.8W)$ and $(1.1L, 0.9W)$, respectively. The features of the instances, which are called $gcut1d, \ldots, gcut12d$, are summarized in Table 5.1, where the number m of item classes and the *total* number of items are reported. For every instance, we consider the case where the items can be rotated and the case where rotation is forbidden. In the following tables, all the computing times are expressed in seconds.

5.4.1 Performance of the MILP Model M_0 for 2DKP

In this section we report the results obtained by solving the 2DKP model M_0 (see Section 5.2.2), and compare it with model M_2 proposed by Lodi and Monaci [109]. We consider in this section as profits associated with the items the values of the corresponding dual variables in RMP. Actually, the aim of this computational comparison is to show that it is better to use M_0 instead of M_2 when one has to solve to optimality SP.

In Tables 5.2 and 5.3, we consider instances $gcut1d, \ldots, gcut12d$, in the cases without and with rotation, respectively. We set as profit associated with each item class i the value of the corresponding dual variable π_i^* at the first iteration of the column generation procedure. The tables report the instance name in the first column. For every instance we consider 3 bin classes, thus we have 3 rows per instance. The value of the objective function (τ) is reported in the second column. The third and fourth columns of the tables report the value of the continuous relaxation of model M_0, and the corresponding computing time. Column 5 reports the time needed to solve model M_0 to optimality, and column 6 the number of nodes explored by CPLEX 12.1 [116]. The corresponding information for model M_2 is reported in columns 7 to 10 of the tables. The last row of the tables reports average values.

Considering the computing times required to solve models M_0 and M_2 to optimality, Table 5.3 shows that it is better to use M_0 for solving the SP when rotation is allowed, while Table 5.2 shows that the two models are equivalent in terms of computing time when rotation is forbidden. On one hand, this is somehow surprising, since the number of nodes explored by CPLEX 12.1 [116] to solve M_2 is on average one order of magnitude smaller than the number of nodes explored when solving M_0. On the other hand, the time needed by CPLEX 12.1 [116] to solve the continuous relaxation of M_0 is much smaller than the corresponding computing time required by M_2. Concerning the upper bounds provided by the continuous relaxation of the models, M_0 provides a better upper bound in 24 cases over 36 when rotation is allowed, while it provides a weaker upper bound in all cases but 1 when rotation is not allowed.

5.4.2 Performance of the Overall Method

In this section we report the results obtained by the overall method, which is compared with the heuristic algorithm proposed by Cintra et al. [108] on the $gcut1d, \ldots, gcut12d$ instances. The latter algorithm is based on column generation to solve MP, and columns are generated through dynamic programming. Given a fractional solution to MP, an iterative reduction of the problem size through rounding-down and fixing of fractional variables is implemented. At each iteration, the current fractional solution is rounded-down and the corresponding items are removed from the problem. The residual problem is then solved again through column generation, and the procedure is iterated until an integer solution is produced.

The results of Cintra et al. were obtained on a PIV at 1.8 GHz with 512 MB under Linux operating system; CPL (COIN-OR LP Solver) [117] was used as LP solver. According to our computational experience, this computing system is about 3 times slower than that used in our experiments. Tables 5.4 and 5.5 report results from [108], for the cases without and with rotation, respectively. The first column of the tables reports the name of the instance, followed by the lower bound (LB) in column 2, the upper bound (UB) (value of the integer solution) in column 3, the global number (Col) of generated columns in column 4, the percentage optimality gap (GAP) in column 5 and the computing time (T) in column 6. The last line of the tables reports average values.

instance	τ	LP_{M0}	T_{LP}	T_{MIP}	*nodes*	LP_{M2}	T_{LP}	T_{MIP}	*nodes*
gcut1d	90000	154866.00	0.00	0.01	5	159596.00	0.00	0.01	5
	105000	149061.00	0.00	0.00	0	158520.00	0.00	0.01	0
	120000	153716.00	0.00	0.00	0	155293.00	0.00	0.01	0
gcut2d	96145.8	117463.00	0.00	0.02	14	117373.00	0.00	0.01	0
	86041.7	116248.00	0.00	0.02	97	116203.00	0.00	0.01	11
	72604.2	112353.00	0.00	0.01	106	112692.00	0.01	0.02	19
gcut3d	90937.5	118319.00	0.00	0.04	165	117721.00	0.01	0.03	17
	92777.8	116869.00	0.00	0.03	143	116567.00	0.01	0.03	5
	73506.9	113112.00	0.01	0.04	239	113106.00	0.01	0.03	47
gcut4d	174688	249804.00	0.01	0.01	19	264128.00	0.03	0.04	0
	180000	263834.00	0.00	0.02	55	262089.00	0.02	0.03	0
	180000	250830.00	0.00	0.02	8	255972.00	0.02	0.02	0
gcut5d	288750	376889.00	0.00	0.00	27	403971.00	0.01	0.01	8
	311250	366406.00	0.00	0.01	0	387834.00	0.00	0.00	0
	332083	391372.00	0.00	0.01	32	399937.00	0.00	0.01	9
gcut6d	330833	481832.00	0.00	0.02	96	484501.00	0.00	0.00	0
	367500	465726.00	0.00	0.01	5	466860.00	0.01	0.01	0
	350417	475803.00	0.00	0.02	60	480453.00	0.00	0.01	0
gcut7d	496250	800509.00	0.00	0.01	0	806118.00	0.01	0.01	9
	495000	785109.00	0.01	0.01	5	780331.00	0.01	0.01	0
	495000	778480.00	0.00	0.01	8	799671.00	0.01	0.01	0
gcut8d	353542	439080.00	0.00	0.08	463	439032.00	0.02	0.04	40
	311736	421248.00	0.00	0.07	196	421863.00	0.02	0.07	48
	364861	434538.00	0.00	0.05	155	434821.00	0.02	0.03	0
gcut9d	1320000	1762300.00	0.00	0.01	8	1747920.00	0.00	0.00	0
	1155000	1666100.00	0.00	0.01	9	1683860.00	0.00	0.01	0
	1245000	1730220.00	0.01	0.00	5	1731900.00	0.01	0.00	0
gcut10d	1980000	2368450.00	0.00	0.01	7	2345850.00	0.00	0.00	0
	1651670	2235250.00	0.00	0.02	33	2260730.00	0.00	0.00	0
	1490000	2323130.00	0.00	0.01	27	2324570.00	0.01	0.01	0
gcut11d	1591250	1801060.00	0.00	0.04	43	1854820.00	0.01	0.03	16
	1529170	1677110.00	0.00	0.01	25	1782170.00	0.01	0.04	30
	1490000	1821250.00	0.00	0.03	95	1836660.00	0.01	0.06	34
gcut12d	1570000	2189500.00	0.00	0.03	182	2178380.00	0.02	0.03	15
	1485000	2085060.00	0.00	0.01	14	2095550.00	0.02	0.02	0
	1320000	2167580.00	0.00	0.03	114	2158570.00	0.02	0.02	0
averages	657944.83	887791.03	0.00	0.02	68.33	895989.78	0.01	0.02	8.69

Table 5.2: Comparison of models M_0 and M_2 with dual variables as profits. Rotation is not allowed.

instance	τ	LP_{M0}	T_{LP}	T_{MIP}	*nodes*	LP_{M2}	T_{LP}	T_{MIP}	*nodes*
gcut1d	81875	109259.00	0.00	0.02	28	105692.00	0.01	0.01	8
	81875	108167.00	0.00	0.01	15	104790.00	0.01	0.01	0
	75625	104889.00	0.00	0.02	70	102082.00	0.01	0.01	0
gcut2d	101667	132640.00	0.01	0.21	523	124338.00	0.02	0.03	0
	101458	131314.00	0.02	0.2	389	123164.00	0.02	0.03	24
	91145.8	127334.00	0.01	0.12	161	119639.00	0.02	0.03	0
gcut3d	86562.5	105961.00	0.00	0.13	505	105478.00	0.06	0.12	21
	85000	104901.00	0.00	0.1	404	104504.00	0.07	0.19	80
	85000	101723.00	0.00	0.07	168	101583.00	0.06	0.11	3
gcut4d	91666.7	113760.00	0.00	0.36	1265	113539.00	0.10	0.29	102
	95000	112623.00	0.00	0.13	419	112412.00	0.11	0.38	0
	90000	109210.00	0.01	0.11	370	109031.00	0.09	0.35	0
gcut5d	333333	403418.00	0.00	0.04	131	399541.00	0.01	0.02	0
	311875	387282.00	0.00	0.03	129	384297.00	0.01	0.02	17
	338125	399384.00	0.00	0.03	51	395730.00	0.01	0.02	13
gcut6d	355000	407714.00	0.00	0.05	151	406810.00	0.02	0.03	0
	317500	391405.00	0.00	0.06	326	390868.00	0.02	0.05	34
	330000	403636.00	0.00	0.05	189	402824.00	0.02	0.04	2
gcut7d	370625	552179.00	0.00	0.05	96	551599.00	0.04	0.04	0
	371250	530092.00	0.00	0.02	64	529736.00	0.04	0.04	0
	371875	546657.00	0.00	0.05	157	546134.00	0.03	0.06	26
gcut8d	506250	580223.00	0.00	0.03	27	573629.00	0.13	0.33	7
	416250	557014.00	0.00	0.12	219	552216.00	0.12	0.35	52
	403681	574421.00	0.00	0.19	575	568275.00	0.13	0.44	29
gcut9d	1490000	1776660.00	0.00	0.02	32	1747920.00	0.01	0.02	0
	1320000	1705590.00	0.00	0.01	19	1683860.00	0.01	0.01	6
	1455000	1758890.00	0.00	0.02	35	1731900.00	0.01	0.02	7
gcut10d	1980000	2391060.00	0.00	0.03	129	2345850.00	0.02	0.02	8
	1635000	2295420.00	0.00	0.04	164	2260730.00	0.02	0.05	53
	1490000	2367150.00	0.00	0.05	262	2324570.00	0.02	0.03	23
gcut11d	1732500	2320330.00	0.00	0.08	257	2351370.00	0.07	0.14	79
	1980000	2317050.00	0.00	0.01	4	2287110.00	0.07	0.12	0
	1980000	2389460.00	0.00	0.04	126	2335300.00	0.07	0.18	67
gcut12d	1505000	2006860.00	0.01	0.33	793	2004540.00	0.08	0.12	81
	1485000	1926590.00	0.01	0.08	295	1888480.00	0.08	0.16	31
	1490000	1986790.00	0.00	0.17	598	1986210.00	0.08	0.18	51
averages	695420.53	898251.56	0.00	0.09	254.06	888215.31	0.05	0.11	22.89

Table 5.3: Comparison of models M_0 and M_2 with dual variables as profits. Rotation is allowed.

instance	*LB*	*UB*	*Col*	*Gap*	*T*
gcut1d	14822813	14880000	397	0.386	0.58
gcut2d	16740782	16820625	492	0.477	1.31
gcut3d	20149804	20267500	7877	0.584	21.83
gcut4d	46523512	46591875	11,569	0.147	60.56
gcut5d	41667500	42022500	110	0.852	0.17
gcut6d	77621563	78167500	539	0.703	0.96
gcut7d	123946563	124257500	1316	0.251	2.9
gcut8d	161074885	161575000	3958	0.310	23.67
gcut9d	130802500	131830000	86	0.786	0.12
gcut10d	260444167	262470000	434	0.778	0.81
gcut11d	303137517	304440000	6926	0.430	18.58
gcut12d	609519417	611230000	5452	0.281	36.65
averages	150537585.3	151212708.3	3263.0	0.498678196	14.012

Table 5.4: Results from Cintra et al. [108], case with no rotation.

instance	*LB*	*UB*	*Col*	*Gap*	*T*
gcut1d	13828125	13908750	416	0.583	0.67
gcut2d	15432372	15474375	4616	0.272	37.05
gcut3d	19310806	19436875	12159	0.653	45.33
gcut4d	44767393	44905000	21902	0.307	166.68
gcut5d	40087188	40382500	341	0.737	0.74
gcut6d	70839625	71162500	2411	0.456	5.49
gcut7d	114817717	115312500	13326	0.431	56.78
gcut8d	152634893	153410000	28128	0.508	394.05
gcut9d	119568000	121040000	756	1.231	1.14
gcut10d	247872858	249260000	1545	0.560	5.68
gcut11d	286973907	289430000	23447	0.856	290.64
gcut12d	562898802	564650000	28565	0.311	690.59
averages	140752640.5	141531041.7	11467.66667	0.575361116	141.237

Table 5.5: Results from Cintra et al. [108], case with rotation.

The results of the algorithms proposed in Section 5.3 are reported in Tables 5.6 and 5.7, for the case without and with rotation, respectively. The first column of the tables reports the name of the instance, followed by the solution value (UB) and the number of columns (Col) of the Initial Heuristic. Columns 4 and 5 report the computing time (T) and the number of columns (Col) generated to solve to optimality the Master Problem (5.1), (5.2) and (5.4). Columns 6, 7 and 8 report the solution value (UB), the percentage optimality gap (Gap) and the computing time (T) of the first strategy. Columns 9, 10, 11 and 12 report the solution value (UB), the percentage optimality gap (Gap), the number of columns (Col) and the computing time (T) of the diving procedure. Columns 13, 14 and 15 report the solution value (UB), the percentage optimality gap (Gap) and the computing time (T) of the third strategy. The last line of the tables reports average values.

By considering the three heuristic strategies described in Section 5.3, the tables show that the best performing one is strategy 3, which consists in considering the columns needed to optimally solve MP defined by (5.1), (5.2) and (5.4), and the columns generated by using strategy 2 (diving), and then solving the MP integer counterpart (5.1)–(5.3). The average percentage optimality gap is 0.2083% in the case without rotation and 0.0911% in the case with rotation. The diving procedure (strategy 2) obtains larger percentage gaps: 0.8976% and 0.8537% in the cases without and with rotation, respectively. However, this procedure in principle does not require the use of a sophisticated ILP solver as CPLEX 12.1 [116], and has very short computing times. Finally, strategy 1, which optimally solves model (5.1)–(5.3) by conidereing only the columns required for the optimal solution of MP, produces just slightly worse gaps than those obtained by strategy 3.

By considering the results reported by Cintra et al. [108] (see Tables 5.4 and 5.5), with average percentage gaps of 0.50% and 0.58% in the cases without and with rotation, respectively, we observe that the corresponding solution values are always worse than those obtained by procedures 1 and 3 proposed in this chaper. Considering that we use an approximately three times faster computer, our computational effort is almost equivalent in the case with no rotation, and smaller in the case with rotation. In particular, we observe that our methods are able to optimally solve MP and to produce final improved solutions by generating a much smaller number of columns.

instance	*Heur.Ini.*		*Root*		*Heur*1			*Heur*2				*Heur*3		
	UB	*Col*	*T*	*Col*	*UB*	*Gap*	*T*	*UB*	*Gap*	*Col*	*T*	*UB*	*Gap*	*T*
gcut1d	16067500	16	0.03	27	14875000	0.352	0.04	14877500	0.369	28	0.04	14875000	0.352	0.31
gcut2d	19651875	32	0.18	64	16775625	0.208	0.25	16876875	0.813	66	0.22	16775625	0.208	0.47
gcut3d	25089375	44	0.74	113	20177500	0.137	1.52	20371250	1.099	113	0.91	20177500	0.137	1.60
gcut4d	56376875	72	3.24	224	46567500	0.095	18.50	46651875	0.276	228	4.03	46566875	0.093	19.13
gcut5d	48302500	16	0.05	28	41705000	0.090	0.06	42007500	0.816	28	0.06	41705000	0.090	0.07
gcut6d	86242500	27	0.17	51	77810000	0.243	15.68	78365000	0.958	52	0.23	77812500	0.246	15.52
gcut7d	146662500	44	0.73	97	124182500	0.190	16.22	124490000	0.438	104	0.96	124172500	0.182	16.35
gcut8d	181427500	74	5.19	191	161102500	0.017	7.09	161810000	0.456	192	5.72	161102500	0.017	10.03
gcut9d	149970000	16	0.09	25	131710000	0.694	0.92	133790000	2.284	26	0.15	131710000	0.694	1.15
gcut10d	311320000	30	0.52	69	261220000	0.298	3.12	263370000	1.123	72	0.70	261220000	0.298	15.96
gcut11d	348380000	46	2.35	106	303510000	0.123	3.17	307350000	1.390	107	3.10	303510000	0.123	4.43
gcut12d	737760000	69	7.15	186	609880000	0.059	7.46	614080000	0.748	192	8.68	609880000	0.059	9.87
averages	177270885.4	40.5	1.703	98.4	150792968.8	0.2088	6.169	152003333.3	0.8976	100.7	2.067	150792291.7	0.2083	7.907

Table 5.6: Results of the proposed algorithms, case with no rotation

instance	*Heur.Ini.*		*Root*		*Heur*1			*Heur*2				*Heur*3		
	UB	*Col*	*T*	*Col*	*UB*	*Gap*	*T*	*UB*	*Gap*	*Col*	*T*	*UB*	*Gap*	*T*
gcut1d	15407500	16	0.10	30	13845625	0.127	0.10	13963750	0.981	30	0.12	13845625	0.127	0.12
gcut2d	18620625	33	2.97	95	15443125	0.070	18.27	15531250	0.641	98	3.12	15443125	0.070	18.36
gcut3d	23811875	45	3.29	129	19335000	0.125	18.57	19609375	1.546	133	3.89	19343750	0.171	19.07
gcut4d	55079375	73	17.63	239	44775000	0.017	18.03	44901875	0.300	245	20.88	44775000	0.017	22.50
gcut5d	47992500	16	0.28	32	40122500	0.088	0.30	40372500	0.712	34	0.36	40112500	0.063	0.37
gcut6d	82945000	31	0.88	71	70915000	0.106	1.32	71387500	0.773	75	1.24	70915000	0.106	1.56
gcut7d	146717500	45	4.49	131	114895000	0.067	5.89	115295000	0.416	137	6.05	114890000	0.063	7.44
gcut8d	180247500	72	28.29	231	152730000	0.062	43.47	153605000	0.636	232	35.52	152740000	0.069	50.63
gcut9d	149970000	16	0.28	30	119930000	0.303	0.33	121020000	1.214	32	0.42	119870000	0.253	0.49
gcut10d	300650000	31	2.44	82	248020000	0.059	2.49	250100000	0.899	83	3.28	248020000	0.059	3.34
gcut11d	351780000	41	15.91	141	287060000	0.030	31.15	291320000	1.514	143	18.05	287050000	0.027	33.23
gcut12d	668570000	69	37.40	214	563340000	0.078	52.63	566350000	0.613	217	46.37	563290000	0.069	61.52
averages	170149322.9	40.7	9.497	118.8	140867604.2	0.0944	16.046	141954687.5	0.8537	121.6	11.608	140857916.7	0.0911	18.219

Table 5.7: Results of the proposed algorithms, case with rotation

5.5 Conclusions

We considered a Two-Dimensional Cutting Stock Problem where stock of different sizes is available, and a set of rectangular items has to be obtained through two-staged guillotine cuts. This NP-hard problem arises in the industry whenever a sheet of chaper, wood, glass, or metal has to be cut.

Starting from the classical cutting stock model of Gilmore and Gomory [105], we proposed a column generation based heuristic algorithm, and tested three different strategies to obtain integer solutions. The column generation scheme require as subproblem the solution of a Two-Dimensional Knapsack Problem with two-staged guillotine cuts. For this Knapsack Problem, we proposed a heuristic algorithm based on dynamic programming, as well as an exact Mixed Integer Linear Programming (MILP) model.

Computational experiments were performed on a set of instances from the literature to evaluate the performance of the proposed algorithms. The MILP model was compared with a well-known compact model from the literature, and the results showed that it behaves very well when used to solve the subproblems which arise during the column generation phase.

The overall heuristic method obtains very small percentage optimality gaps and outperforms the most effective algorithm from the literature, producing improved solutions in comparable computing times.

List of Figures

List of Tables

Bibliography

[1] C.Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance : *Branch-and-Price: Column Generation for Huge Integer Programs* Operations Research, 1998, 316–329

[2] K. Aardal and A. K. Lenstra : *Hard equality constrained integer knapsacks* Mathematics of Operations Research, 2004, 724-738

[3] E. Balas : *Disjunctive programming: properties of the convex hull of feasible points* Discrete Applied Mathematics, 1998, 1-44

[4] A. Balakrishnan, T.L. Magnanti and R.T. Wong : *A dual ascent procedure for large-scale uncapacitated network design* Operations Research , 1989, 716-740

[5] E. Balas and W. Pulleyblank : *The perfectly matchable subgraph polytope of a bipartite graph* Networks, 1983, 495516

[6] F. Barahona and R. Anbil : *The algorithm: Producing primal solutions with a subgradient method* Mathematical Programming, 2000, 385-399

[7] J.J. Bartholdi, J.B. Orlin and H. Ratliff : *Cyclic scheduling via integer programs with circular ones* Operations Research, 1980, 1074-1085

[8] J.F. Benders : *Partitioning procedures for solving mixed variables programming problems* Numerische Mathematik, 1962, 238-252

[9] H. Ben Amor, J. Desrosiers and A. Frangioni : *On the choice of explicit stabilizing terms in column generation* Discrete Applied Mathematics (to appear), 2008

[10] O. Bilde and J. Krarup : *Sharp lower bounds and efficient algorithms for the simple plant location problem* Annals of Discrete Mathematics, 1977, 79-97

[11] G.H. Bradley, P.L. Hammer and L.A. Wolsey : *Coefficent reduction for inequalities in 0-1 variables* Mathematical Programming, 1974, 263-282

[12] O. Briant, C. Lemarechal, Ph. Meurdesoif, S. Michel, N. Perrot and F. Vanderbeck : *Comparison of bundle and classical column generation* Mathematical Programming, 2008, 299-344

[13] M.Conforti, M. Di Summa, F. Eisenbrand and L. A. Wolsey : *Network formulations of mixed integer programs* Mathematics of Operations Research (to appear)

[14] M. Conforti and L. A. Wolsey : *Compact formulations as unions of polyhedra* Mathematical Programming, 2008, 277-289

[15] G.B. Dantzig and P. Wolfe : *Decomposition principle for linear programs* Operations Research, 1960, 101-111

[16] J.V. de Carvalho : *Exact solution of bin packing problems using column generation and branch-andbound* Ann. Oper. Res, 1999, 629-659

[17] J.V. de Carvalho : *Using Extra Dual Cuts to Accelerate Column Generation* on Computing, 2003, 175-182

[18] J. Desrosiers and F. Soumis : *A column generation approach to the urban transit crew scheduling problem* Transportation Science, 1989, 1-13

[19] J. Desrosiers, F. Soumis and M. Desrochers : *Routing with time windows by column generation* Networks, 1984, 545-565

[20] G. Desaulniers, J. Desrosiers, M.M. Solomon : *Column Generation* Springer, Berlin, 2005

[21] O. du Merle, D. Villeneuve, J. Desrosiers and P. Hansen : *Stabilized column generation* Discrete Math., 1999, 229-237

[22] B.P. Dzielinski and R.E. Gomory : *Optimal programming of lot sizes* Operations Research, 1965, 874-890

[23] I. Elhallaoui, D. Villeneuve, F. Soumis and G. Desaulniers : *Dynamic aggregation of set-partitioning constraints in column generation* Operations Research, 2005, 632-645

[24] G. Eppen and R.K. Martin : *Solving multi-item capacitated lot-sizing problems using variable redefinition* Operations Research, 1992, 832-848

[25] D. Erlenkotter : *A dual-based procedure for uncapacitated facility location* Operations Research, 1978, 992-1009

[26] H. Everett III : *Generalized Lagrange multiplier method for solving problems of optimal allocation of resources* Operations Research, 1963, 399-417

[27] M.L. Fisher : *The Lagrangean relaxation method for solving integer programming problems* Management Science, 1981, 1-18

[28] R. Fukasawa, H. Longo, J. Lysgaard, M.P. Aragao, M. Reis, E. Uchoa and R.F. Werneck : *Robust branchand-cut-and-price for the capacitated routing problem* Mathematical Programming, 2006, 491-512

[29] P.C. Gilmore and R.E. Gomory : *A linear programming approach to the cutting stock problem* Operations Research, 1961, 849-859

[30] P.C. Gilmore and R.E. Gomory : *A linear programming approach to the cutting stock problem: Part II* Operations Research, 1963, 863-888

[31] R. Jans and Z. Degraeve : *Improved lower bounds for the capacitated lot sizing problem with set-up times* Operations Research Letters, 2004, 185-195

[32] R. Jans and Z. Degraeve : *Optimal Integer Solutions to Industrial Cutting-Stock Problems: Part 2,Benchmark Results* on Computing, 2003, 58-81

[33] R. Jans and Z. Degraeve : *A New Dantzig-Wolfe Reformulation and Branch-and-Price Algorithm for the Capacitated Lot-Sizing Problem with Setup Times* Operations Research, 2007, 909-921

[34] Z. Degraeve : *Scheduling joint product operations with proposal generation methods* on Computing, 1992, The University of Chicago, Graduate School of Business, Chicago, IL., Ph.D. Thesis

[35] Z. Degraeve, L. Schrage : *Scheduling joint product operations with proposal generation methods* on Computing, 406-419 1999

[36] H.W. Lenstra, Jr. : *Integer programming with a fixed of variables* Mathematics of Operations Research, 1983, 538-547

[37] E. Uchoa, R. Fukasawa, J. Lysgaard, A. Pessoa, M.P. Aragao and D. Andrade : *Robust branch-and-cut-and-price for the capacitated minimum spanning tree problem over an extended formulation* Mathematical Programming, 2008, 443-472

[38] Vanderbeck, F. : *Computational study of a column generation algorithm for bin packing and cutting stock problems* Mathematical Programming, 86, 565-594, 1999

[39] F. Vanderbeck : *On Dantzig-Wolfe decomposition in integer programming and ways to perform branchingin a branch-and-price algorithm* Operations Research, 2000, 111-128

[40] F. Vanderbeck and L. A. Wolsey : *An exact algorithm for IP column generation* Operations Research Letters, 1996, 151-159

[41] Vanderbeck, F. and L.A. Wolsey : *Reformulation and Decomposition of Integer Programs* 50 s of Integer Programming 1958-2008 From the Early s to the State-of-the-Art, Springer, Berlin Heidelberg, 2010, 431–502

[42] F. Vanderbeck : *BaPCod - a generic branch-and-price code* https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/BaPCod, 2005

[43] F. Vanderbeck and M. Savelsbergh : *A generic view of Dantzig-Wolfe decomposition in mixed integer programming* Operations Research Letters, 2006, 296-306

[44] M. Van Vyve : *Linear programming extended formulations for the single-item lot-sizing problem with backlogging and constant capacity* Mathematical Programming, 2006, 53-78

