Approximation of geometric set packing and hitting set problems

Approximation of Geometric Set Packing and Hitting Set Problems

Approximation of Geometric Set Packing and Hitting Set Problems

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit Maastricht,
op gezag van de Rector Magnificus,
Prof. dr. A.C. Nieuwenhuijzen Kruseman,
volgens het besluit van het College van Decanen,
in het openbaar te verdedigen
op woensdag 24 september 2003 om 14.00 uur

door

Sofia Kovaleva

Contents

Contents

Chapter 1

Introduction

This thesis gives an illustration of the design of approximation algorithms for some combinatorial optimization problems, notably *set packing* and *hitting set* problems.

Set packing and hitting set are two of the most basic combinatorial optimization problems. In the set packing problem (SP), as well as in the hitting set problem (HS), we are given a ground set $\mathcal{E}$ and a collection $\mathcal{S}$ of subsets of $\mathcal{E}$. In the set packing problem one aims to find a collection of maximum cardinality of pairwise non-overlapping subsets from $\mathcal{S}$. For the hitting set problem the task is to identify a subset of $\mathcal{E}$ of smallest cardinality which intersects (or "hits") each subset of $\mathcal{S}$. The weighted version of the set packing problem arises when given positive weights for the subsets in $\mathcal{S}$, one aims to maximize the total weight of the selected subsets. In the weighted version of the hitting set problem the weights are specified for the elements of $\mathcal{E}$ and the objective is to minimize the total weight of the identified subset.

By an easy transformation one can show that two well-known problems, *set cover* and *independent set*, are equivalent to HS and SP respectively. Furthermore, an important type of integer programming problem, $\max\{wx|Ax \leq (\geq)e\}$, where A is a 0,1-matrix, e is a unit vector and vector w is nonnegative, can be represented as SP (HS). Surveys on these problems can be found in Balas & Padberg [7], Garfinkel & Nemhauser [29], Trotter [49], Cornuejols [19] and Schrijver [45]. See also Vemuganti [51] for a survey on applications of set packing and set cover problems.

In this thesis we study several special cases of SP and HS. Since each of the problems we investigate is NP-hard, there is little hope to find a polynomial algorithm that always solves them to optimality. In general, different

methods have been developed in combinatorial optimization to deal with NP-hard problems. The two basic types of these methods are: *exact algorithms*, which always solve a problem to optimality, but may have an exponential running time for some instances, and *heuristic algorithms*, which always output feasible solutions, although not necessarily optimal. Heuristic algorithms come in many different types. They may or may not run in polynomial time. In this thesis we focus on one type of them, *approximation algorithms*. Approximation algorithms are heuristics with a polynomial running time and a *worst-case performance ratio*, which is the worst-case ratio between the value of the solution produced by the algorithm and the optimum (see e.g. Hochbaum [32]). The worst-case ratio gives a guarantee for the quality of the solutions and thus serves as a criterion of the quality of the algorithm. In addition to this the design and analysis of approximation algorithms provides an insight in the structure of the problem that can later be used in the design of other solution methods.

Problems. As was mentioned above, the problems we study in this thesis are special cases of the set packing and hitting set problems. It is well known that in general SP and HS can not be approximated within a constant factor unless $\mathcal{P} = \mathcal{NP}$ (see Hastad [30] and Raz & Safra [44]). However, if the subsets in the input are known to possess a certain structure these problems may become approximable within a constant factor or even polynomially solvable. We are interested in the cases when the subsets possess a certain geometric structure, explicitly given in the input.

Let us give some examples. In the first example the ground set consists of all the points on the real line and the subsets are intervals. Both the problems SP and HS reduce then to the maximum weight independent set and minimum weight clique cover problems in an interval graph, which can be solved in polynomial time (see e.g. Golumbic [28]).