[45] D. Villeneuve, J. Desrosiers, M.E. Lubbecke and F. Soumis : *On compact formulations for integer programs solved by column generation* Annals of Operations Research, 2005, 375-388

[46] A. Caprara, F. Furini, and E. Malaguti : *Exact algorithms for the temporal knapsack problem* DEIS, University of Bologna, OR-10-7, 2010

[47] F. Furini, E. Malaguti, R. Medina, A. Persiani, P. Toth : *A Column Generation Heuristic for the Two-Dimensional Cutting Stock Problem with Multiple Stock Size* DEIS, University of Bologna, OR-10-9, 2010

[48] F. Furini, E. Klerides , E.Hadjiconstantinou : *Multi-load AGV dispatching in automated container port terminals* Imperial College London, OR-10-9, 2010

[49] F. Furini A. Persiani, P. Toth : *UAV Mission Planning in non segregated Air Space using Integer Programming* DEIS, University of Bologna, OR-10-11, 2010

[50] "S. Martello and P. Toth" : *"Knapsack Problems: Algorithms and Computer Implementations"* "John Wiley & Sons", "Chichester", 1990

[51] T. Achterberg : *SCIP: Solving constraint integer programs* Mathematical Programming Computation , 2009, 1-41

[52] G. Desaulniers, J. Desrosiers, and S. Spoorendonk : *Cutting planes for branch-and-price algorithms* HEC Montreal, G-2009-52, 2009

[53] M. Junger and S. Thienel : *The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization* Softw. Pract. Exper., 2000, 1325-1352

[54] T. Ralphs and M. Galati : *Decomposition and dynamic cut generation in integer linear programming* Mathematical Programming, 2006, 261-285

[55] Correa, A. I. and Langevin, A. and Rousseau, L. M. : *Scheduling and routing of automated guided vehicles: A hybrid approach*, Computers & Operations Research, 34, 6, 1688 - 1707, 2007,

[56] Grunow, M. and Gonther H.O. and Lehmann M. : *Strategies for dispatching AGVs at automated seaport container terminals*, OR Spectrum, 2006, 28, 587-610,

[57] Grunow, M. and Gonther H.O. and Lehmann M. : *Dispatching multi-load AGVs in highly automated seaport container terminals*, OR Spectrum, 2004, 26, 211-235,

[58] Lau, H. Y. K. and Woo, S. O. : *An agent-based dynamic routing strategy for automated material handling systems*, International of Computer Integrated Manufacturing, 2008, 21, 3, 269-288,

[59] Lin, L. and Gen, M. Huang, YF and Gen, M and Kim, KH, : *A random key-based genetic algorithm for AGV dispatching problem in FMS*, Proceedings of the 4th International Conference on Intelligent Logistics Systems, 2008, 318-330,

[60] Lin, L. and Shinn, S. W. and Gen, M. and Hwang, H. : *Network model and effective evolutionary approach for AGV dispatching in manufacturing system*, of Intelligent Manufacturing, 2006, 17, 4, 465-477,

[61] Mohring, R. H. and Kohler, E. and Gawrilow, E. and and Stenzel, B. : *Conflict-free Real-time AGV Routing*, Operations Research Proceedings 2004, 2005, 2004, series Operations Research Proceedings, 18-24, publisher Springer Berlin Heidelberg,

[62] Nishi, T. and Morinaka, S and Konishi, M. : *A distributed routing method for AGVs under motion delay disturbance*, Robotics and Computer-Integrated Manufacturing, 2007, 23, 5, 517-532,

[63] R. Stahlbock and S. Voss : *Operations research at container terminals: a literature update*, OR Spectrum, 2008, 30, 1-52,

[64] D. Steenken and S. Voss and R. Stahlbock : *Container terminal operation and operations research: a classiffication and literature review*, OR Spectrum, 2004, 26, 3, 3-49,

[65] I. Vis : *Survey of research in the design and control of automated guided vehicle systems*, European of Operational Research, 2006, 170, 677-709,

[66] H. Wren : *Automation - a case for the future*, Port Technology International, 2009, 42, 41-42,

[67] Y. Saanen : *Automated Container Handling*, Freight International, 2008, URL http://www.freight-int.com/categories/automated-container-handling/automated-container-handling.asp,

[68] D. Steenken and S. Voss and R. Stahlbock : *Container terminal operation and operations research: a classiffication and literature review*, OR Spectrum, 2004, 26, 3, 3-49,

[69] Bae, J. and Kim, K. : *A pooled dispatching strategy for automated guided vehicles in port container terminals*, International of Management Science, 2000, 6, 47-67,

[70] Koo, P. and Lee, W. and Koh, S. : *Vehicle dispatching for for container transportation in seaport container terminals*, Proceedings of the 7th international conference of computers and industrial engineering, 2004, address Jeju, Korea,

[71] Europe Container Terminals : *Euromax: a new standard in container handling*, Port Technology International, 2009, 41, 56-61,

[72] Vis, I. F. A. : *Survey of research in the design and control of automated guided vehicle systems*, European of Operational Research, 2006, 170, 677-709,

[73] Stahlbock, R. and Voss, S. : *Operations research at container terminals: a literature update*, OR Spectrum, 2008, 30, 1, 1-52,

[74] Lin, L. and Gen, M. Huang, YF and Gen, M and Kim, KH, : *A random key-based genetic algorithm for AGV dispatching problem in FMS*, Proceedings of the 4th International Conference on Intelligent Logistics Systems, 2008, 318-330,

[75] Lau, H. Y. K. and Woo, S. O. : *An agent-based dynamic routing strategy for automated material handling systems*, International of Computer Integrated Manufacturing, 2008, 21, 3, 269-288,

[76] Kim, K. and Bae, J. : *A look-ahead dispatching method for automated guided vehicles in automated port container terminals*, Transportation Science, 2004, 38, 2, 224-234,

[77] Zhang, L. W. and Ye, R. and Huang, S. Y. and Hsu, W. J. : *Mixed integer programming models for dispatching vehicles at a container terminal*, of Applied Mathematics and Computing, 2005, 17, 145170,

[78] D. Briskorn and A. Drexl and S. Hartmann : *Inventory-based dispatching of automated guided vehicles on container terminals*, OR Spectrum, 2006, 28, 611-630,

[79] D.G. Espinoza, : *Computing with Multi-Row Gomory Cuts*, Oper. Res. Lett., 2010, 38, 115–120

[80] Ferris, M.C. and Horn, J.D., : *Partitioning mathematical programs for parallel solution*, Math. Program., 80, 1, 1998,

[81] du Merle, O. and D. Villeneuve and J. Desrosiers and P. Hansen, : *Stabilized Column Generation*, Discr. Math., 1999, 194, 229–237

[82] H.D. Sherali and Y. Lee and Y. Kim, : *Partial convexification cuts for 0-1 mixed-integer programs*, European J. Oper. Res., 165, 3, 2005, 625-648,

[83] author "T. Achterberg and T. Koch and A. Martin", title "MIPLIB 2003", "Oper. Res. Lett.", 34, 4, "361–372", 2006,

[84] key bapcod, F. Vanderbeck, : *BaPCod – a generic Branch-And-Price Code*, https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/BaPCod, 2005,

[85] J. Puchinger and P.J. Stuckey and M.G. Wallace and S. Brand, : *Dantzig-Wolfe decomposition and branch-and-price solving in G12*, 2010,

[86] T.K. Ralphs and M.V. Galati, : *DIP – Decomposition for Integer Programming*, https://projects.coin-or.org/Dip, 2009,

[87] G. Gamrath and M.E. Lübbecke, : *Experiments with a Generic Dantzig-Wolfe Decomposition for Integer Programs*, 2010, 239–252, 6049,

[88] F. Vanderbeck and L. Wolsey, : *Reformulation and Decomposition of Integer Programs*, 2010,

[89] Aykanat, C. and Pinar, A. and ÇatalyÜrek,Ü.V., : *Permuting Sparse Rectangular Matrices into Block-Diagonal Form*, SIAM J. Sci. Comput., 25, 6, 2004, 1860–1879,

[90] G. Karypis and V. Kumar, : *A fast and high Quality Multilevel Scheme for Partitioning Irregular Graphs*, "SIAM J. Comput.", 1998, 20, 1, 359–392

[91] M. Bergner, A. Caprara, F. Furini, M. Luebbecke, E. Malaguti, E. Traversi, : *Partial Convexification of General MIPs by Dantzig-Wolfe Reformulation*, O. Gunluk, G.J. Woeginger (eds.) Proceedings of the Fifteenth Conference on Integer Programming and Combinatorial Optimization (IPCO'11), 2011 (in press)

[92] A. Caprara and E. Malaguti and P. Toth, : *A Freight Service Design Problem for a Railway Corridor*, "Tran. Sci.", 2010 (in press),

[93] Darmann, A., and Pferschy, U. and J. Schauer, : *Resource Allocation with Time Intervals* "2009"

[94] Calinescu, Gruia and Chakrabarti, Amit and Karloff, Howard J. and Rabani, Yuval, : *Improved Approximation Algorithms for Resource Allocation*, booktitle Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization, 2002, 401–414,

[95] M. Bartlett, A. M. Frisch, Y. Hamadi, I. Miguel, S. A. Tarim and C. Unsworth, : *The Temporal Knapsack Problem and its Solution*, Proceedings of the International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 2005, 34–48,

[96] Arkin, E. M. and Silverberg, E. B., : *Scheduling jobs with fixed start and end times*, Discrete Applied Mathematics, 18, 1, 1987, 1–8,

[97] Hall, N. G. and M. J. Magazine : *Maximizing the value of a space mission* EJOR, "78", "224-241", "1994"

[98] Chen, B. and Hassin, R. and M. Tzur : *Allocation of Bandwidth and Storage* "IIE Transactions", "24", "501-507", "2002"

[99] Vanderbeck, F. and L.A. Wolsey : *Reformulation and Decomposition of Integer Programs* 2010, "431–502"

[100] Fekete, S. and Schepers, J. : *New classes of fast lower bounds for bin packing problems* "Mathematical Programming", "91", "11-31", "2001"

[101] Vanderbeck, F. : *Computational study of a column generation algorithm for bin packing and cutting stock problems* "Mathematical Programming", "86", "565-594", "1999"

[102] Gabrel, V. and Minoux, M. : *A scheme for exact separation of extended cover inequalities and application to multidimensional knapsack problems* "Operations Research Letters", "30", "252-264", "2002"

[103] Gilmore, P.C. and R.E. Gomory : *A Linear Programming Approach to the Cutting Stock Problem* OR, 1961, 9, "849–859"

[104] Gilmore, P.C. and R.E. Gomory : *A Linear Programming Approach to the Cutting Stock Problem – Part II* OR, 1963, 11, "863–888"

[105] Gilmore, P.C. and R.E. Gomory : *Multistage Cutting Stock Problems of Two and More Dimensions* OR, 1965, 13, "94–120"

[106] Alvarez-Valdes, R. and Parajon, A. and J.M. Tamarit : *A Computational Study of LP-based Heuristic Algorithms for Two-Dimensional Guillotine Cutting Stock Problems* "OR Spektrum", 2002, 24, "179–192"

[107] Riehme, J. and Scheithauer, G. and J. Terno : *The solution of two-stage guillotine cutting stock problems having extremely varying order demands* "European Journal of Operational Research", 1996, 91, "543–552"

[108] Cintra, G.F. and Miyazawa, F.K. and Wakabayashi, Y. and E.C. Xavier : *Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation* "European Journal of Operational Research", 2008, 191, "61–85"

[109] Lodi, A. and M. Monaci : *Integer linear programming models for 2-staged two-dimensional Knapsack problems* "Mathematical Programming", 2003, 94, "257–178"

[110] Lodi, A. and Martello, S. and M. Monaci : *Two-dimensional packing problems: A survey* "European Journal of Operational Research", 2002, 141, "241–252"

[111] "F. Furini and E. Malaguti and R. Medina Durán and A. Persiani and P. Toth" : *"A Column Generation Heuristic for the Two-Dimensional Two-Staged Guillotine Cutting Stock Problem with Multiple Stock Size"* "submitted to an International Journal", "2011"

[112] P. Bonsma and J. Schulz and A. Wiese, : *A Constant Factor Approximation Algorithm for Unsplittable Flow on Paths*, CoRR, abs/1102.3643, 2011,

[113] Wäscher, G. and Haussner, H. and H. Schumann : *"An Improved Typology of Cutting and Packing Problems"* "European Journal of Operational Research", 2007, 183, "1109–1130"

[114] "http://people.brunel.ac.uk/ mastjjb/jeb/info.html"

[115] CPLEX 10, ILOG "User's Manual and Reference Manual, ILOG, S.A., http://www.ilog.com/ (2006)"

[116] IBM ILOG CPLEX v12.1 "User's Manual for CPLEX, ftp://ftp.boulder.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf (2011)"

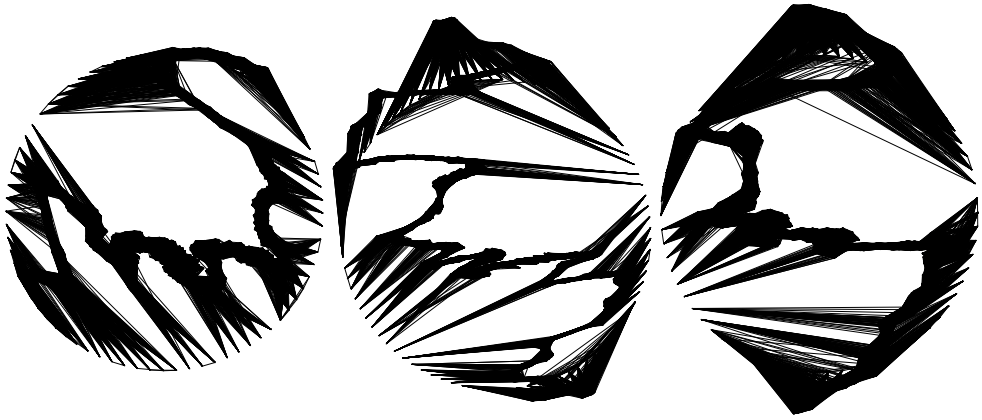
[117] COIN-OR Linear Program Solver "An Open Source code for Solving Linear Programming Problems, http://www.coin-or-org/Clp/index.html"

[118] J.R. Tebboth, "A Computational Study of Dantzig-WolfeDecomposition" University of Buckingham 2001 Ph.D. Thesis

Appendix A

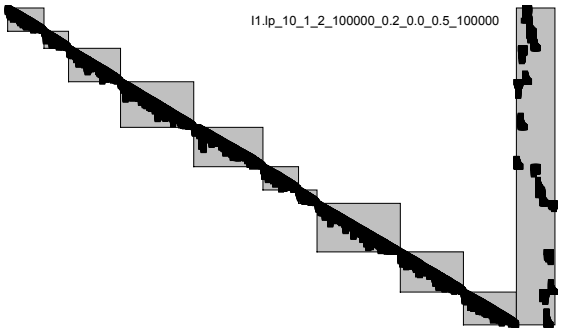
Decomposition Pictures

(a) *I45* (b) *I55* (c) *I65*

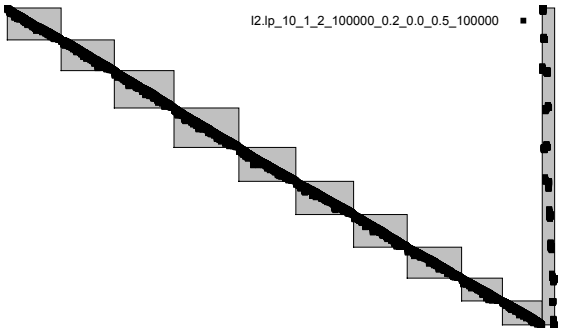


(d) *I75* (e) *I85* (f) *I95*

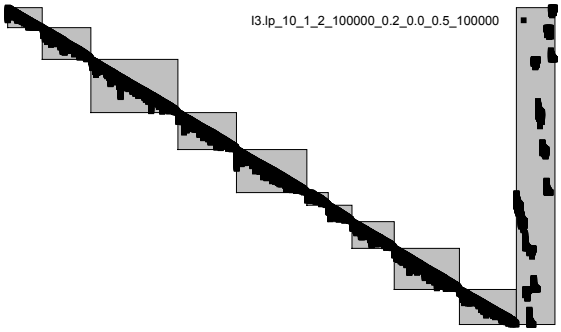
Figure A.1: Selected TKP instances Clique Graph Drawings



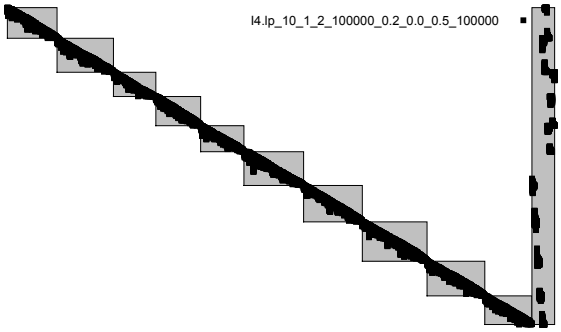
(a) *I45*



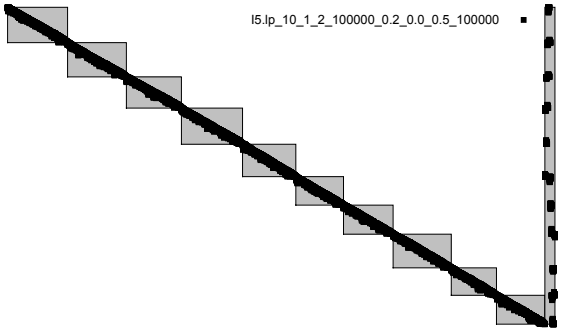
(b) *I55*



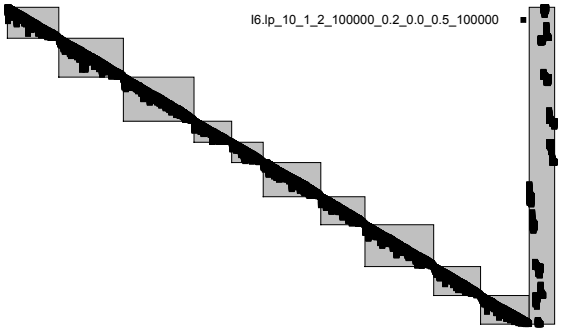
(c) *I65*



(d) *I75*

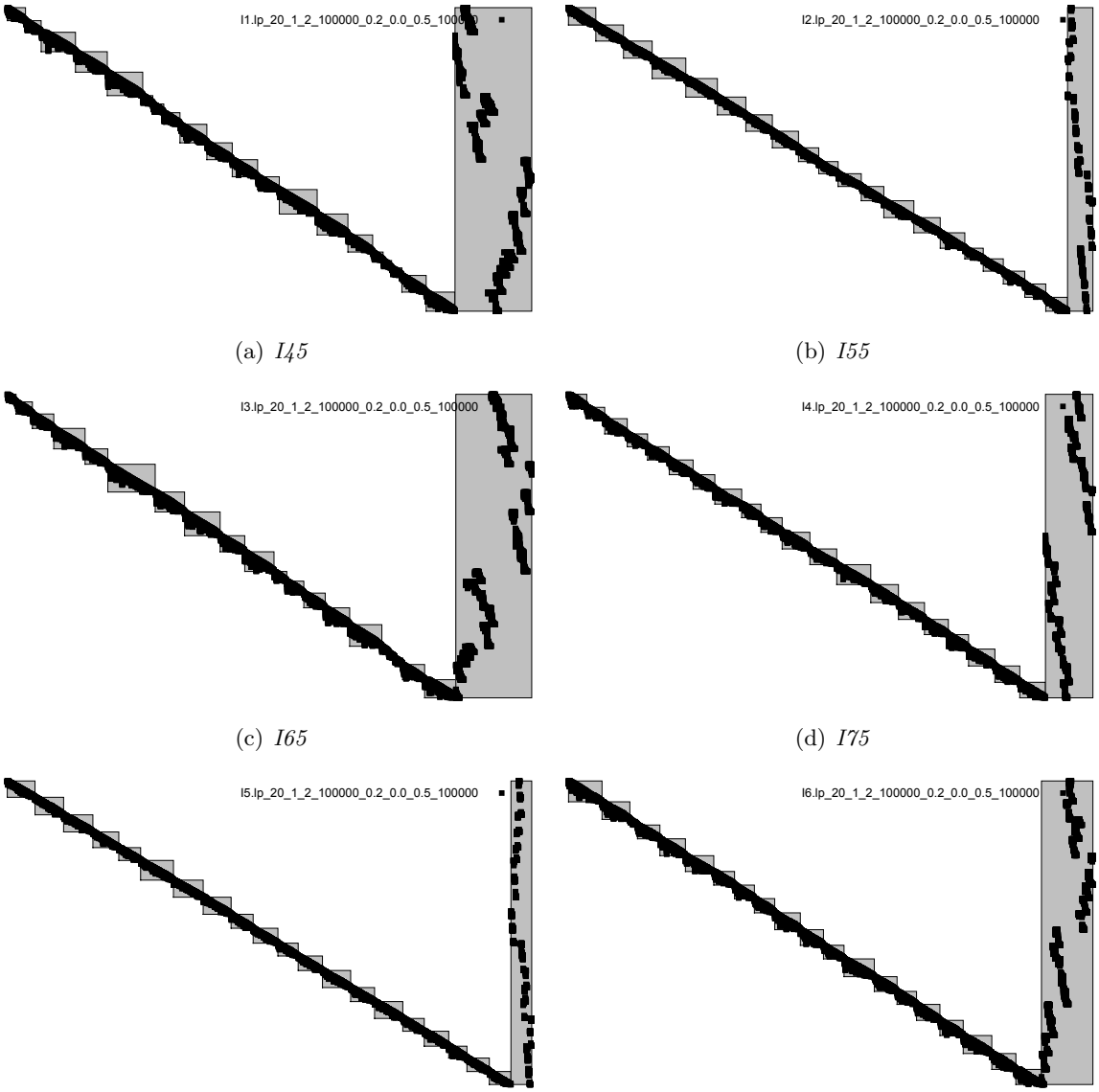


(e) *I85*



(f) *I95*

Figure A.2: 10 Blocks Decomposition

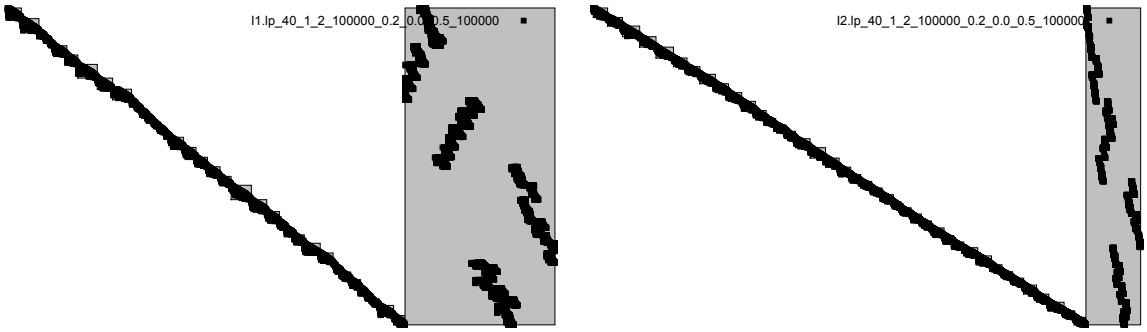


(a) *I45* (b) *I55*

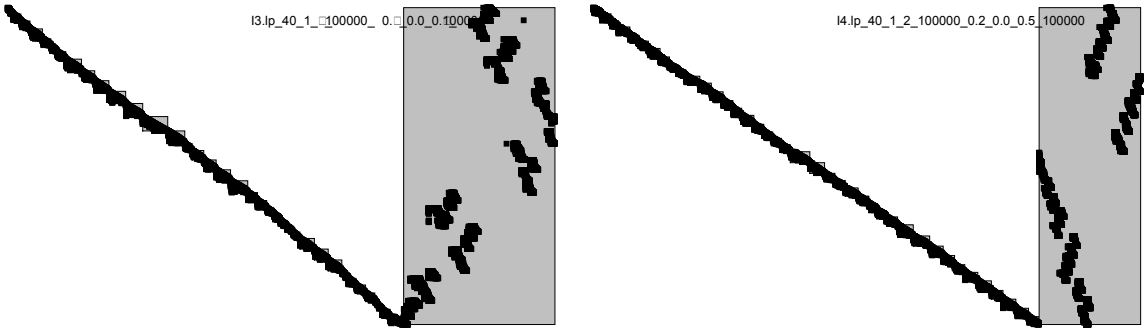
(c) *I65* (d) *I75*

(e) *I85* (f) *I95*

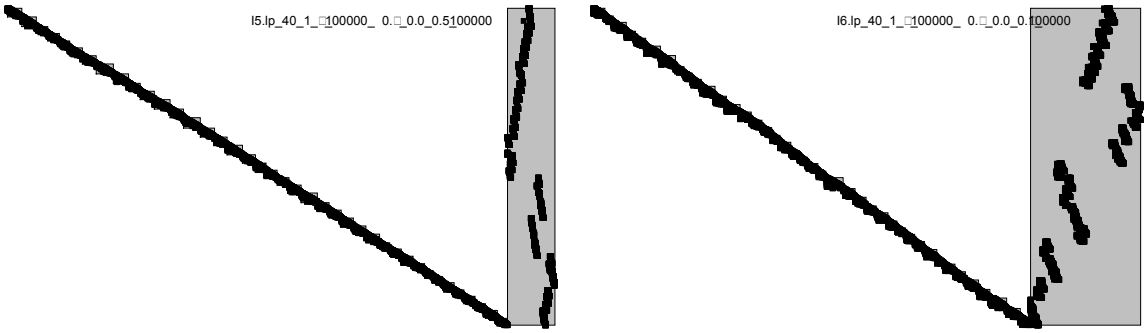
Figure A.3: 20 Blocks Decomposition



(a) *I45* (b) *I55*



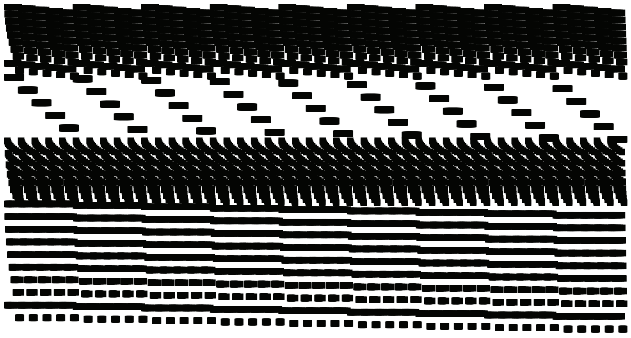
(c) *I65* (d) *I75*



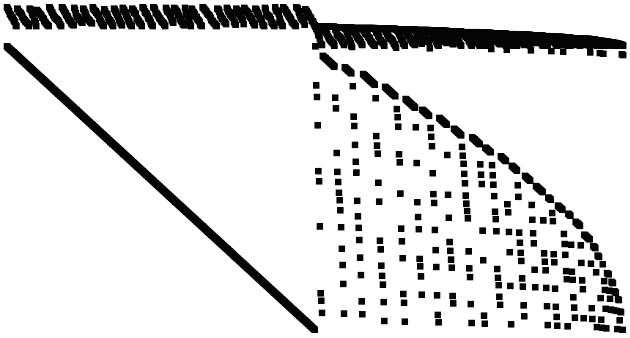
(e) *I85* (f) *I95*

Figure A.4: 40 Blocks Decomposition

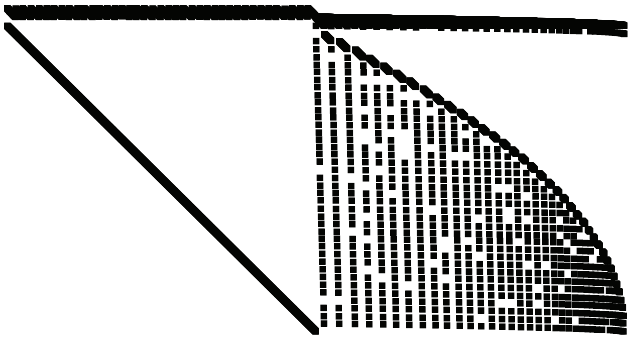
10teams



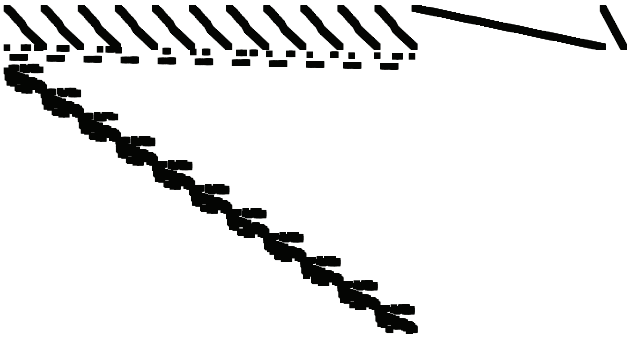
aflow30a



aflow40b

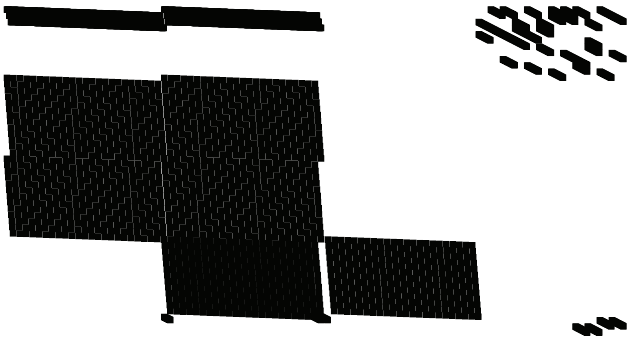


fiber



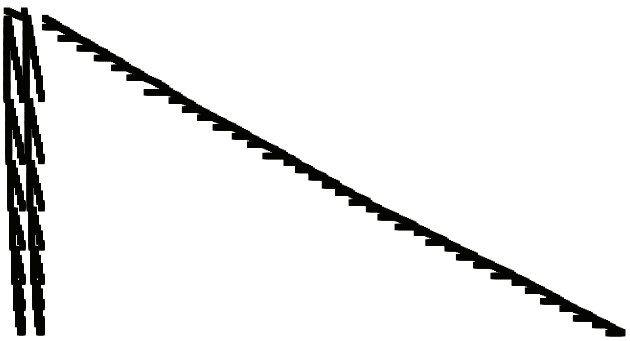
fixnet6

gesa2-o

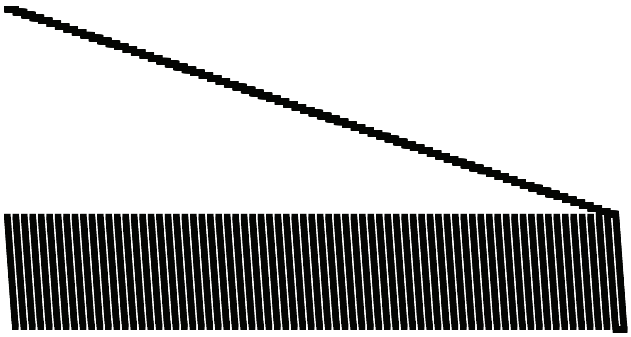


gesa2

glass4

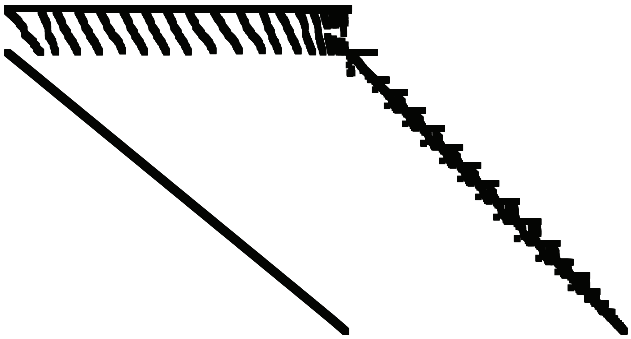


harp2

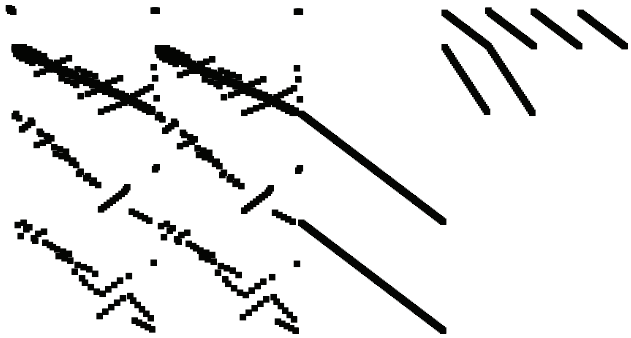


manna81

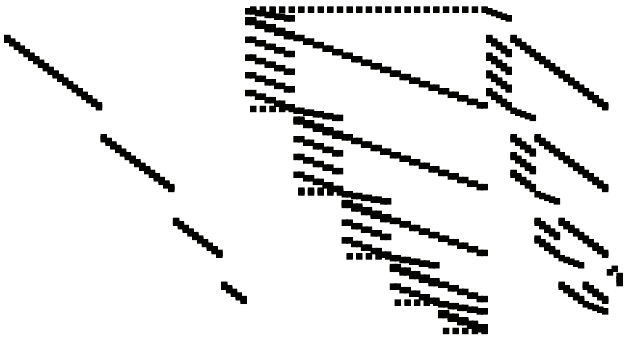
mkc



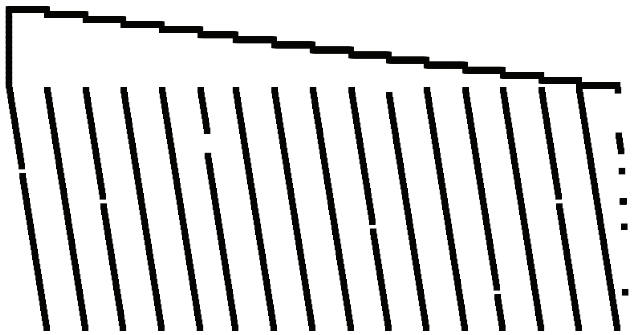
modglob



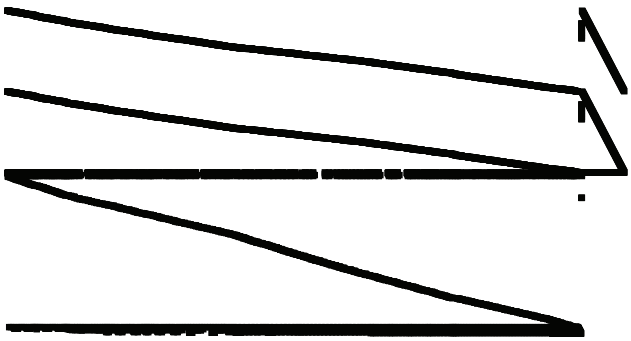
noswot



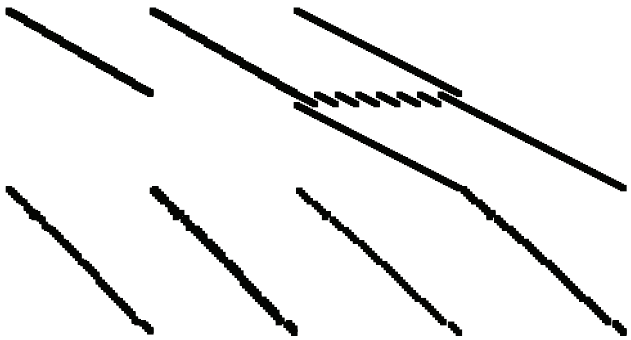
opt1217



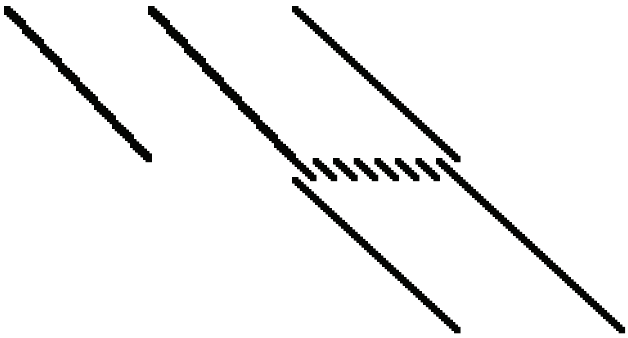
p2756



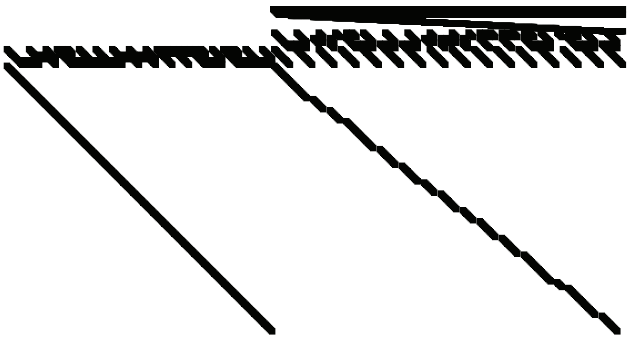
pp08aCUTS



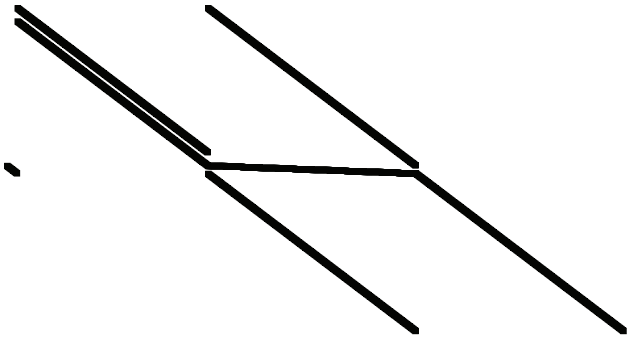
pp08a



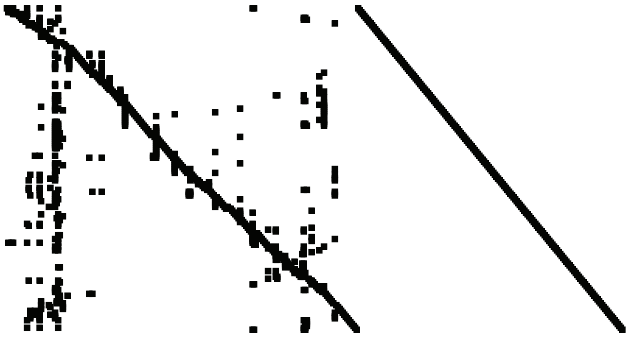
rout



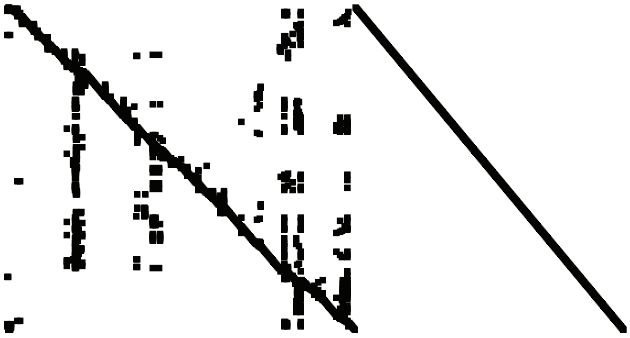
set1ch



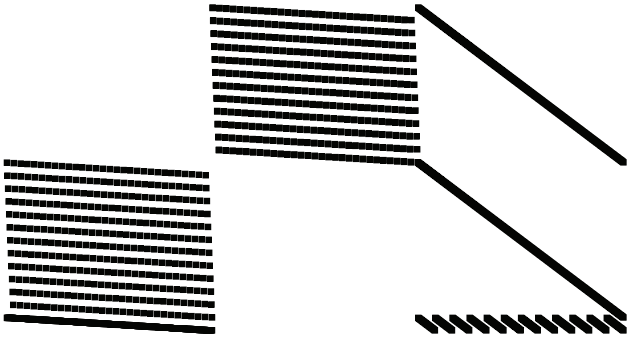
timtab1



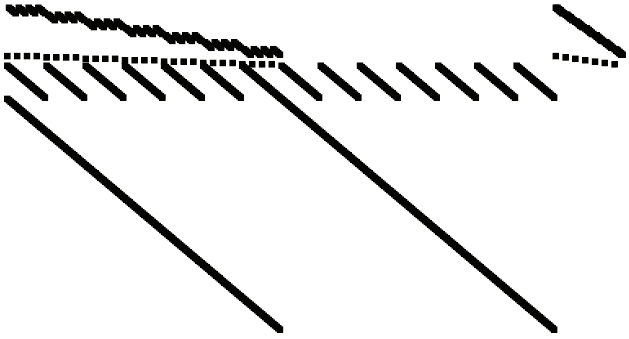
timtab2



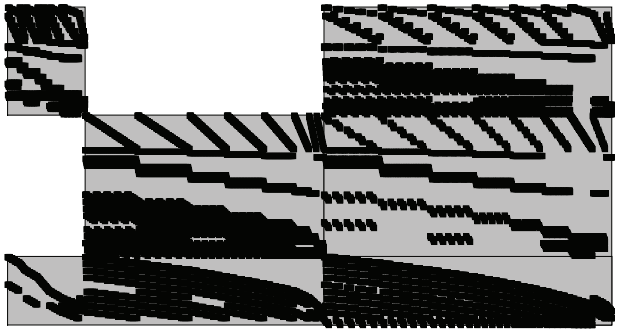
tr12-30



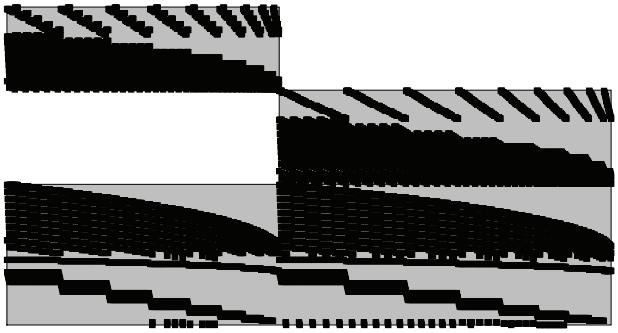
vpm2



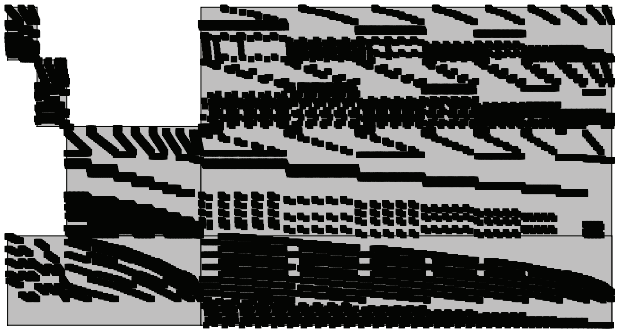
10teams_2_1_2_100000_0.2_0_0.5



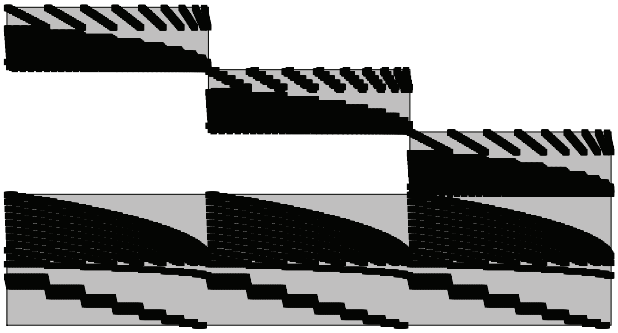
10teams_2_1_2_5_0.2_0_0.5



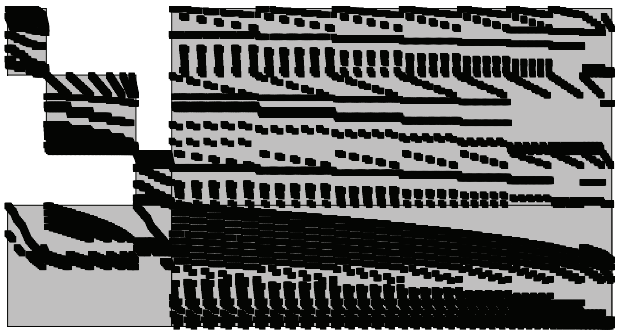
10teams_3_1_2_100000_0.2_0_0.5



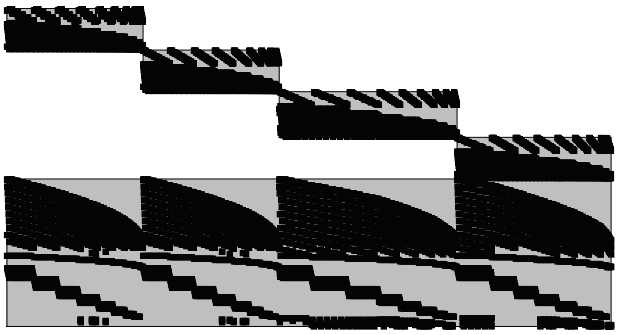
10teams_3_1_2_5_0.2_0_0.5



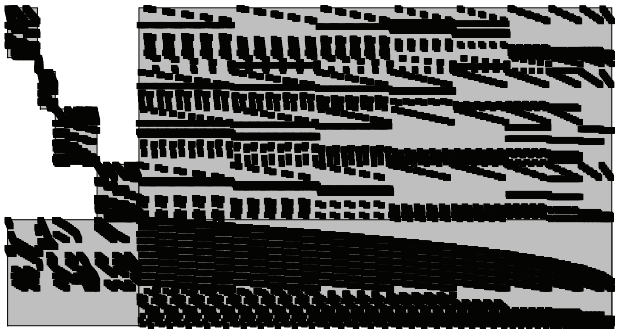
10teams_4_1_2_100000_0.2_0_0.5



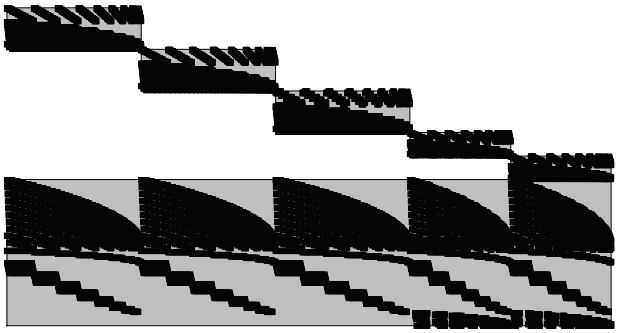
10teams_4_1_2_5_0.2_0_0.5



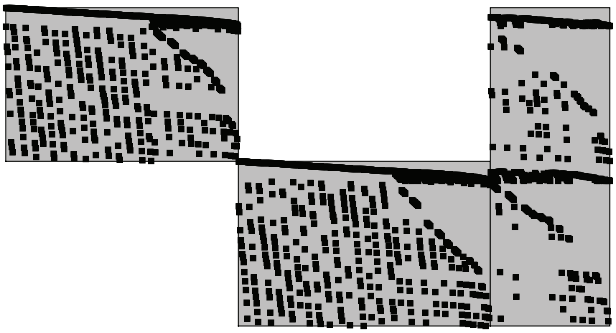
10teams_5_1_2_100000_0.2_0_0.5



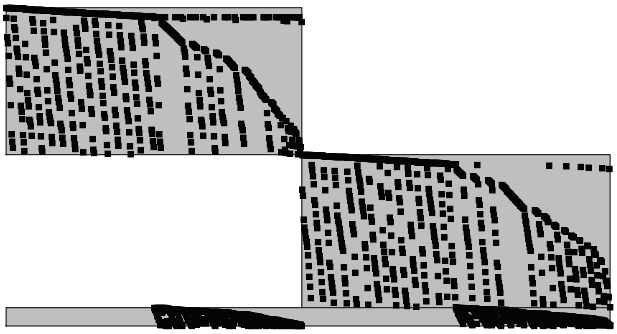
10teams_5_1_2_5_0.2_0_0.5



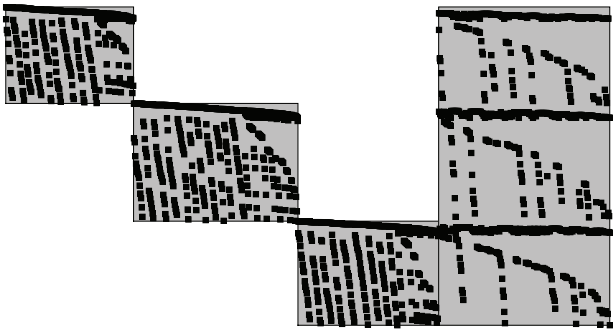
aflow30a_2_1_2_100000_0.2_0_0.5



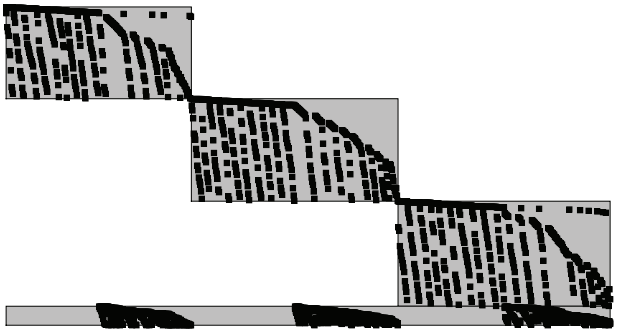
aflow30a_2_1_2_5_0.2_0_0.5



aflow30a_3_1_2_100000_0.2_0_0.5

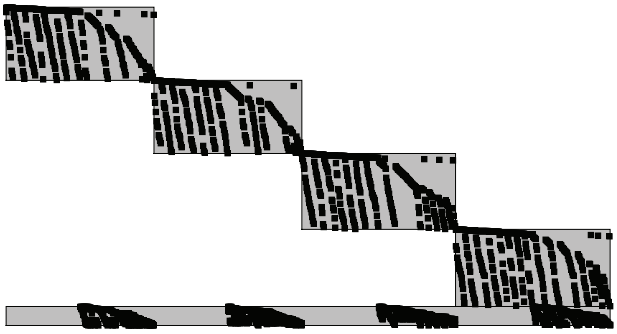


aflow30a_3_1_2_5_0.2_0_0.5

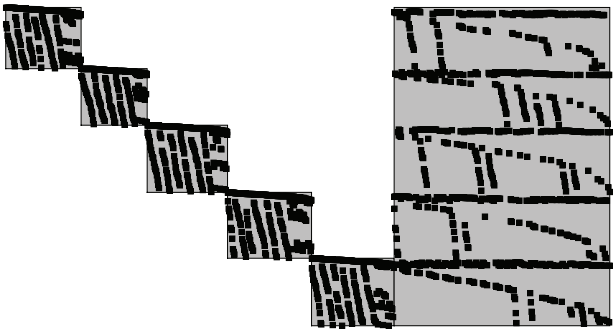


aflow30a_4_1_2_100000_0.2_0_0.5

aflow30a_4_1_2_5_0.2_0_0.5

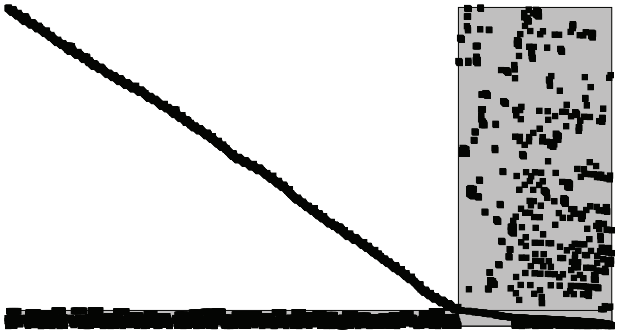


aflow30a_5_1_2_100000_0.2_0_0.5

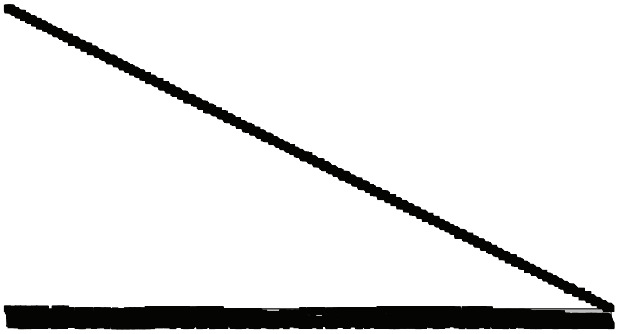


aflow30a_5_1_2_5_0.2_0_0.5

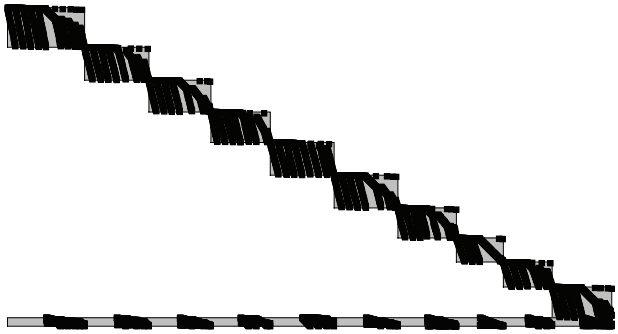
aflow40b_100_1_2_100000_0.2_0_0.5



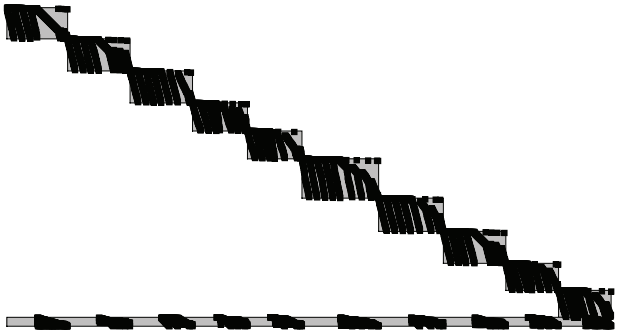
aflow40b_100_1_2_5_0.2_0_0.5



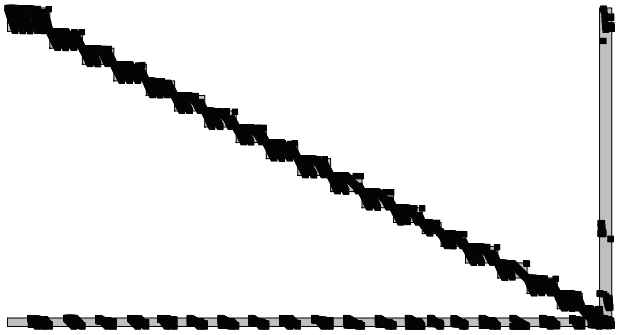
aflow40b_10_1_2_10000040.24040.5



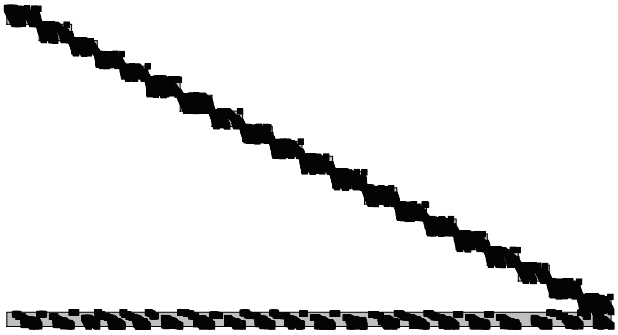
aflow40b_10_1_2_5_0.2_0_0.5



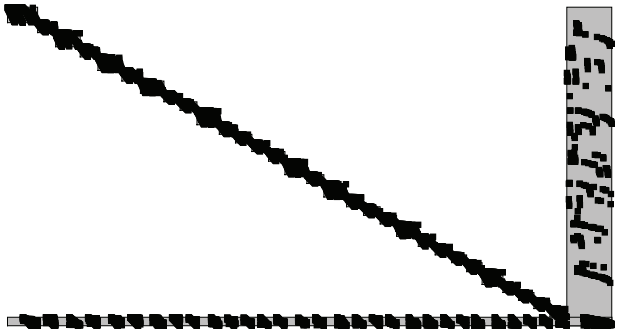
aflow40b_20_1_2_100000_0.2_0_0.5



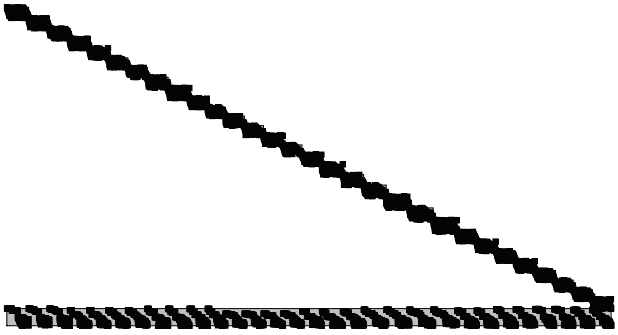
aflow40b_20_1_2_5_0.2_0_0.5



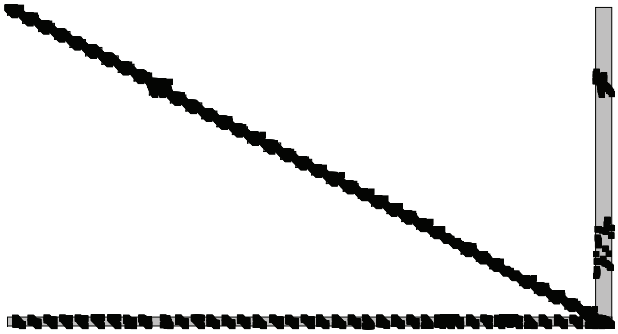
aflow40b_30_1_2_100000_0.2_0_0.5



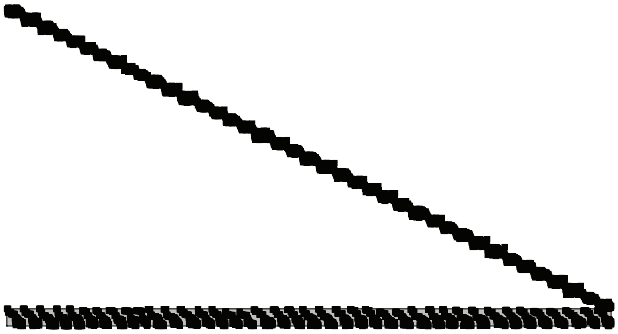
aflow40b_30_1_2_5_0.2_0_0.5



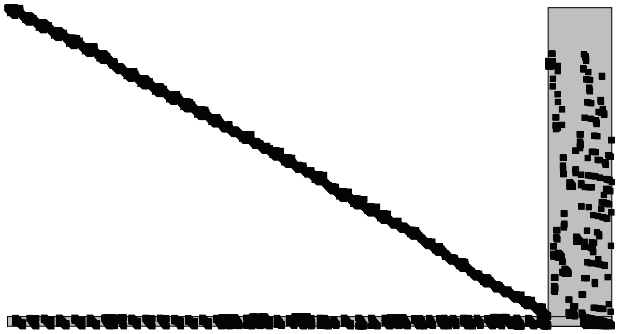
aflow40b_40_1_2_10000040.24040.5



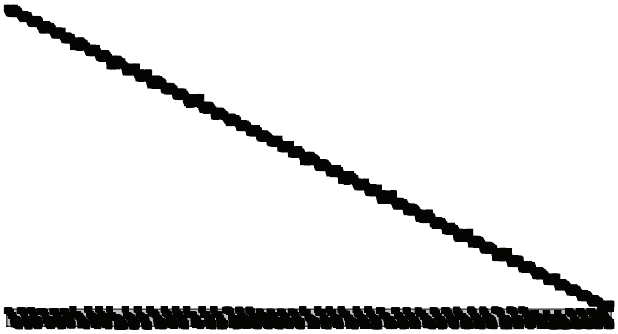
aflow40b_40_1_2_5_0.2_0_0.5



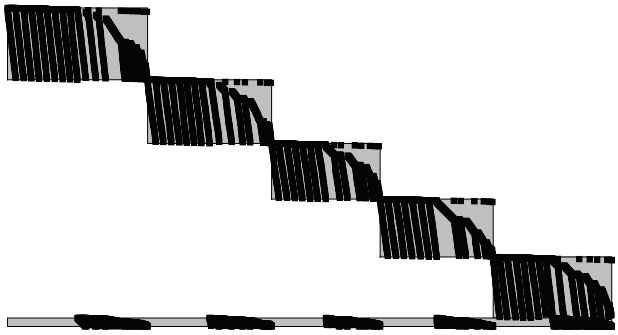
aflow40b_50_1_2_100000_0.2_0_0.5



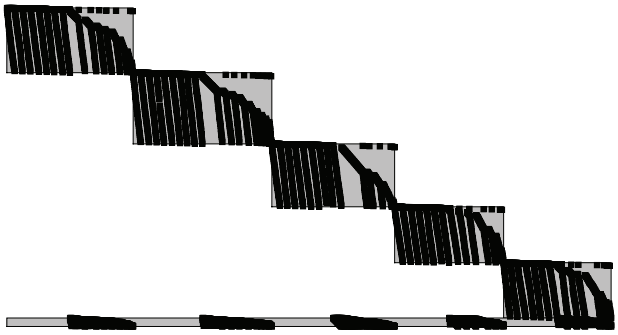
aflow40b_50_1_2_5_0.2_0_0.5



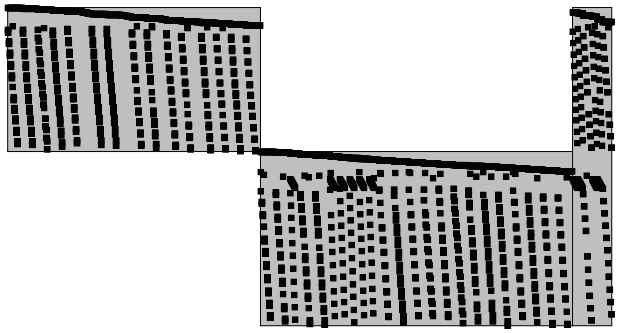
aflow40b_5_1_2_100000_0.2_0_0.5



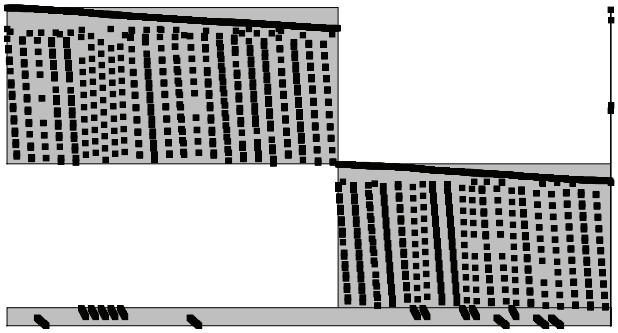
aflow40b_5_1_2_5_0.2_0_0.5



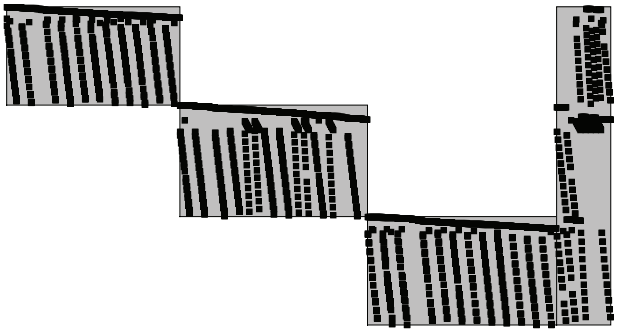
fiber_2_1_2_100000_0.2_0_0.5



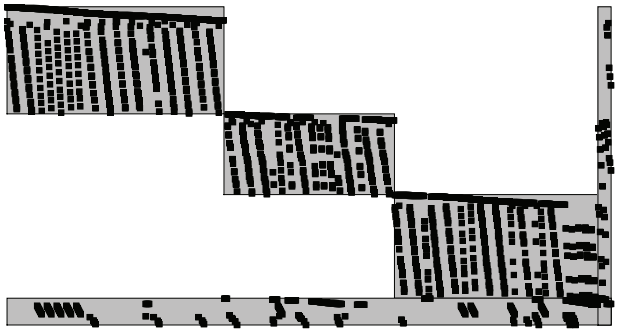
fiber_2_1_2_5_0.2_0_0.5



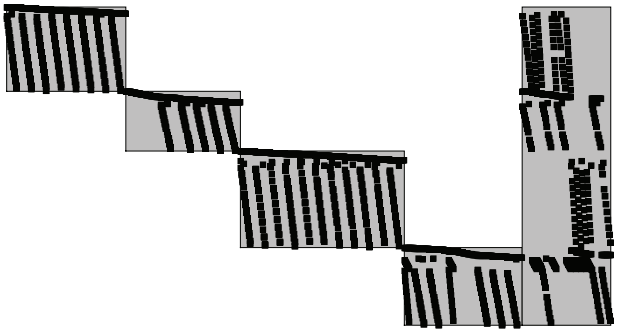
fiber_3_1_2_100000_0.2_0_0.5



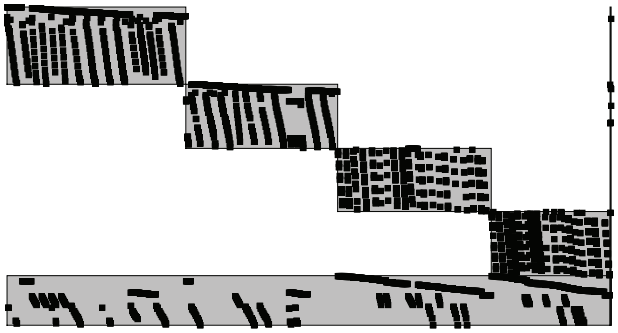
fiber_3_1_2_5_0.2_0_0.5



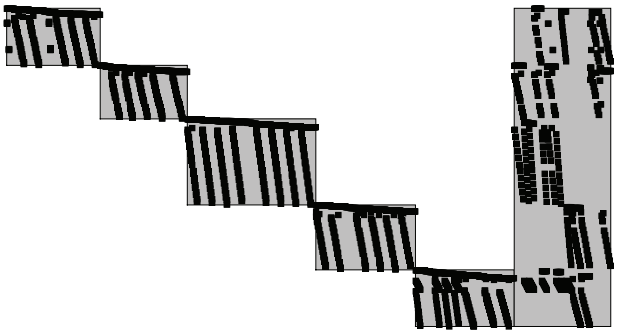
fiber_4_1_2_100000_0.2_0_0.5



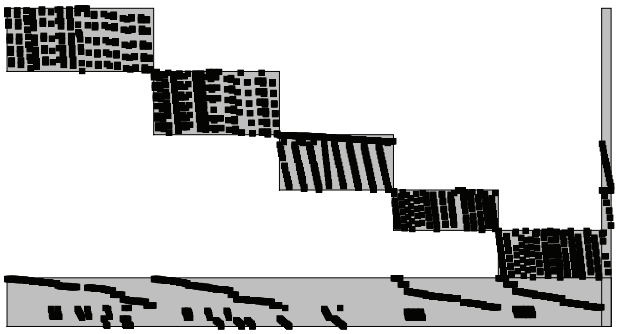
fiber_4_1_2_5_0.2_0_0.5



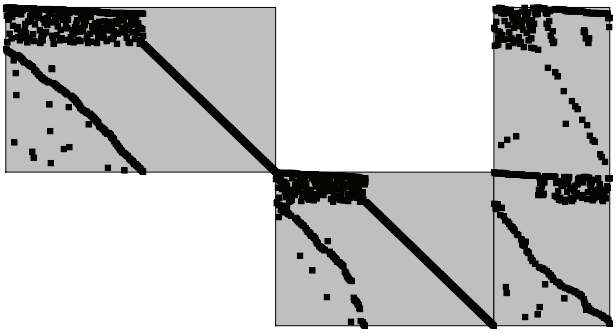
fiber_5_1_2_100000_0.2_0_0.5



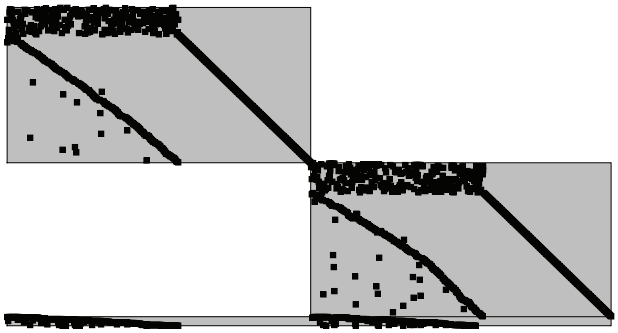
fiber_5_1_2_5_0.2_0_0.5



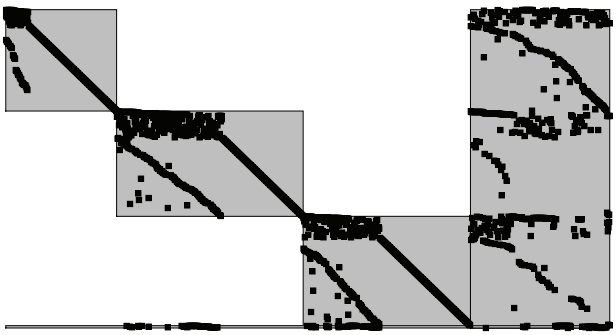
fixnet6_2_1_2_100000_0.2_0_0.5



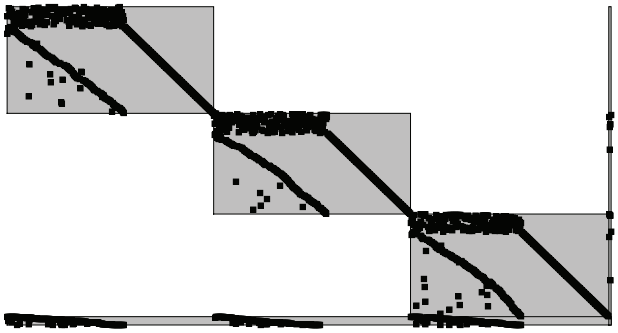
fixnet6_2_1_2_5_0.2_0_0.5



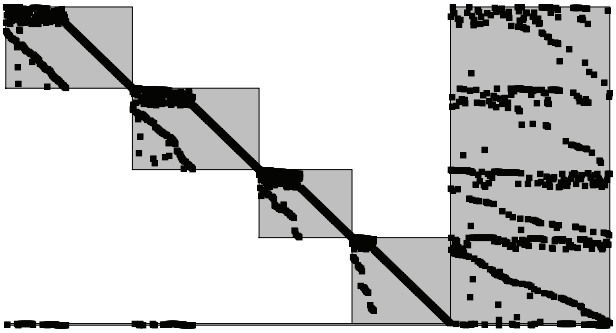
fixnet6_3_1_2_100000_0.2_0_0.5



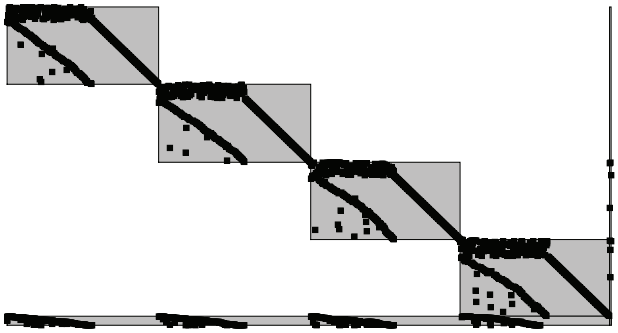
fixnet6_3_1_2_5_0.2_0_0.5



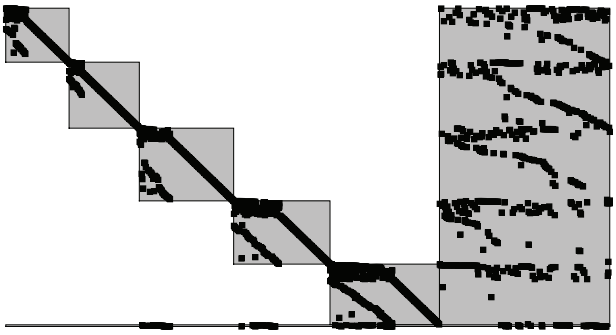
fixnet6_4_1_2_100000_0.2_0_0.5



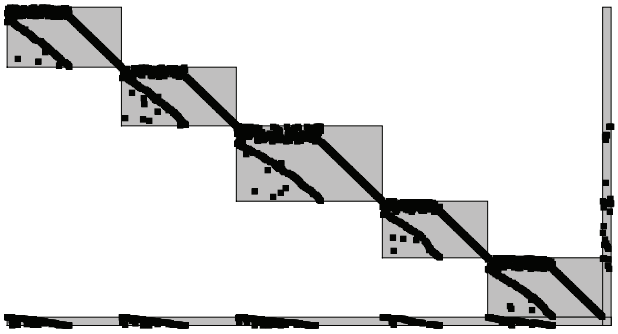
fixnet6_4_1_2_5_0.2_0_0.5



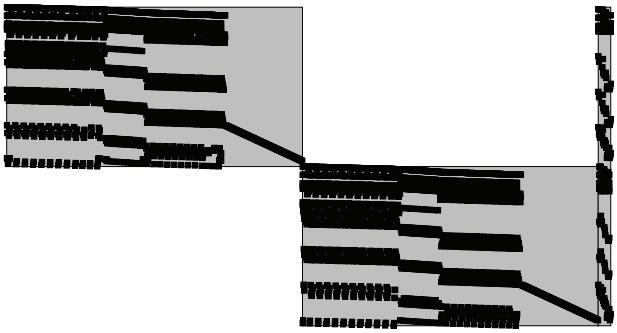
fixnet6_5_1_2_100000_0.2_0_0.5



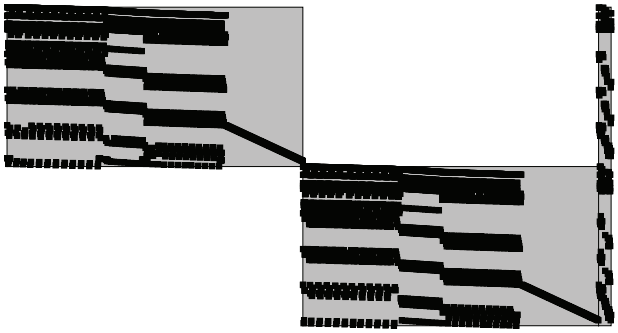
fixnet6_5_1_2_5_0.2_0_0.5



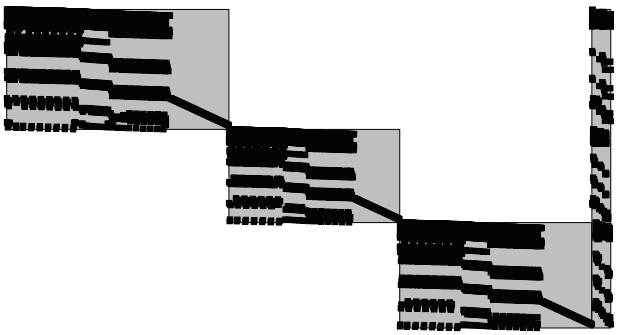
gesa2_2_1_2_100000_0.2_0_0.5



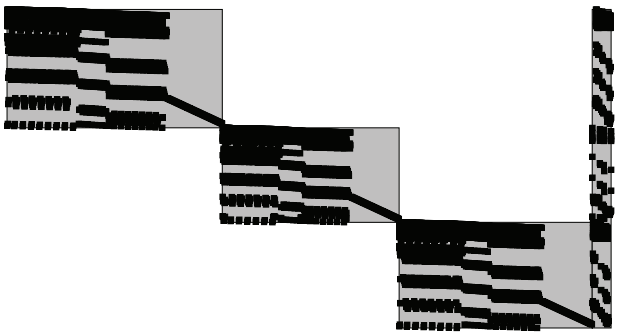
gesa2_2_1_2_5_0.2_0_0.5



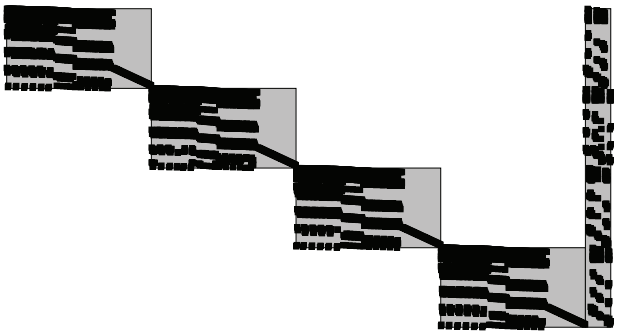
gesa2_3_1_2_100000_0.2_0_0.5



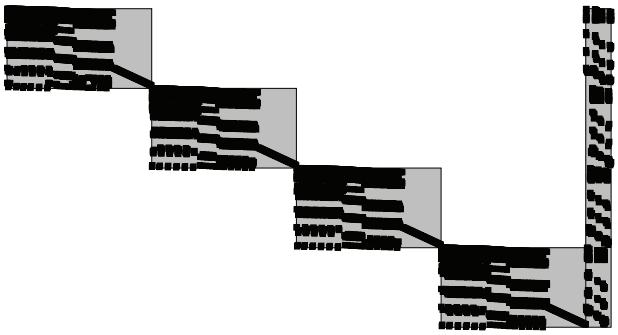
gesa2_3_1_2_5_0.2_0_0.5



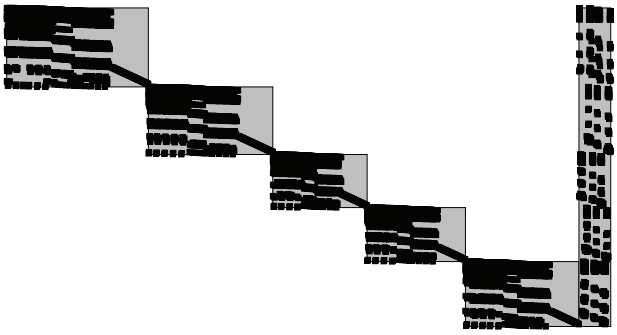
gesa2_4_1_2_100000_0.2_0_0.5



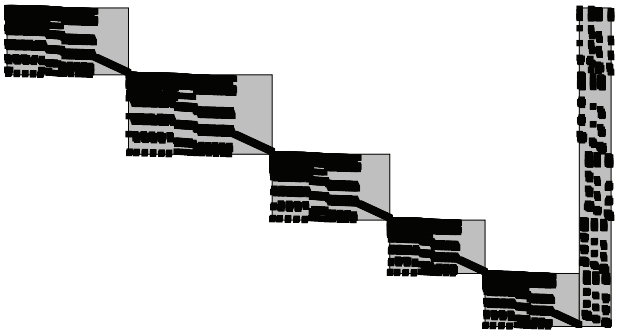
gesa2_4_1_2_5_0.2_0_0.5



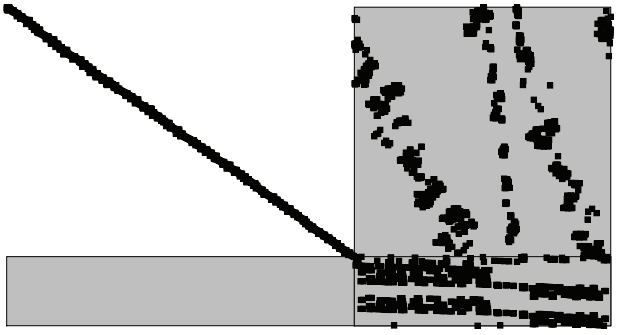
gesa2_5_1_2_100000_0.2_0_0.5



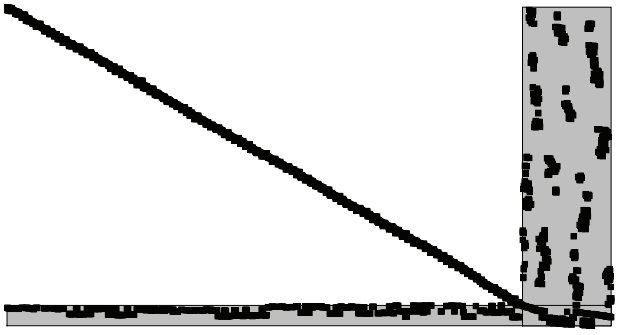
gesa2_5_1_2_5_0.2_0_0.5



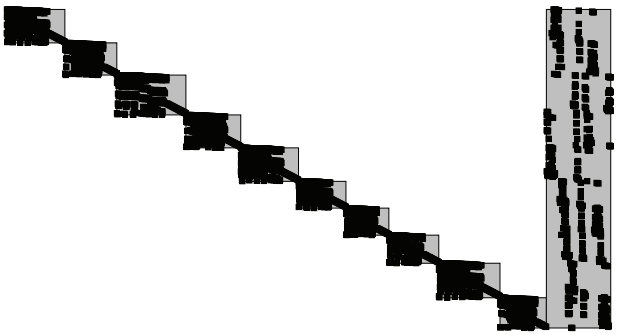
gesa2-o_100_1_2_100000_0.2_0_0.5



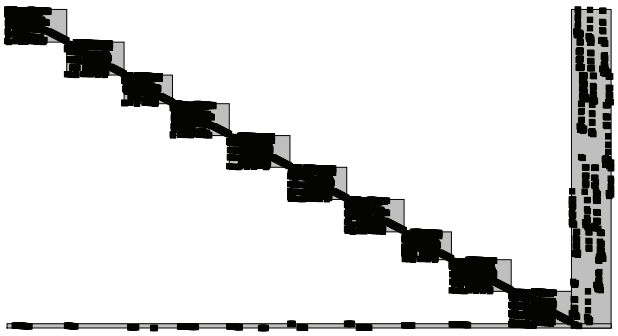
gesa2-o_100_1_2_5_0.2_0_0.5



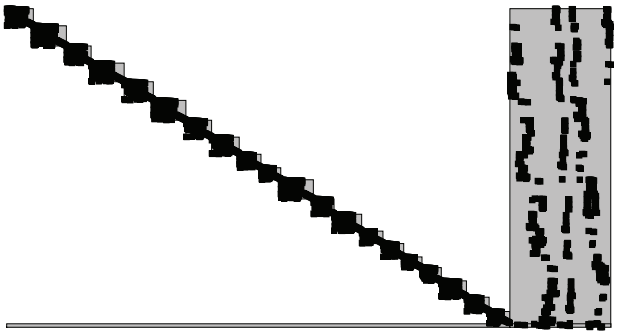
gesa2-o_10_1_2_100000_0.2_0_0.5



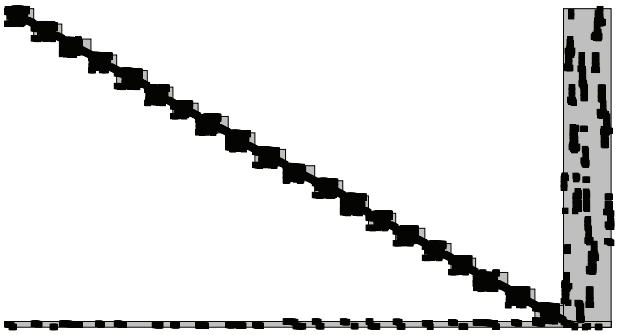
gesa2-o_10_1_2_5_0.2_0_0.5



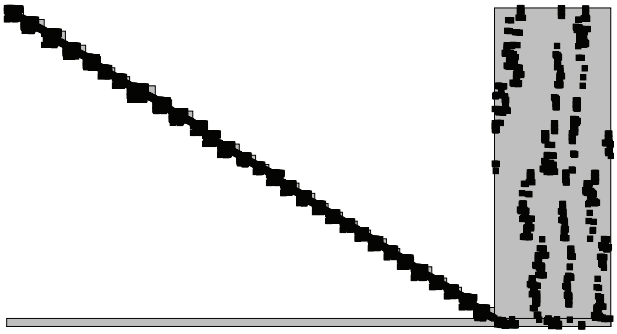
gesa2-o_20_1_2_100000_0.2_0_0.5



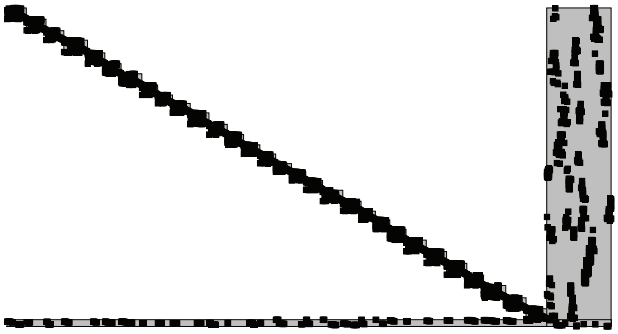
gesa2-o_20_1_2_5_0.2_0_0.5



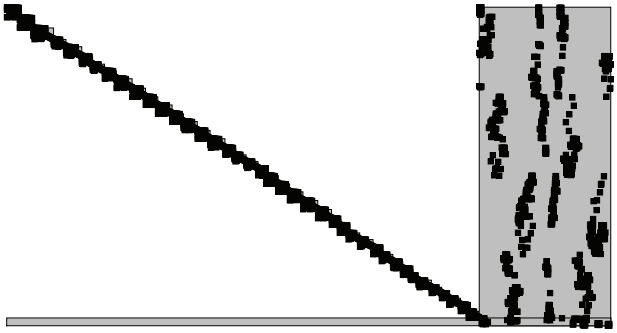
gesa2-o_30_1_2_100000_0.2_0_0.5



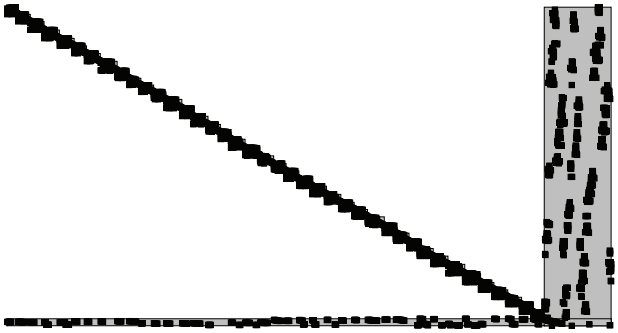
gesa2-o_30_1_2_5_0.2_0_0.5



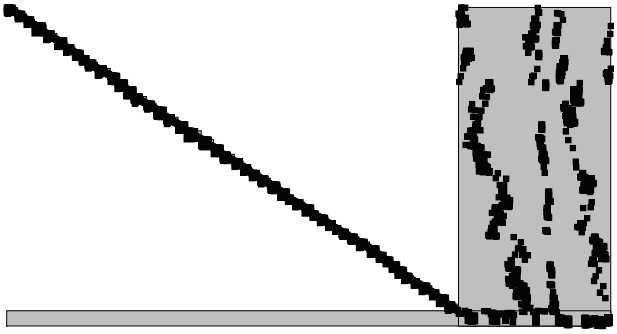
gesa2-o_40_1_2_100000_0.2_0_0.5



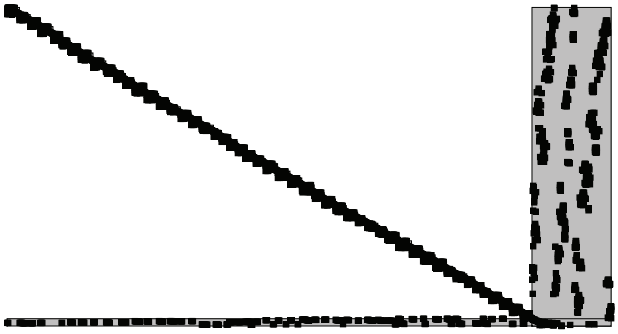
gesa2-o_40_1_2_5_0.2_0_0.5



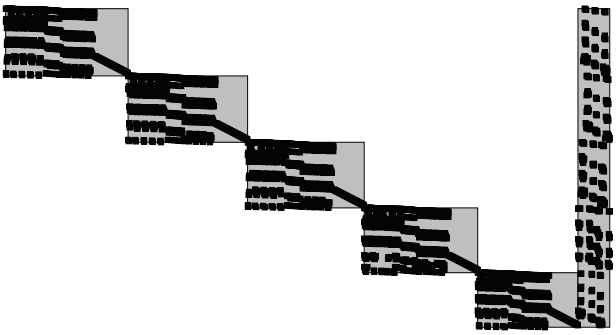
gesa2-o_50_1_2_100000_0.2_0_0.5



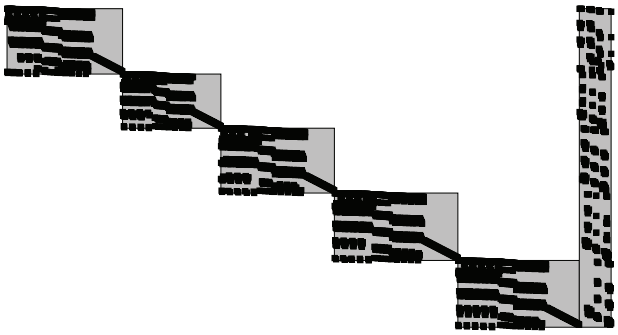
gesa2-o_50_1_2_5_0.2_0_0.5



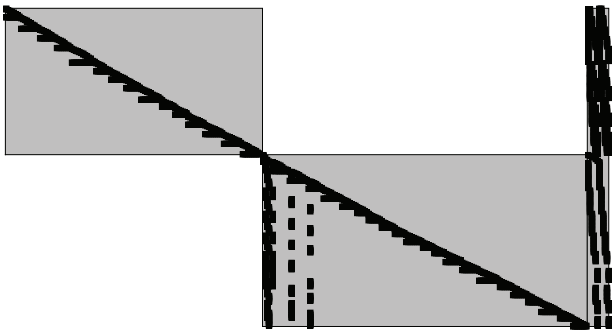
gesa2-o_5_1_2_100000_0.2_0_0.5



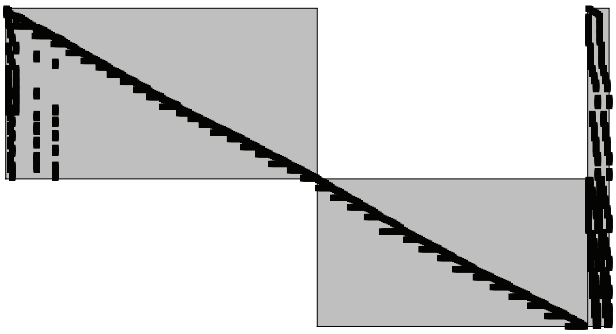
gesa2-o_5_1_2_5_0.2_0_0.5



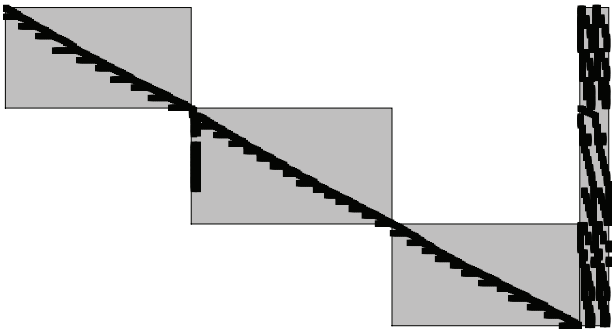
glass4_2_1_2_100000_0.2_0_0.5



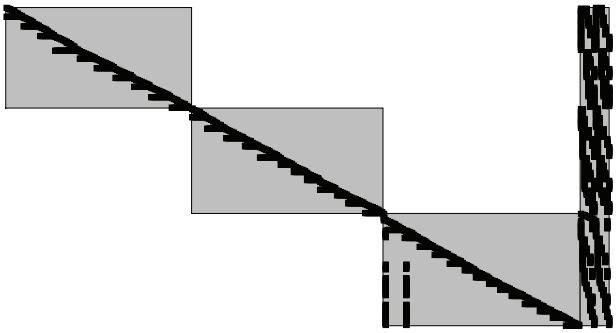
glass4_2_1_2_5_0.2_0_0.5



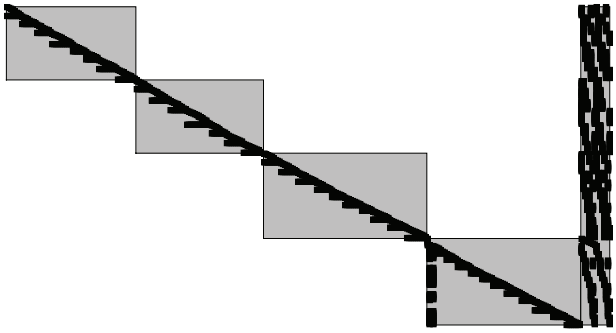
glass4_3_1_2_100000_0.2_0_0.5



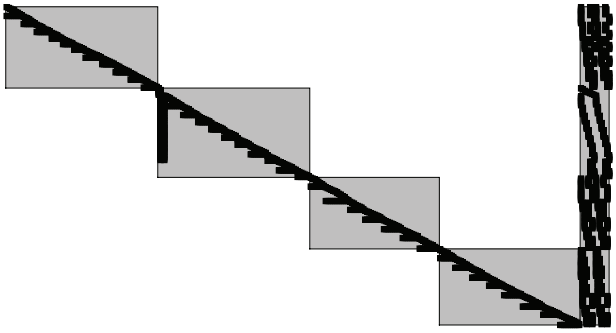
glass4_3_1_2_5_0.2_0_0.5



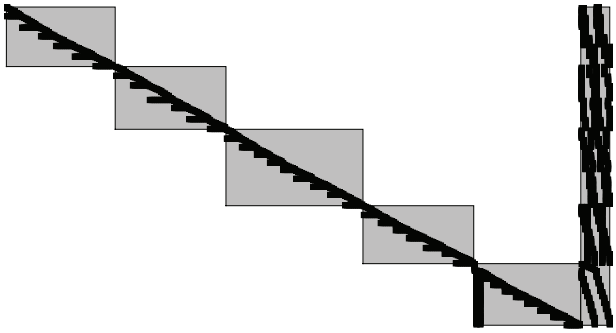
glass4_4_1_2_100000_0.2_0_0.5



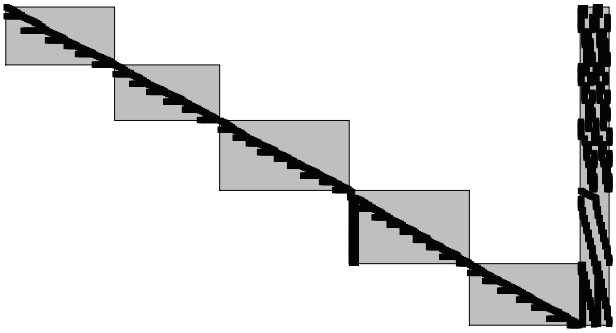
glass4_4_1_2_5_0.2_0_0.5



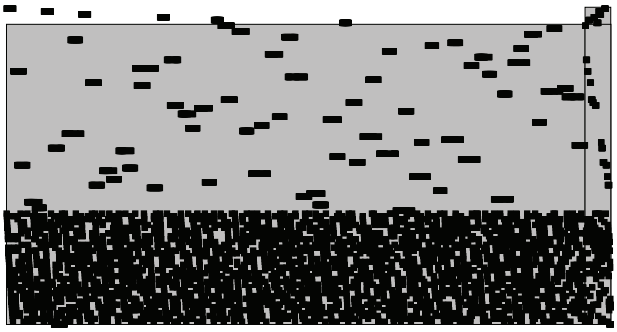
glass4_5_1_2_100000_0.2_0_0.5



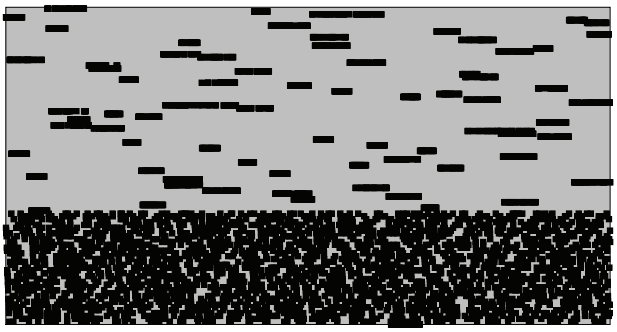
glass4_5_1_2_5_0.2_0_0.5



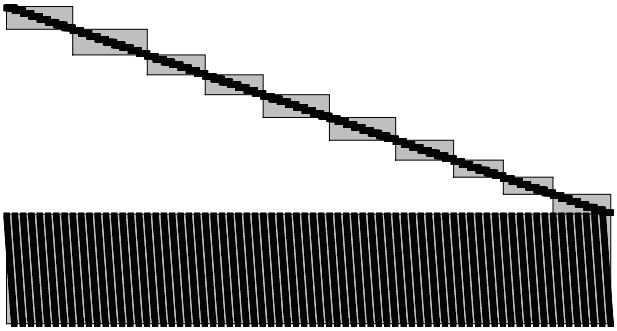
harp2_100_1_2_100000_0.2_0_0.5



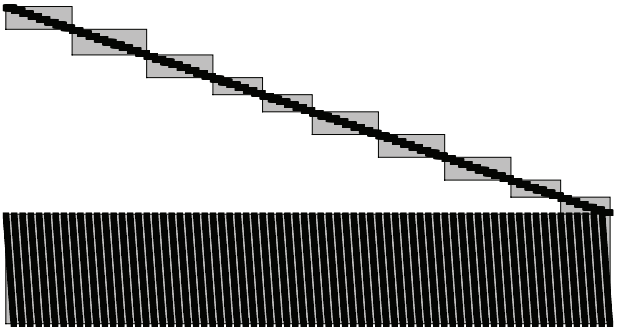
harp2_100_1_2_5_0.2_0_0.5



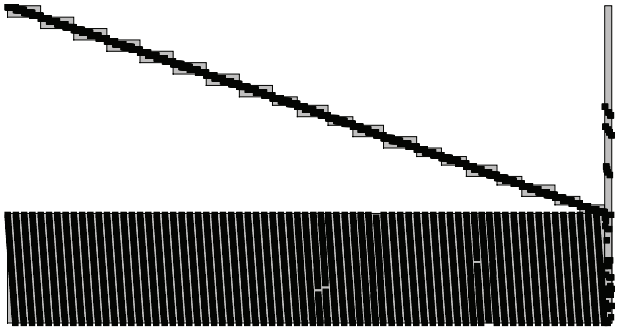
harp2_10_1_2_100000_0.2_0_0.5



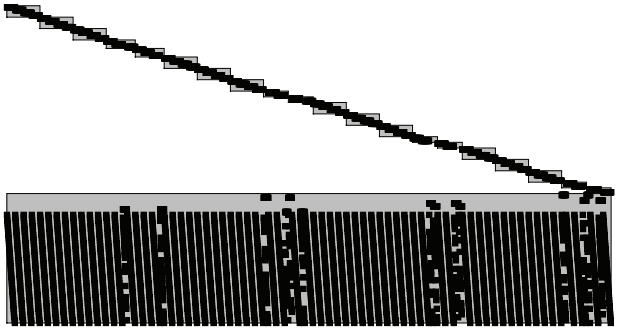
harp2_10_1_2_5_0.2_0_0.5



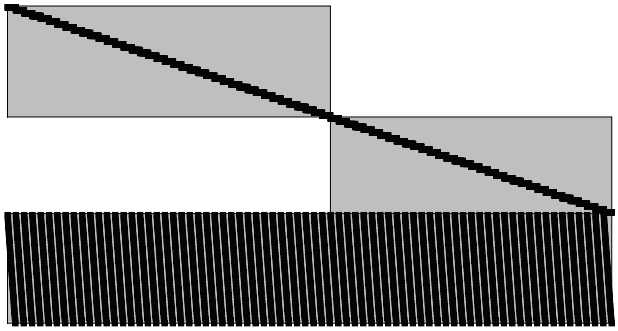
harp2_20_1_2_100000_0.2_0_0.5



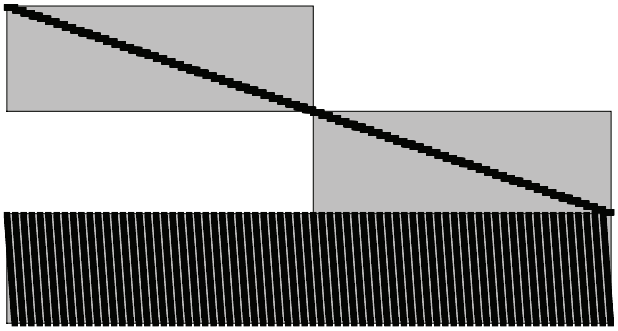
harp2_20_1_2_5_0.2_0_0.5



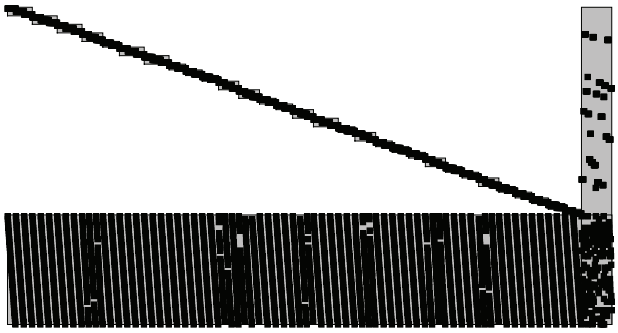
harp2_2_1_2_100000_0.2_0_0.5



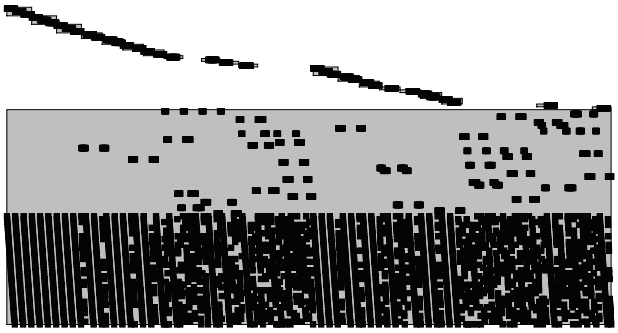
harp2_2_1_2_5_0.2_0_0.5



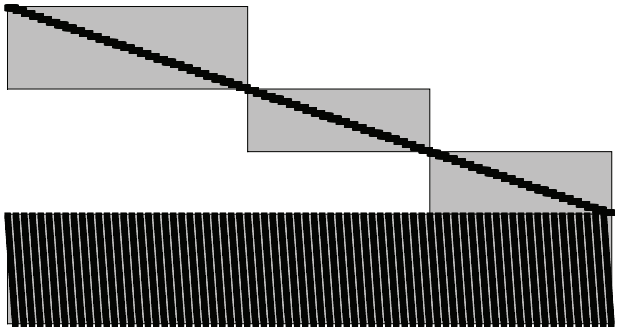
harp2_30_1_2_100000_0.2_0_0.5



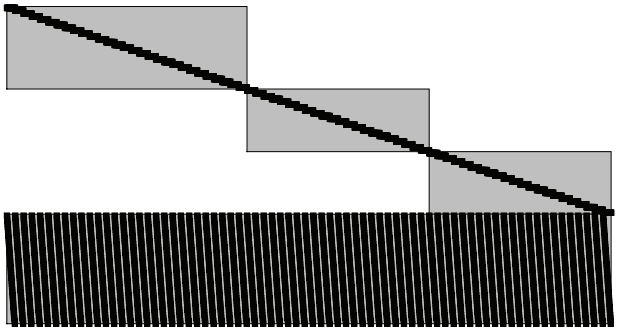
harp2_30_1_2_5_0.2_0_0.5



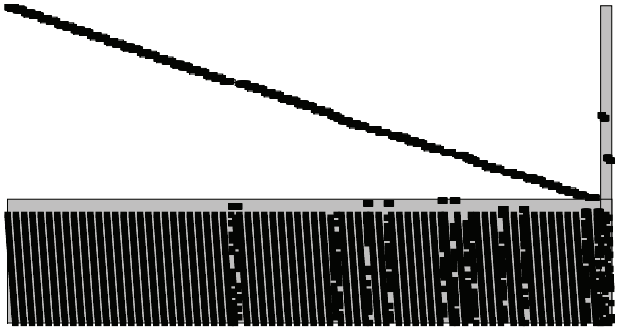
harp2_3_1_2_100000_0.2_0_0.5



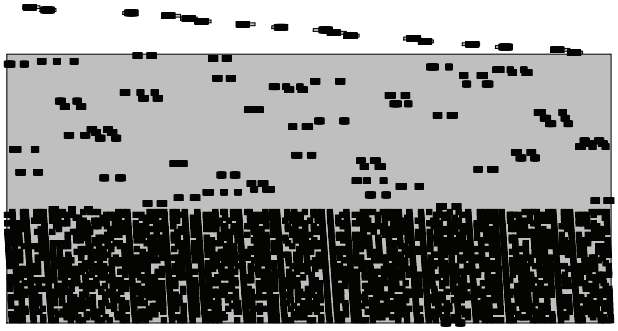
harp2_3_1_2_5_0.2_0_0.5



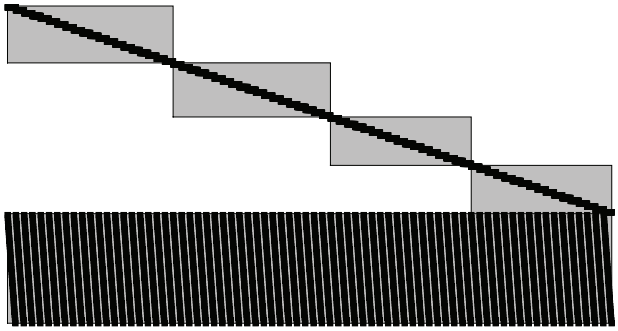
harp2_40_1_2_100000_0.2_0_0.5



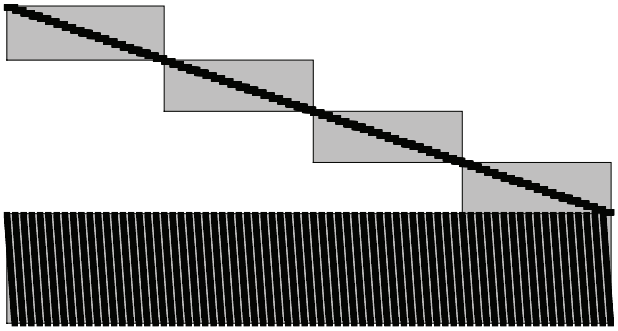
harp2_40_1_2_5_0.2_0_0.5



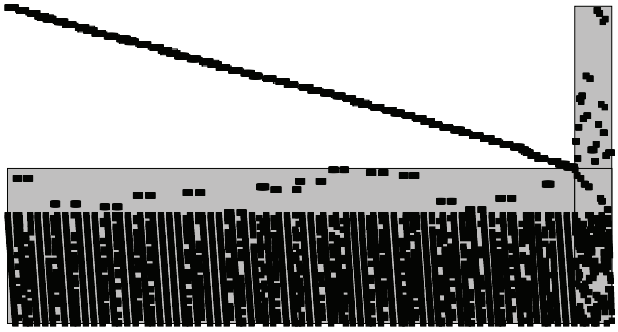
harp2_4_1_2_100000_0.2_0_0.5



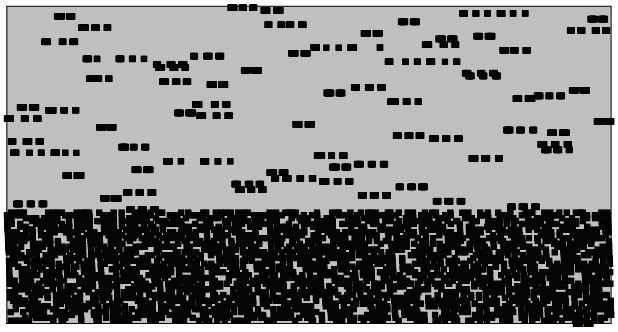
harp2_4_1_2_5_0.2_0_0.5



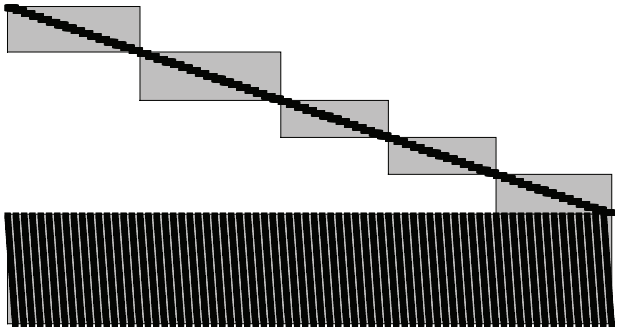
harp2_50_1_2_100000_0.2_0_0.5



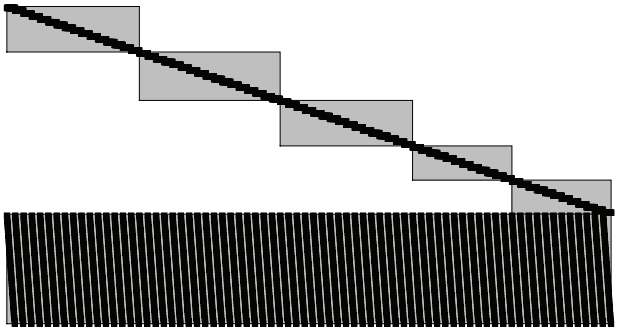
harp2_50_1_2_5_0.2_0_0.5



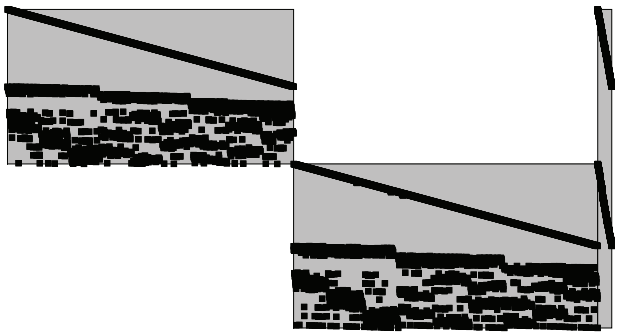
harp2_5_1_2_100000_0.2_0_0.5



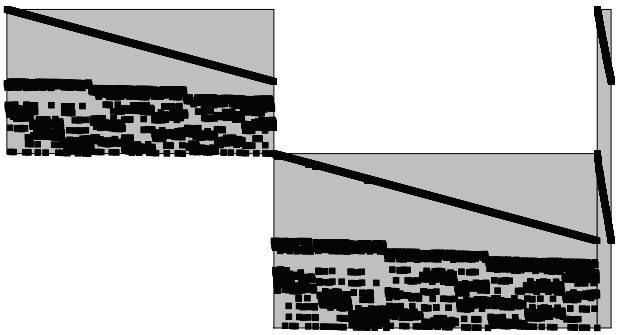
harp2_5_1_2_5_0.2_0_0.5



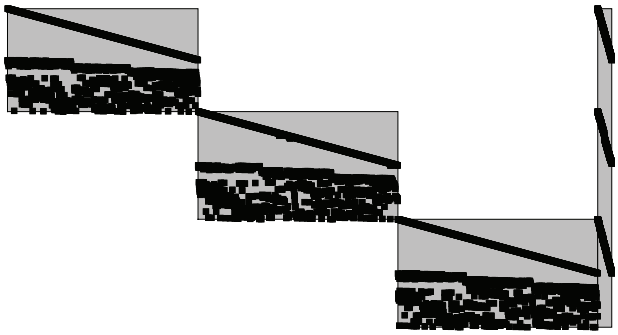
manna81_2_1_2_100000_0.2_0_0.5



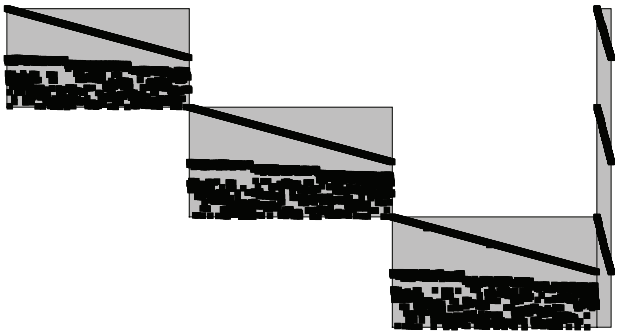
manna81_2_1_2_5_0.2_0_0.5



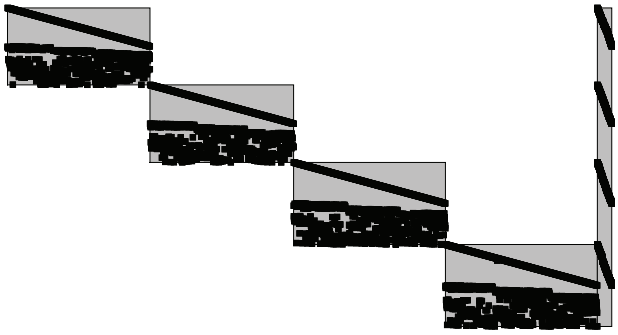
manna81_3_1_2_100000_0.2_0_0.5



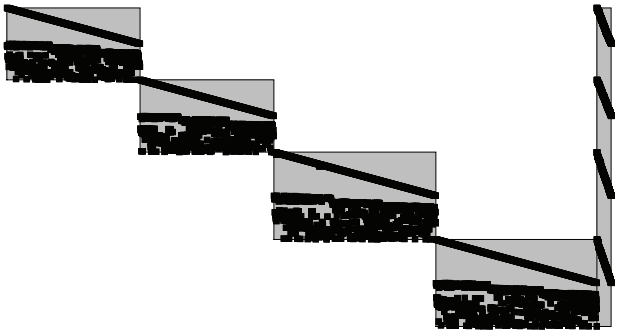
manna81_3_1_2_5_0.2_0_0.5



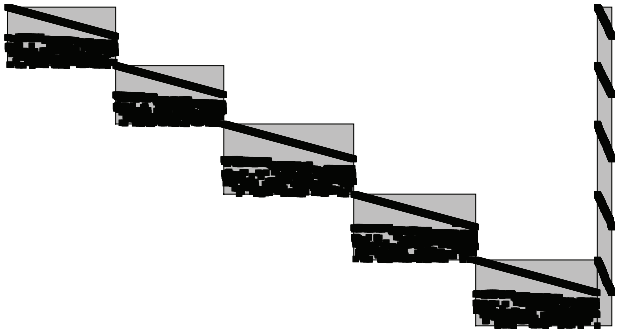
manna81_4_1_2_100000_0.2_0_0.5



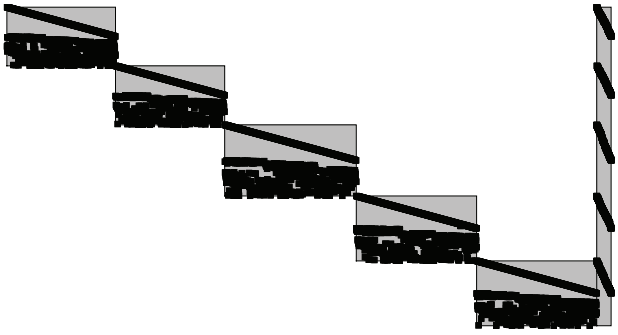
manna81_4_1_2_5_0.2_0_0.5



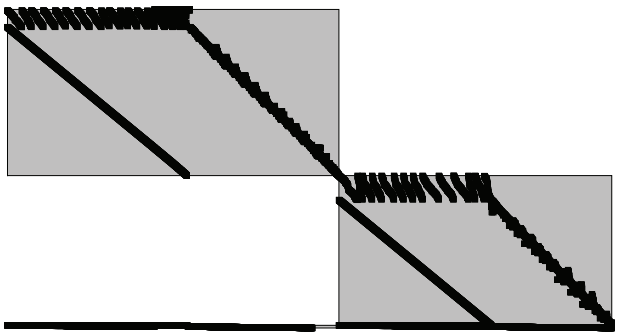
manna81_5_1_2_100000_0.2_0_0.5



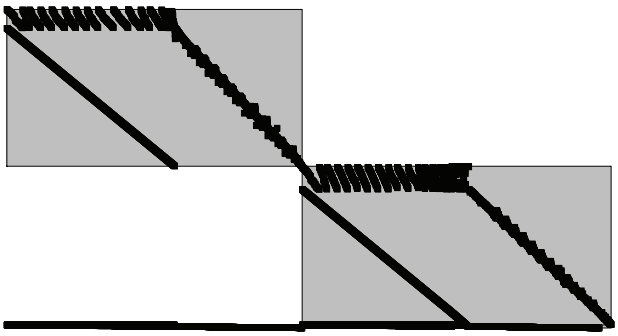
manna81_5_1_2_5_0.2_0_0.5



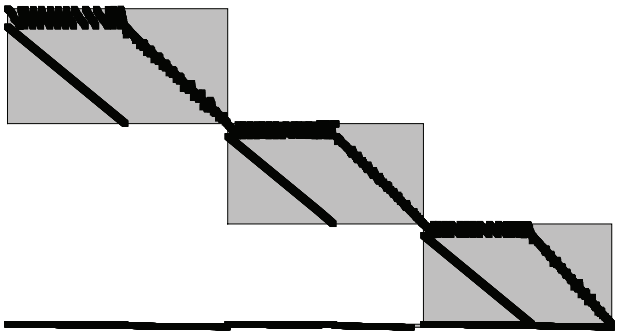
mkc_2_1_2_100000_0.2_0_0.5



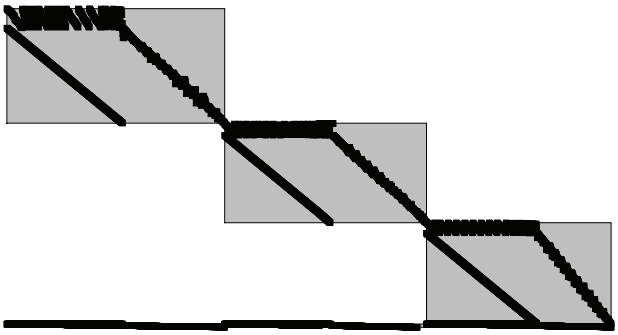
mkc_2_1_2_5_0.2_0_0.5



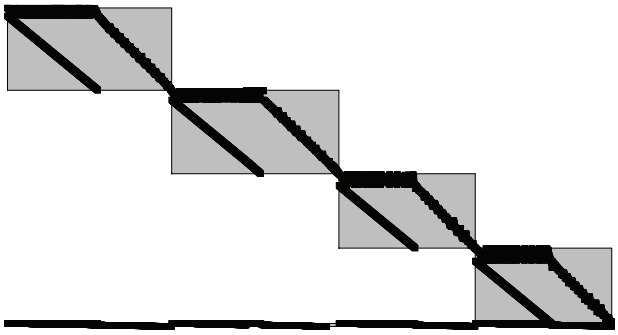
mkc_3_1_2_100000_0.2_0_0.5



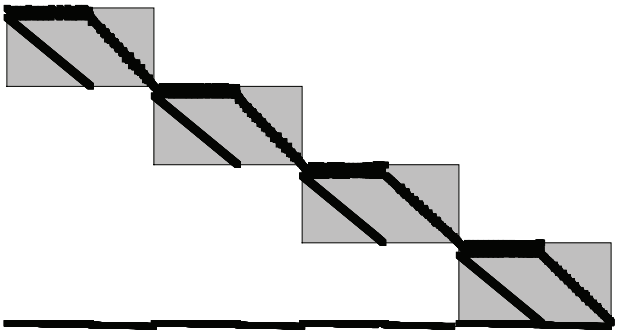
mkc_3_1_2_5_0.2_0_0.5



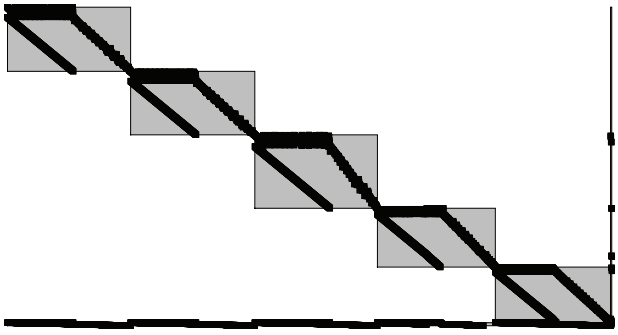
mkc_4_1_2_100000_0.2_0_0.5



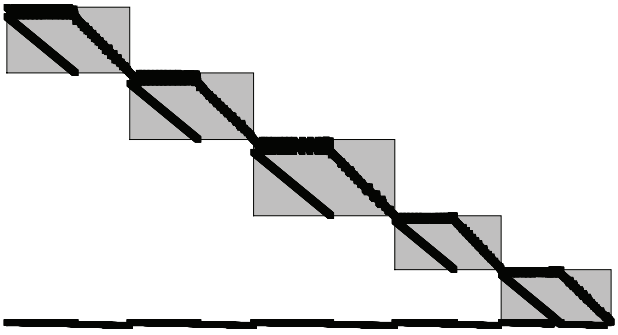
mkc_4_1_2_5_0.2_0_0.5



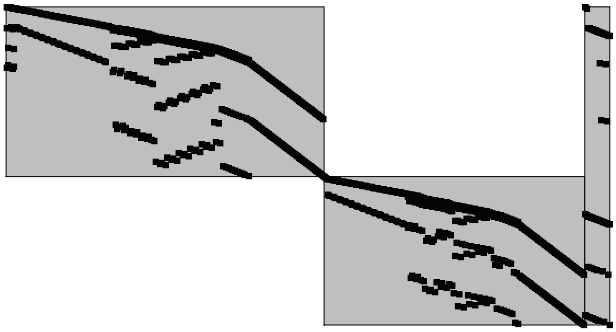
mkc_5_1_2_100000_0.2_0_0.5



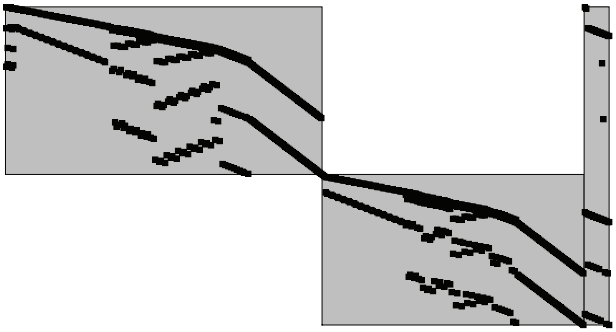
mkc_5_1_2_5_0.2_0_0.5



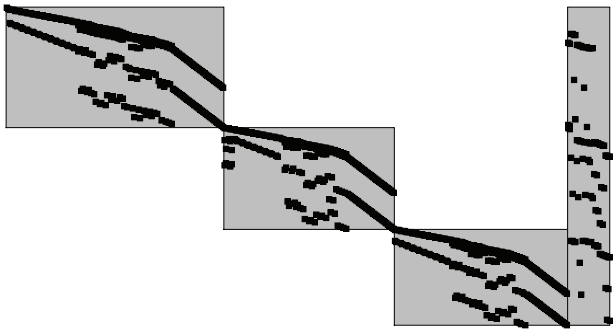
modglob_2_1_2_100000_0.2_0_0.5



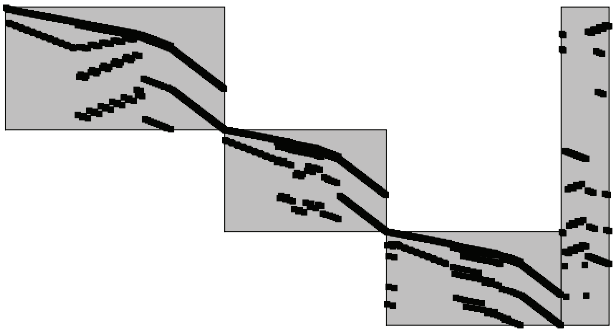
modglob_2_1_2_5_0.2_0_0.5



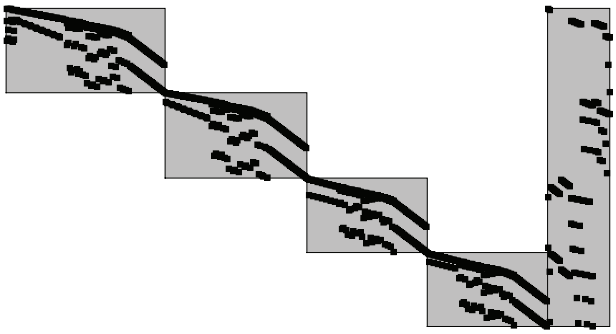
modglob_3_1_2_100000_0.2_0_0.5



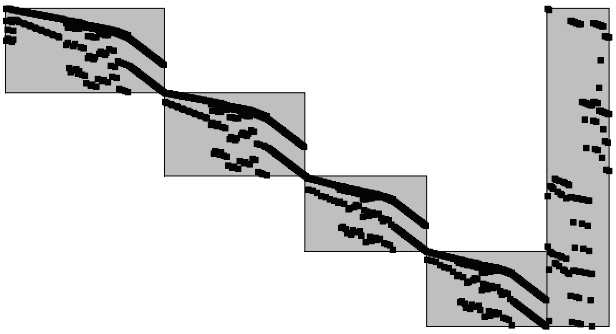
modglob_3_1_2_5_0.2_0_0.5



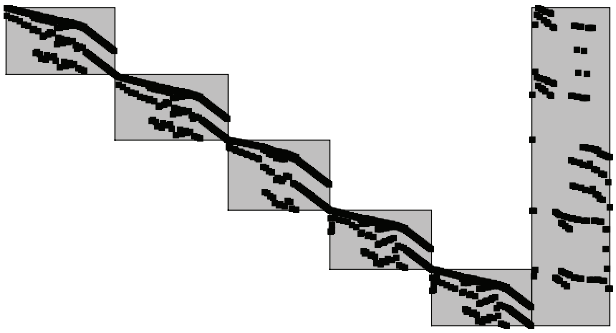
modglob_4_1_2_100000_0.2_0_0.5



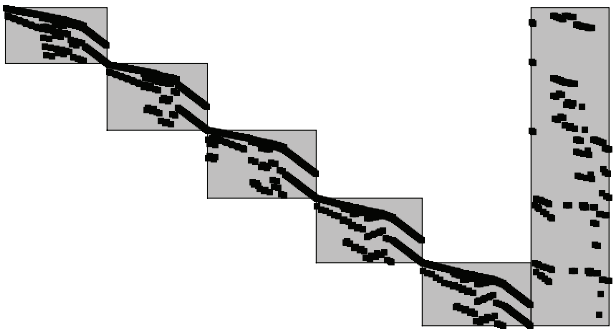
modglob_4_1_2_5_0.2_0_0.5



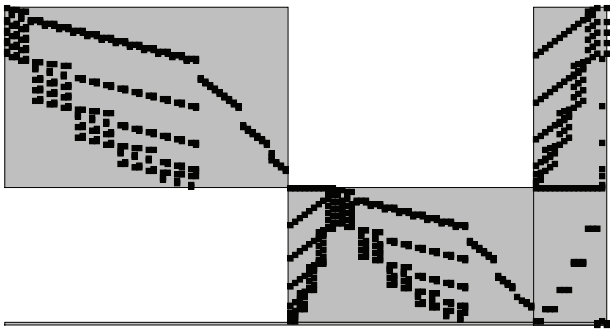
modglob_5_1_2_100000_0.2_0_0.5



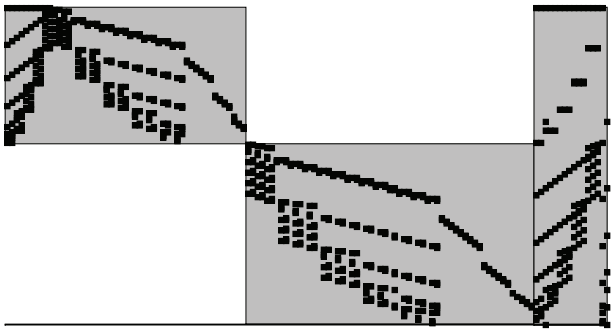
modglob_5_1_2_5_0.2_0_0.5



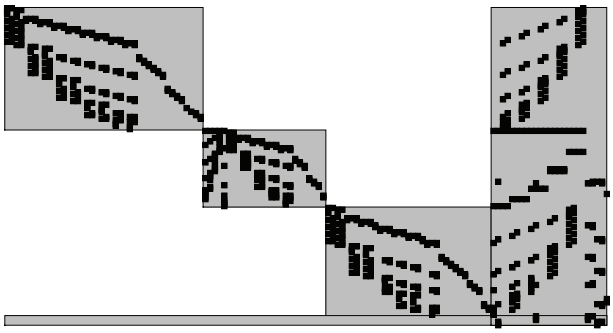
noswot_2_1_2_100000_0.2_0_0.5



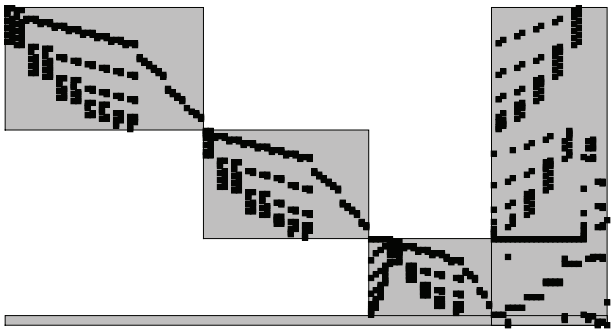
noswot_2_1_2_5_0.2_0_0.5



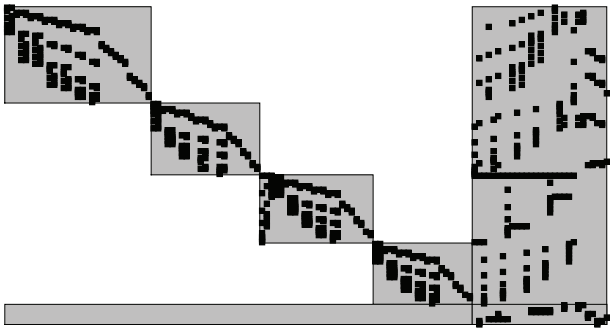
noswot_3_1_2_100000_0.2_0_0.5



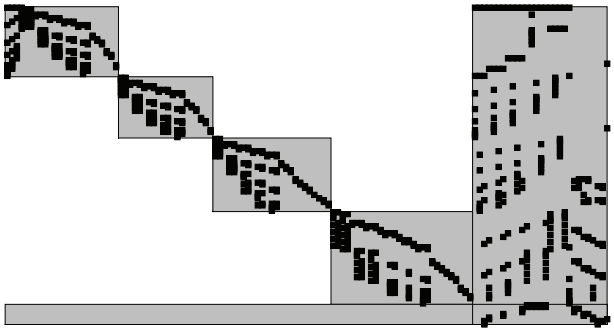
noswot_3_1_2_5_0.2_0_0.5



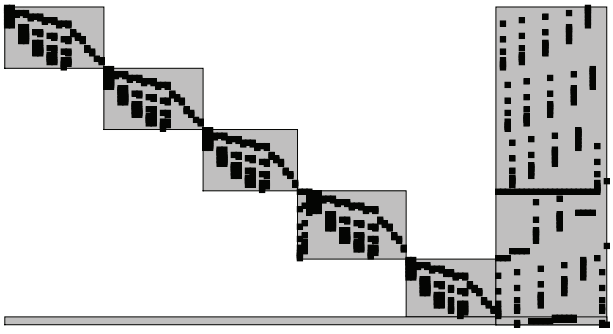
noswot_4_1_2_100000_0.2_0_0.5



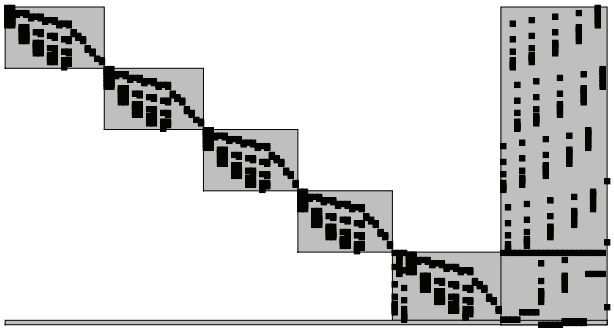
noswot_4_1_2_5_0.2_0_0.5



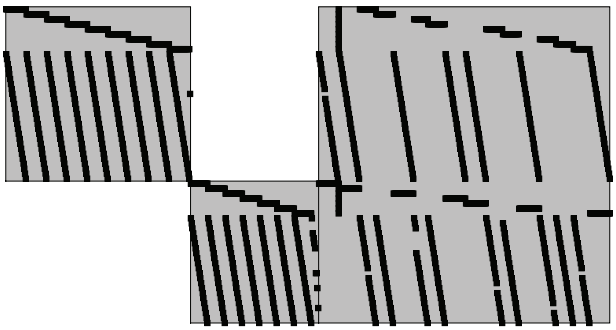
noswot_5_1_2_100000_0.2_0_0.5



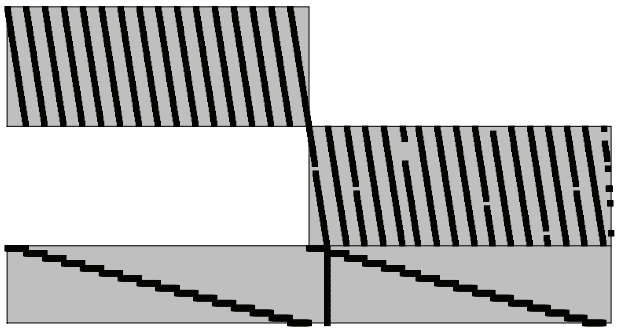
noswot_5_1_2_5_0.2_0_0.5



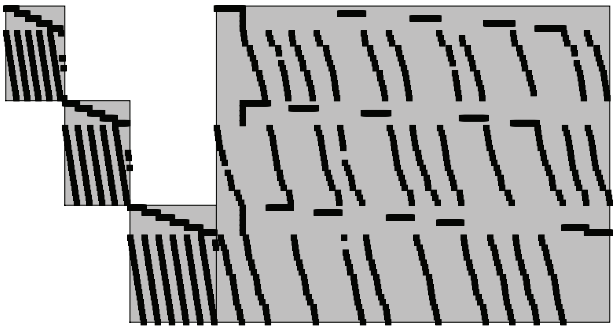
opt1217_2_1_2_100000_0.2_0_0.5



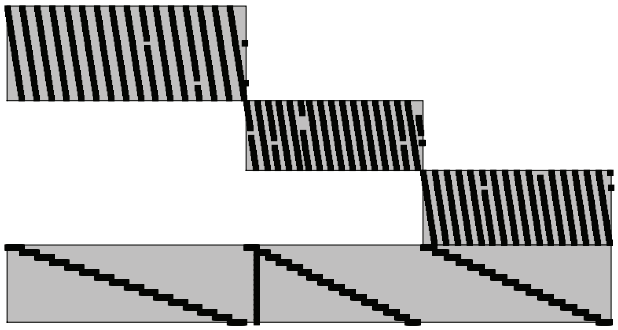
opt1217_2_1_2_5_0.2_0_0.5



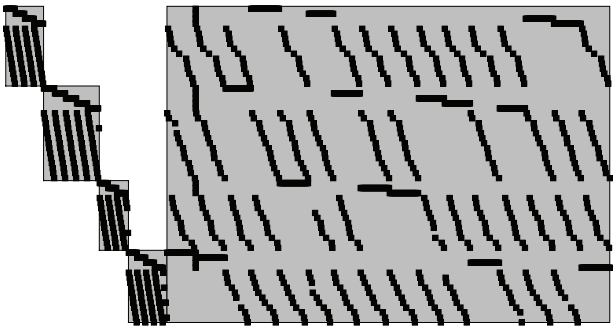
opt1217_3_1_2_100000_0.2_0_0.5



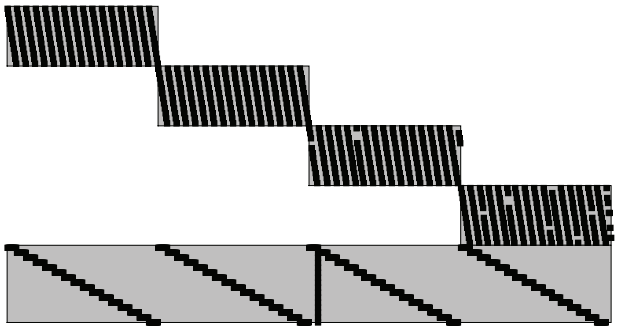
opt1217_3_1_2_5_0.2_0_0.5



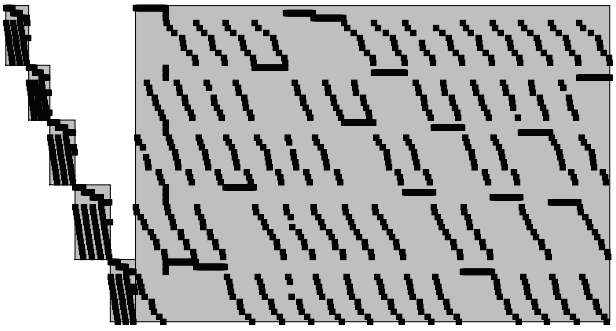
opt1217_4_1_2_100000_0.2_0_0.5



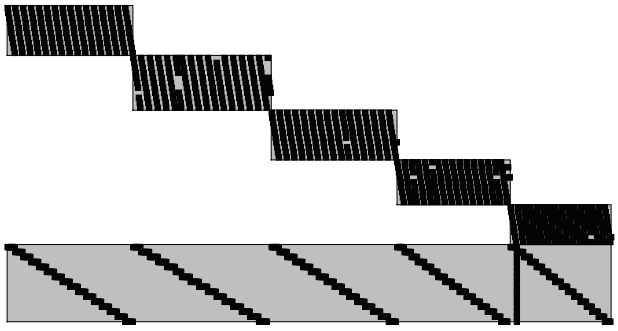
opt1217_4_1_2_5_0.2_0_0.5



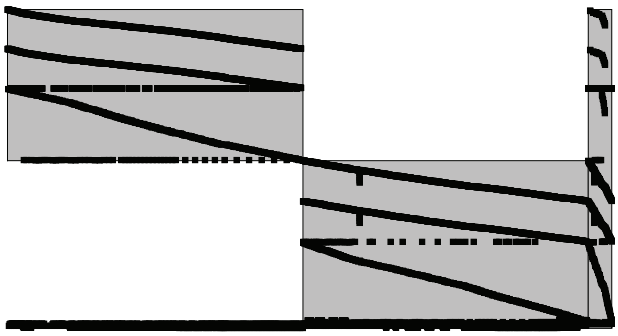
opt1217_5_1_2_100000_0.2_0_0.5



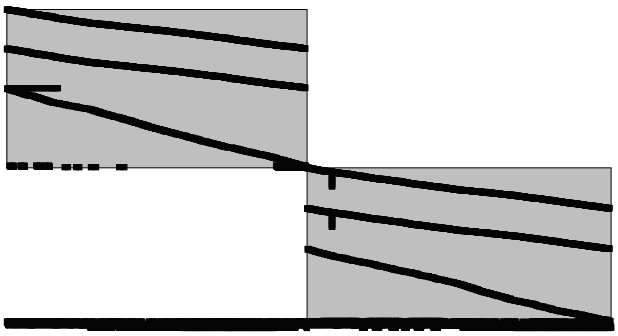
opt1217_5_1_2_5_0.2_0_0.5



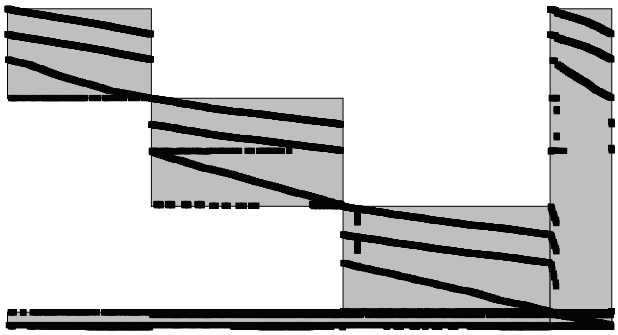
p2756_2_1_2_100000_0.2_0_0.5



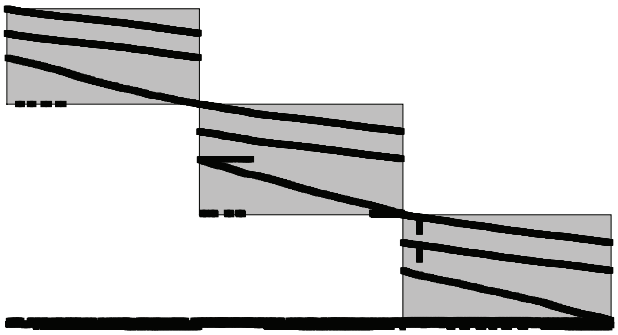
p2756_2_1_2_5_0.2_0_0.5



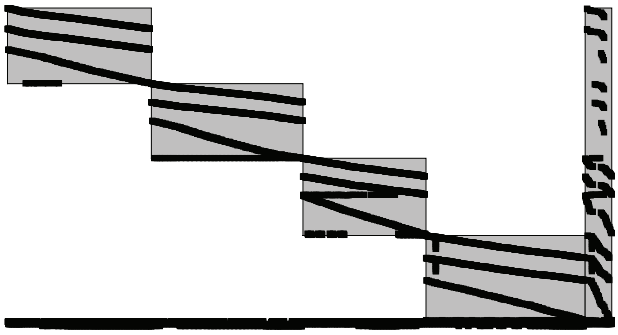
p2756_3_1_2_100000_0.2_0_0.5



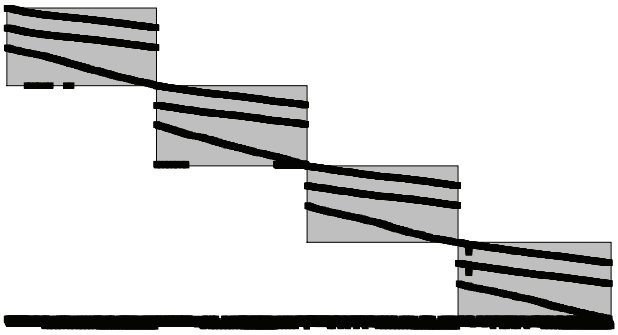
p2756_3_1_2_5_0.2_0_0.5



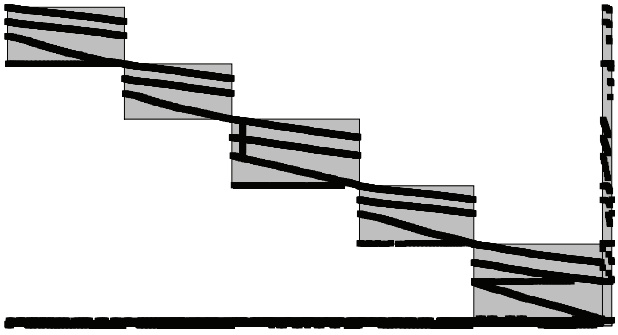
p2756_4_1_2_100000_0.2_0_0.5



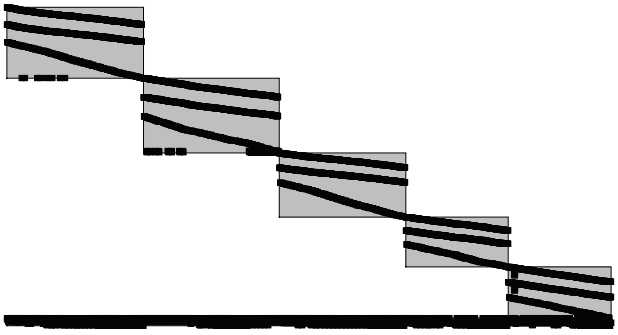
p2756_4_1_2_5_0.2_0_0.5



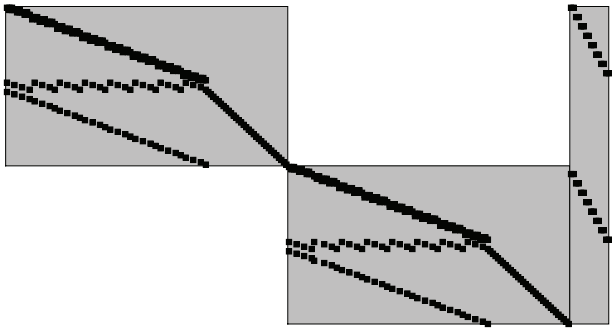
p2756_5_1_2_100000_0.2_0_0.5



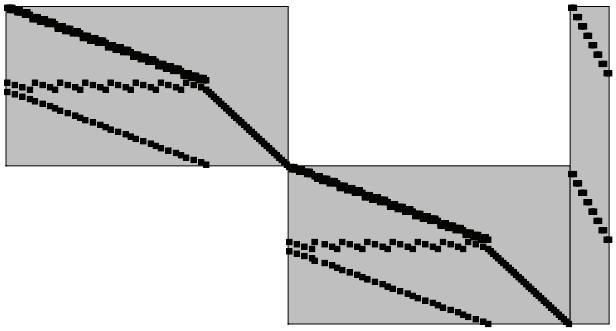
p2756_5_1_2_5_0.2_0_0.5



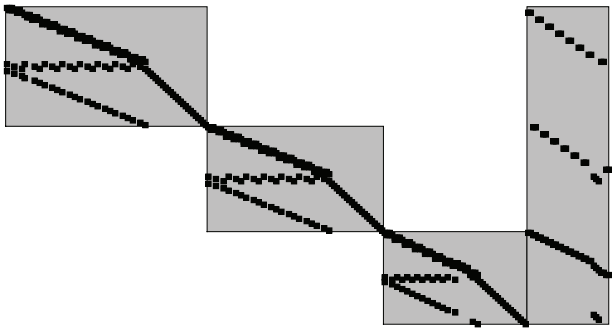
pp08a_2_1_2_100000_0.2_0_0.5



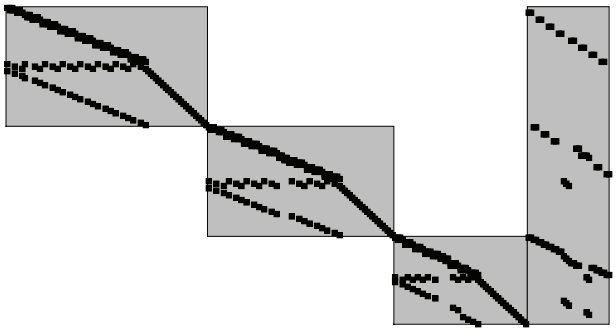
pp08a_2_1_2_5_0.2_0_0.5



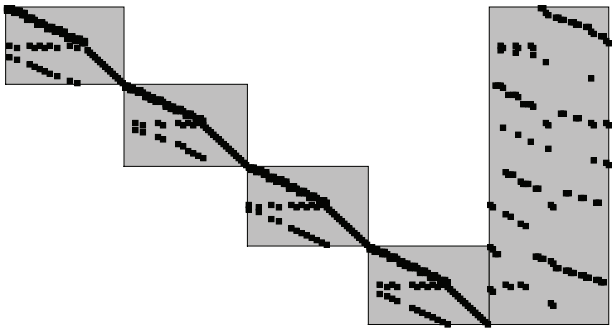
pp08a_3_1_2_100000_0.2_0_0.5



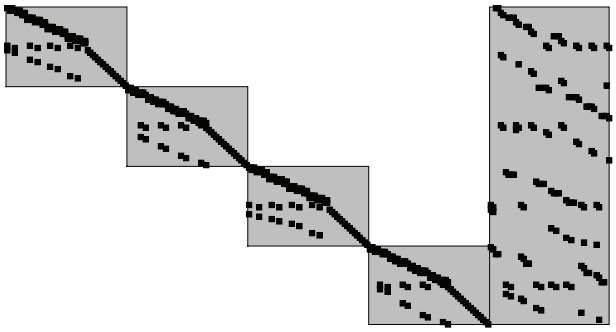
pp08a_3_1_2_5_0.2_0_0.5



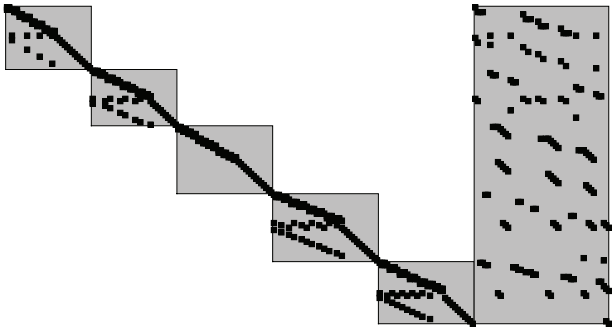
pp08a_4_1_2_100000_0.2_0_0.5



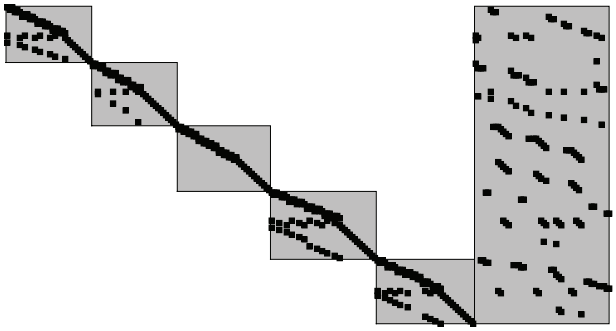
pp08a_4_1_2_5_0.2_0_0.5



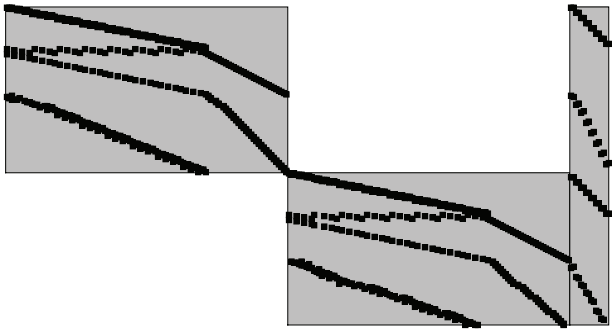
pp08a_5_1_2_100000_0.2_0_0.5



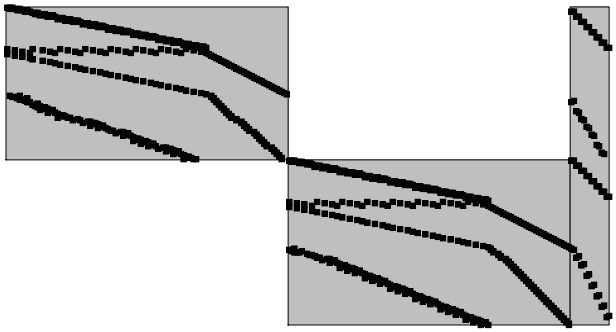
pp08a_5_1_2_5_0.2_0_0.5



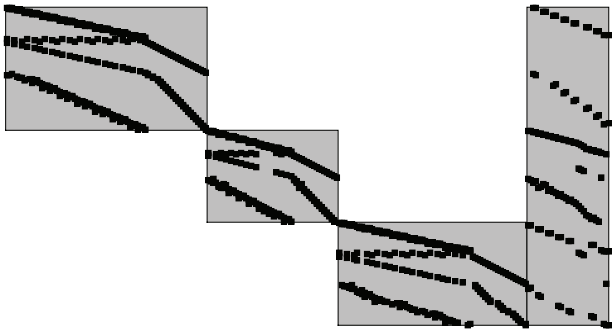
pp08aCUTS_2_1_2_100000_0.2_0_0.5



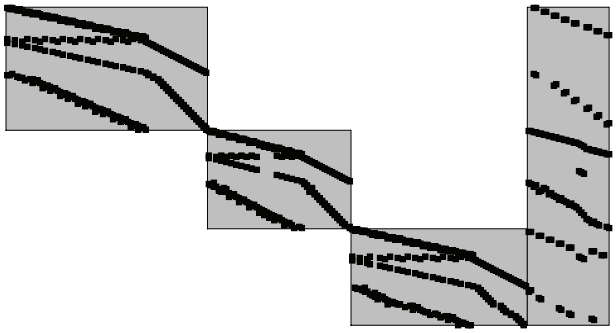
pp08aCUTS_2_1_2_5_0.2_0_0.5



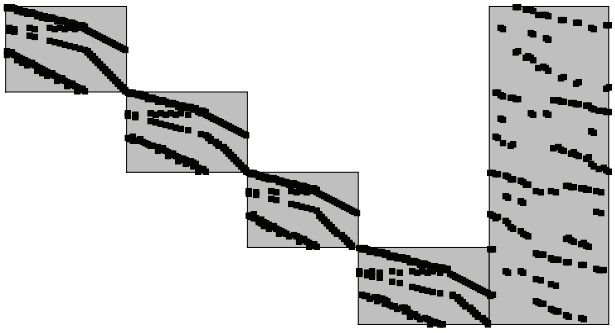
pp08aCUTS_3_1_2_100000_0.2_0_0.5



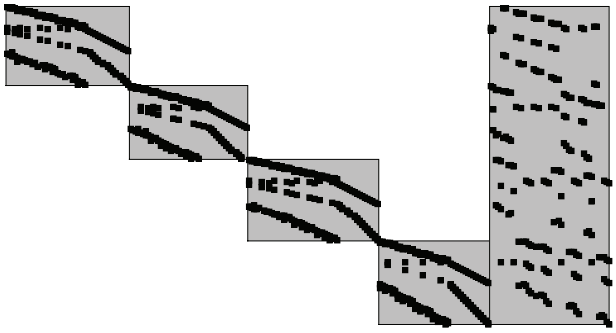
pp08aCUTS_3_1_2_5_0.2_0_0.5



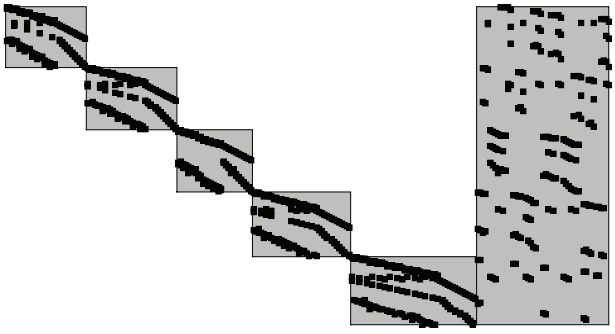
pp08aCUTS_4_1_2_100000_0.2_0_0.5



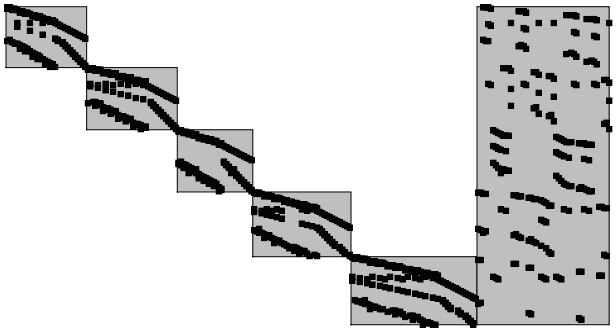
pp08aCUTS_4_1_2_5_0.2_0_0.5



pp08aCUTS_5_1_2_100000_0.2_0_0.5

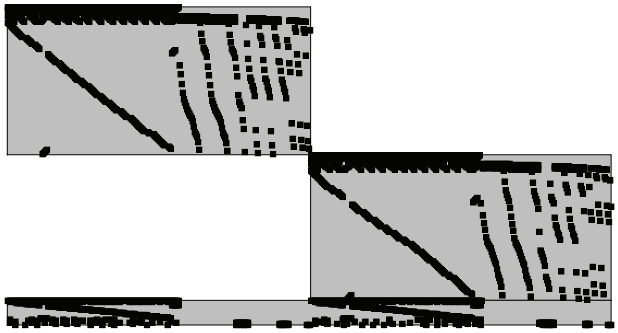


pp08aCUTS_5_1_2_5_0.2_0_0.5

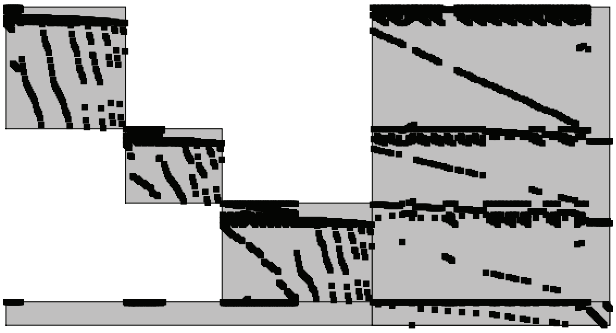


rout_2_1_2_100000_0.2_0_0.5

rout_2_1_2_5_0.2_0_0.5

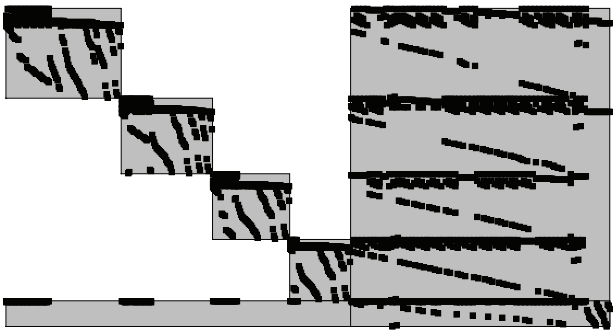


rout_3_1_2_100000_0.2_0_0.5



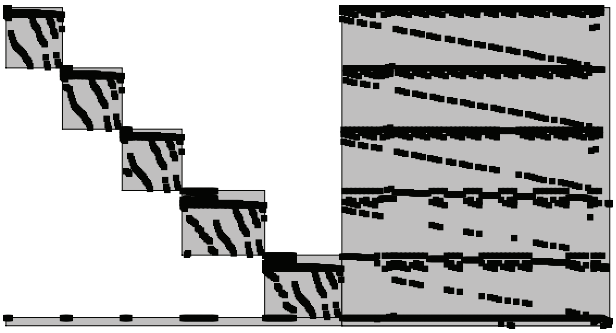
rout_3_1_2_5_0.2_0_0.5

rout_4_1_2_100000_0.2_0_0.5

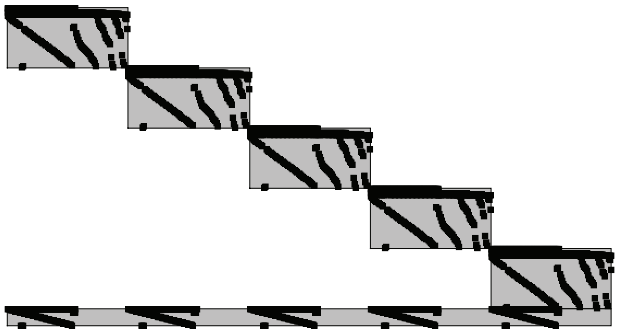


rout_4_1_2_5_0.2_0_0.5

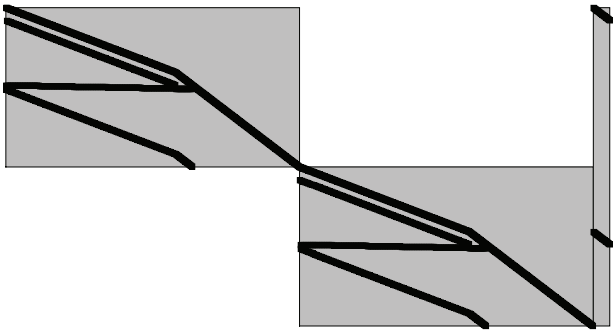
rout_5_1_2_100000_0.2_0_0.5



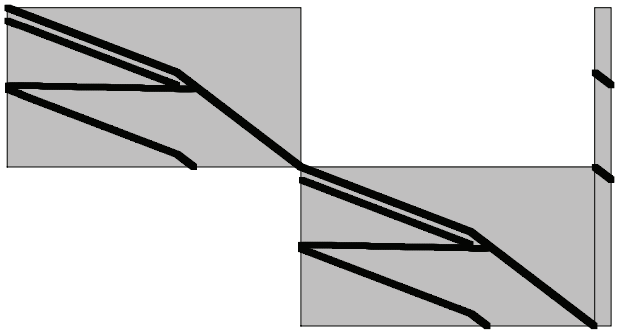
rout_5_1_2_5_0.2_0_0.5



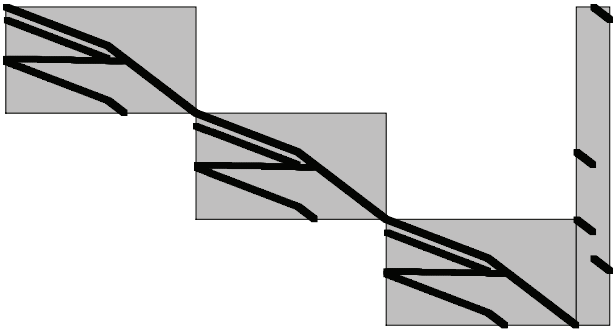
set1ch_2_1_2_100000_0.2_0_0.5



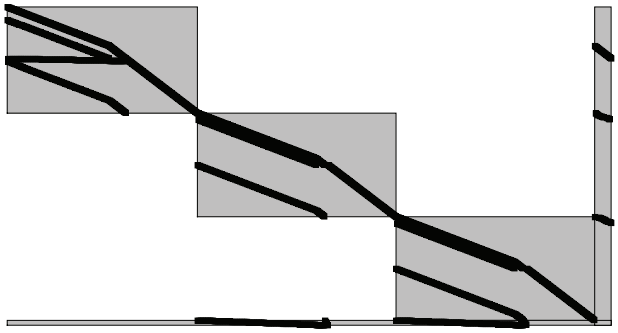
set1ch_2_1_2_5_0.2_0_0.5



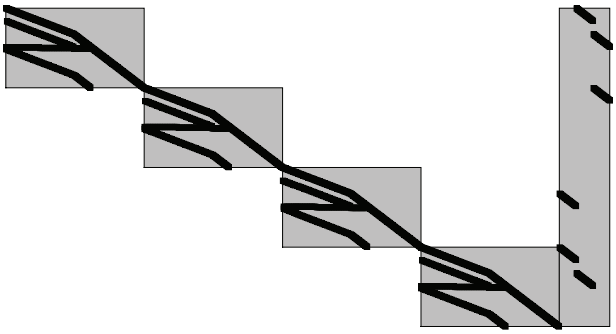
set1ch_3_1_2_100000_0.2_0_0.5



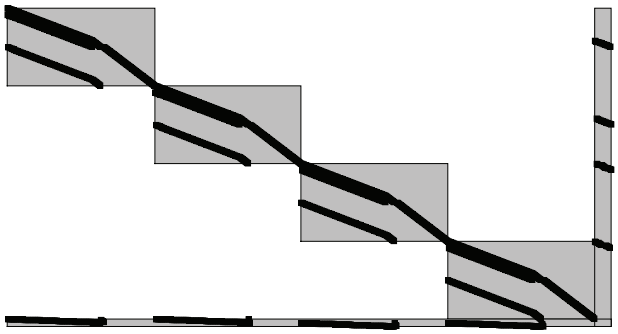
set1ch_3_1_2_5_0.2_0_0.5



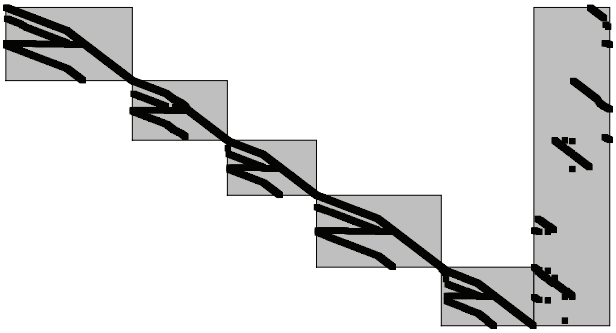
set1ch_4_1_2_100000_0.2_0_0.5



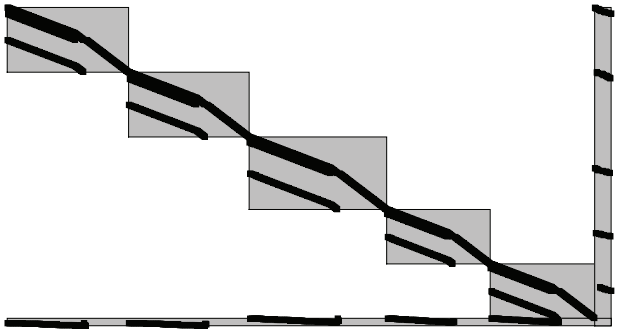
set1ch_4_1_2_5_0.2_0_0.5



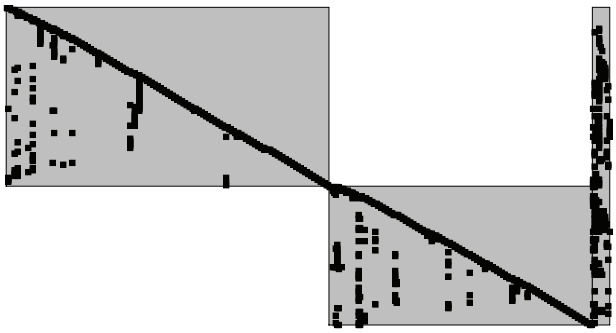
set1ch_5_1_2_100000_0.2_0_0.5



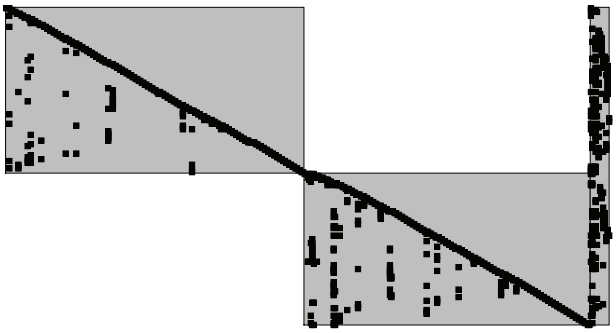
set1ch_5_1_2_5_0.2_0_0.5



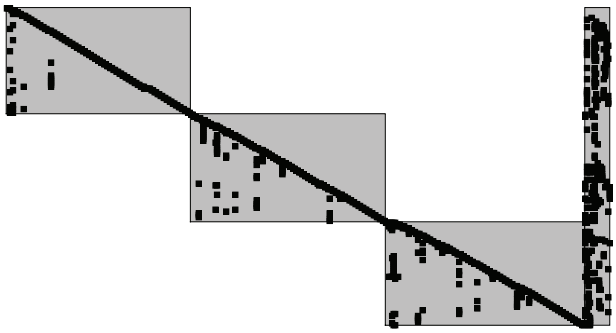
timtab1_2_1_2_100000_0.2_0_0.5



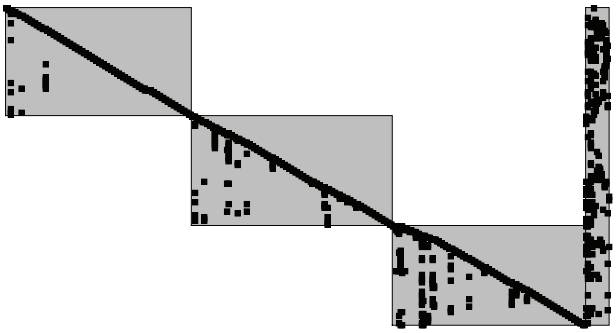
timtab1_2_1_2_5_0.2_0_0.5



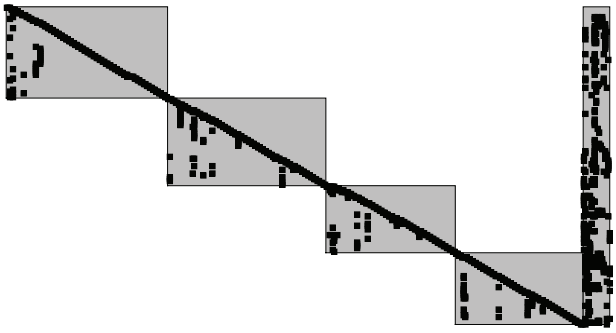
timtab1_3_1_2_100000_0.2_0_0.5



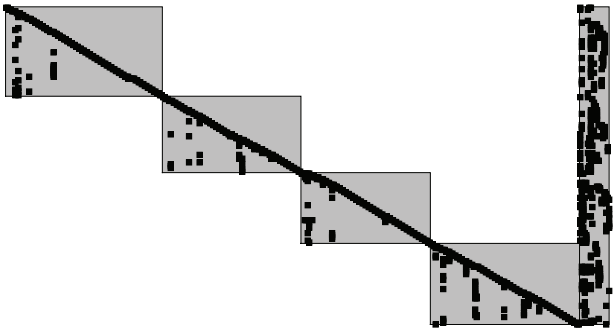
timtab1_3_1_2_5_0.2_0_0.5



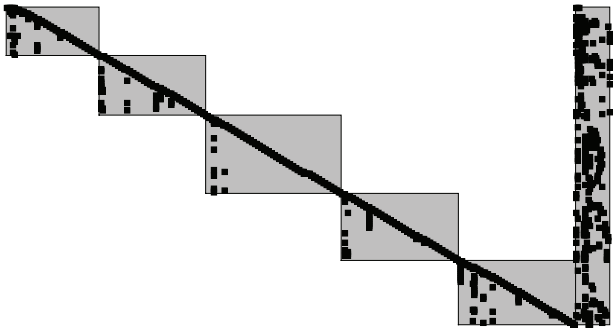
timtab1_4_1_2_100000_0.2_0_0.5



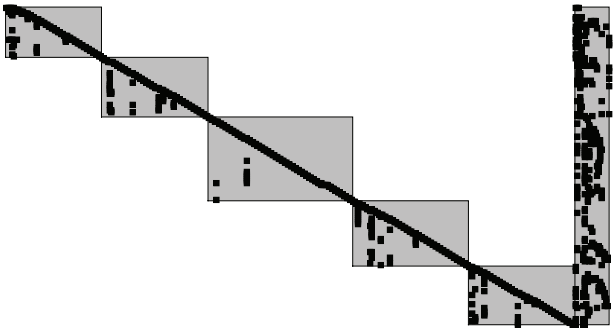
timtab1_4_1_2_5_0.2_0_0.5



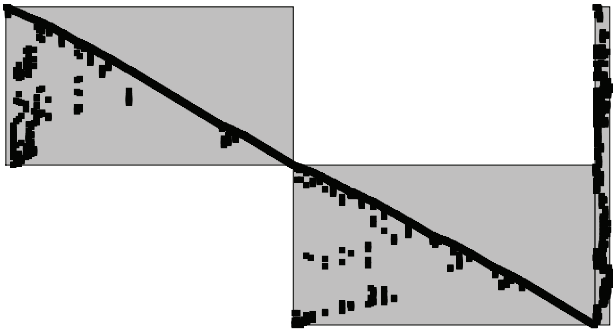
timtab1_5_1_2_100000_0.2_0_0.5



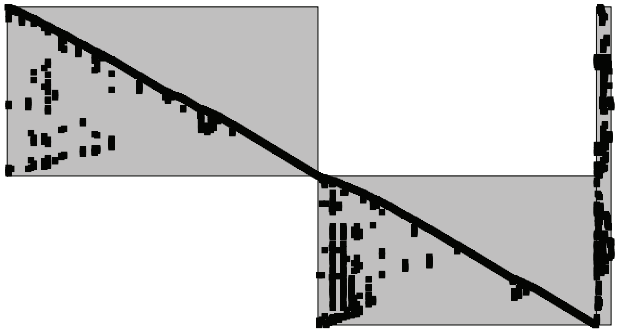
timtab1_5_1_2_5_0.2_0_0.5



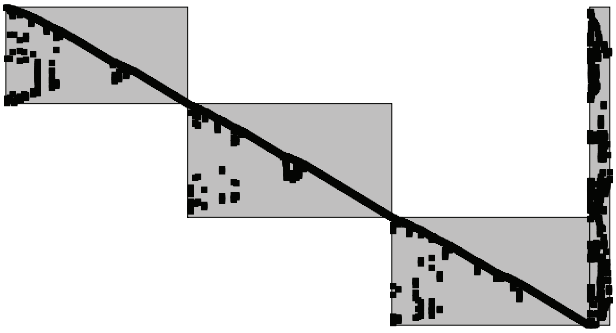
timtab2_2_1_2_100000_0.2_0_0.5



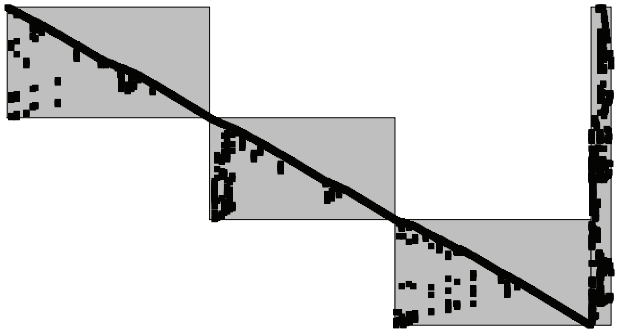
timtab2_2_1_2_5_0.2_0_0.5



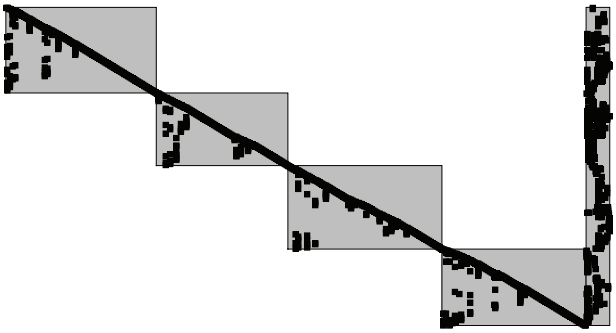
timtab2_3_1_2_100000_0.2_0_0.5



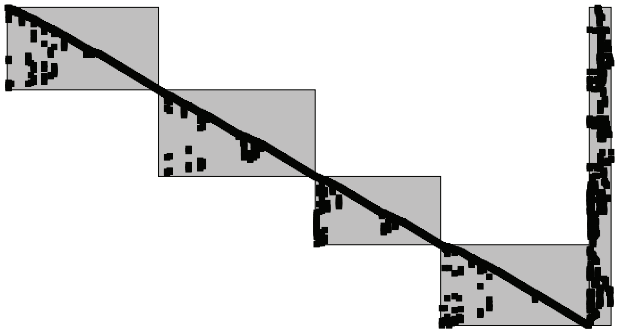
timtab2_3_1_2_5_0.2_0_0.5



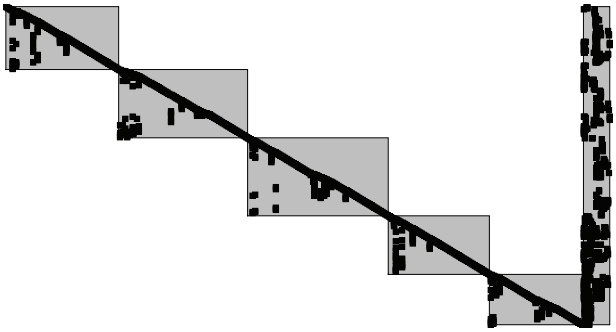
timtab2_4_1_2_100000_0.2_0_0.5



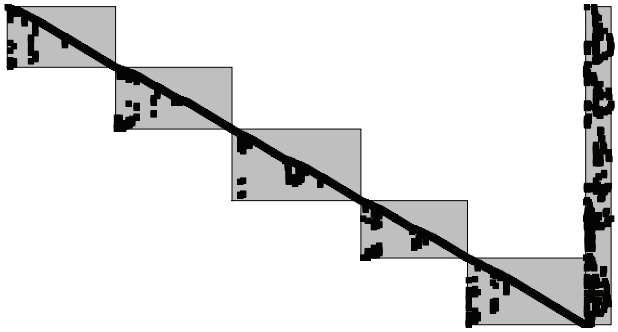
timtab2_4_1_2_5_0.2_0_0.5



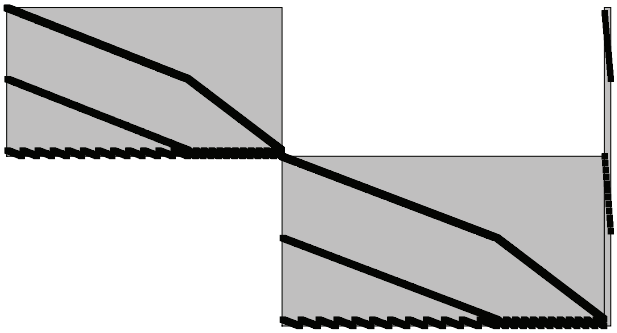
timtab2_5_1_2_100000_0.2_0_0.5



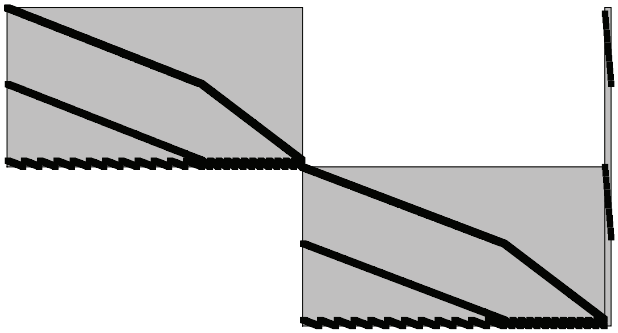
timtab2_5_1_2_5_0.2_0_0.5



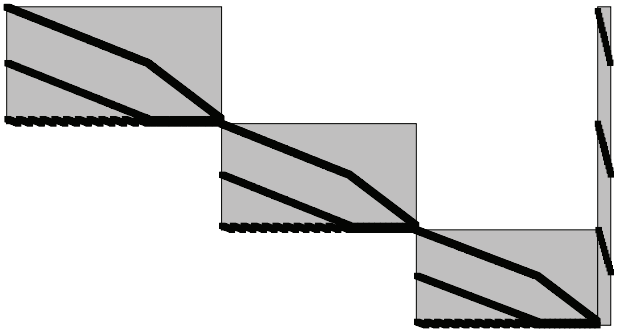
tr12-30_2_1_2_100000_0.2_0_0.5



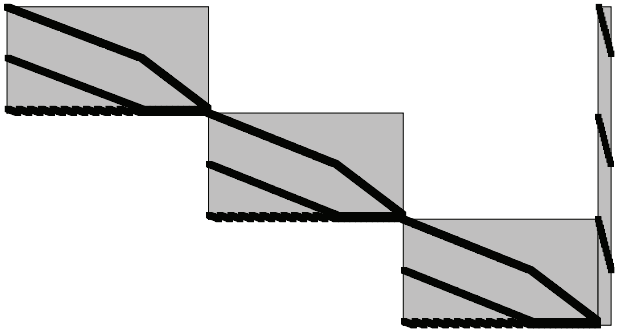
tr12-30_2_1_2_5_0.2_0_0.5



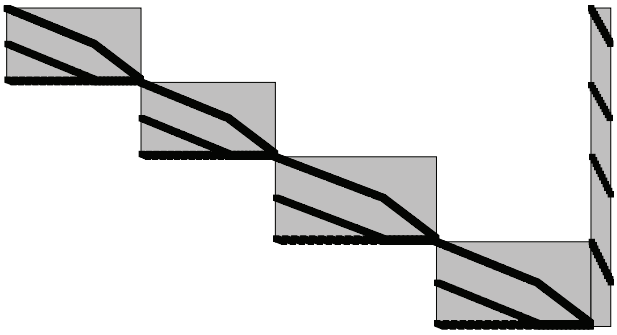
tr12-30_3_1_2_1000000_.2_0_0.5



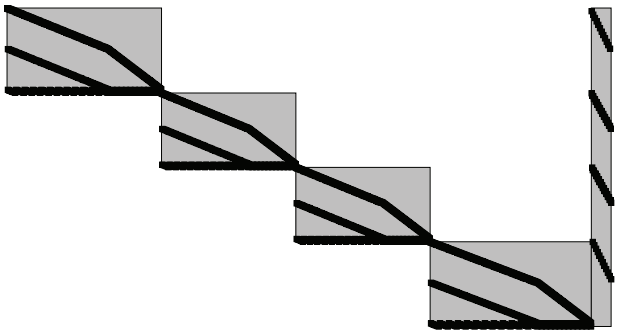
tr12-30_3_1_2_5_0.2_0_0.5



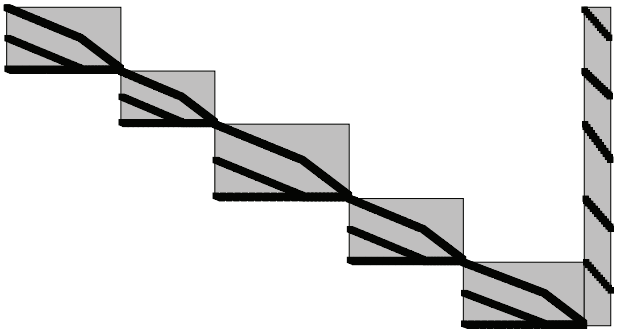
tr12-30_4_1_2_100000_0.2_0_0.5



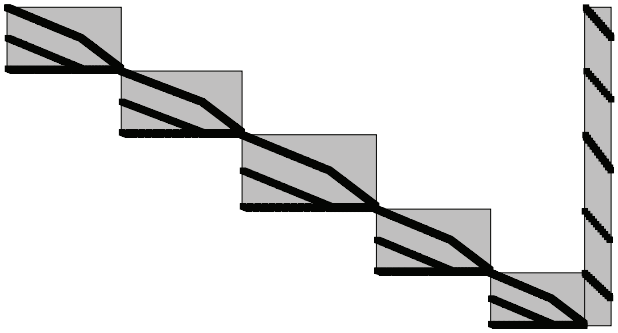
tr12-30_4_1_2_5_0.2_0_0.5



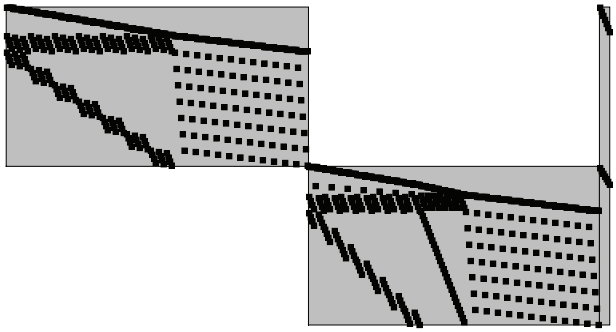
tr12-30_5_1_2_100000_0.2_0_0.5



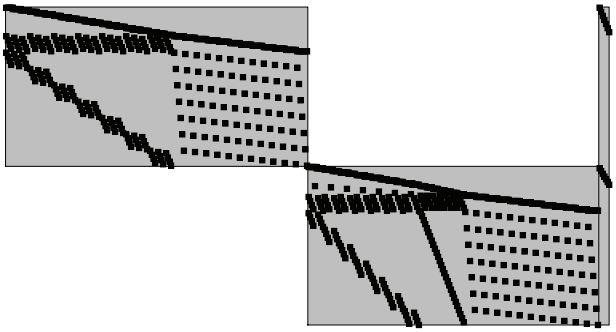
tr12-30_5_1_2_5_0.2_0_0.5



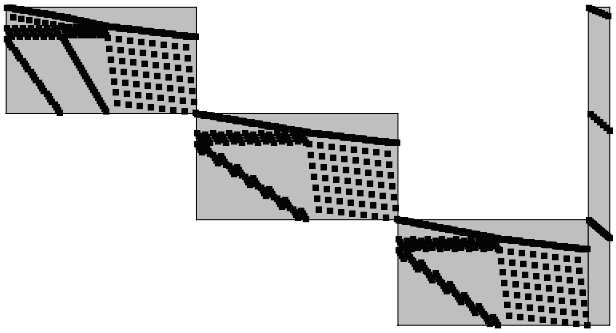
vpm2_2_1_2_100000_0.2_0_0.5



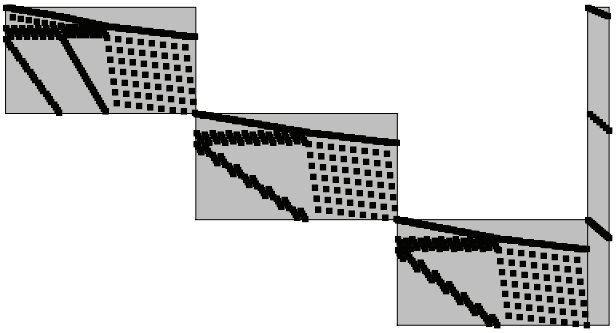
vpm2_2_1_2_5_0.2_0_0.5



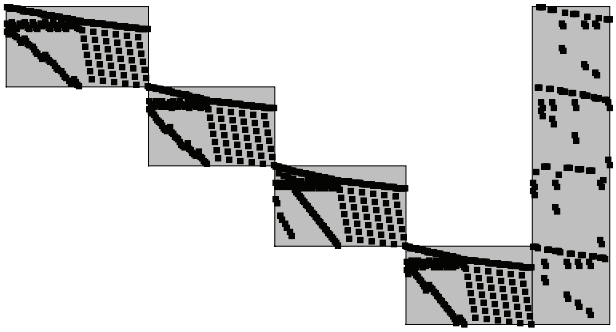
vpm2_3_1_2_100000_0.2_0_0.5



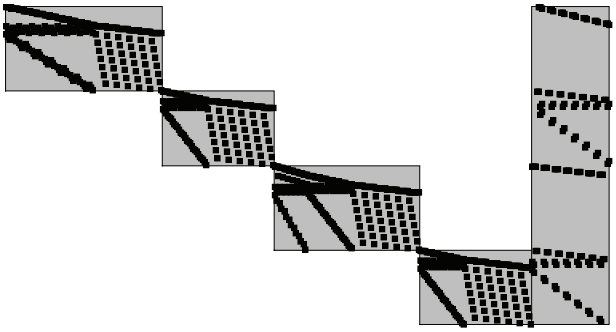
vpm2_3_1_2_5_0.2_0_0.5



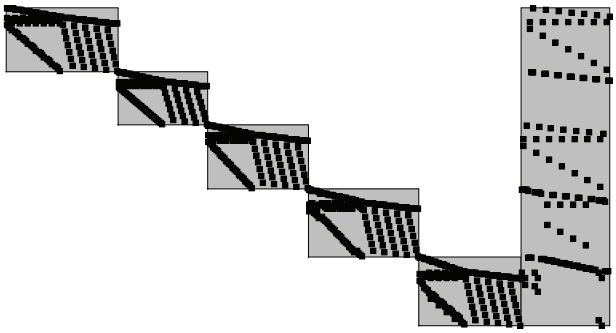
vpm2_4_1_2_100000_0.2_0_0.5



vpm2_4_1_2_5_0.2_0_0.5



vpm2_5_1_2_100000_0.2_0_0.5



vpm2_5_1_2_5_0.2_0_0.5