In the second example, the ground set consists of all the points on the axes x and y. Given a collection of axis-parallel rectangles, we construct a collection of subsets as follows: each subset is formed by the points of the projections of some rectangle on both the axes. Then two subsets intersect iff the projections of the corresponding rectangles on at least one of the axes overlap. Figure 1.1 gives an illustration of the set packing and hitting set problems in this setting. It is known that these special cases of SP and HS are both MAX SNP-hard, which means that they can not be approximated with an arbitrary good precision in polynomial time unless $\mathcal{P} = \mathcal{NP}$ (see Spieksma [47] and Section 4.3 of this thesis). However there exist approxi-

mation algorithms for these problems, one with a performance ratio of 4 for SP (Bar-Yehuda et al. [10]) and one with a ratio of 2 for HS (Gaur et al. [26]).

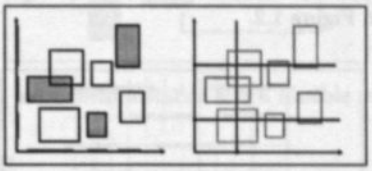

Figure 1.1: An illustration of the setting described in the second example, and optimal solutions to the problems SP (left) and HS (right).

For the third example consider a ground set consisting of all the points in the plane, and axis-parallel rectangles as the subsets from the collection. Then two such subsets intersect iff the corresponding rectangles overlap. The set packing problem defined on this input is NP-hard, and so far the question about existence of a constant factor approximation algorithm remains open.

We now introduce the three problems which we study in this thesis. The first two, *job interval packing* (JIP) and *job interval stabbing* (JIS), are close to the problems described in the second example, but, on one hand, the structure of subsets is slightly more restricted, and, on the other hand, we consider generalizations of the weighted SP and HS, which involve such parameters as "capacities" associated with elements of $\mathcal{E}$ and "demands" attached to the subsets in $\mathcal{S}$. A collection of subsets is now a feasible solution to SP if the number of subsets (with multiplicities) sharing an element does not exceed its capacity. A subset of $\mathcal{E}$ is a feasible solution to HS if the number of times it hits each subset in $\mathcal{S}$ equals at least the subset's demand. When the capacities and demands are unit, we obtain the regular weighted SP and HS.

The third problem we consider, *rectangle packing problem with a bounded height ratio* (RP^B), is a restricted case of the set packing problem described in the third example.

Job interval packing problem (JIP): given is a grid consisting of columns and rows, and a set of intervals lying on the rows of the grid. A positive integral capacity is associated with each column, row and interval, and each interval is given a positive integral weight. The task is to assign integral

multiplicities to the intervals, not exceeding their capacities, maximizing the total weight, such that the sum of multiplicities of the intervals intersecting any column (or lying on any row) does not exceed the corresponding column (row) capacity. An example of an instance of JIP together with an optimal solution is shown in Figure 1.2.

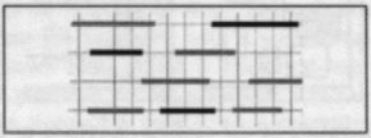

Figure 1.2: An instance of JIP with unit weights and capacities, and its optimal solution.

Job interval stabbing problem (JIS): given is the same input as for JIP with the only difference that the column, row and interval capacities are now considered as weights and the interval weights are now referred to as interval demands. The task now is to specify integral multiplicities for the columns, rows and intervals, minimizing the total weight, such that for each *interval the sum of the multiplicities of the columns stabbing the interval*, the multiplicity of the row where the interval lies and the own multiplicity of the interval equal at least its demand. See Figure 1.3 for an illustration.

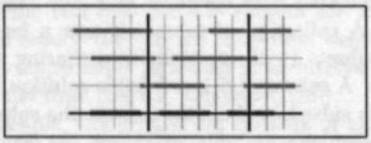

Figure 1.3: An instance of JIS with unit weights and demands and its optimal solution.

Rectangle Packing problem with a bounded height ratio ($\mathbf{RP}^B$): Given is a set of weighted axis parallel closed rectangles in the plane, such that the ratio between the largest height of a rectangle and the smallest is bounded by B. The task is to find a maximum weight subset of non-overlapping rectangles. See Figure 1.4 for an illustration.

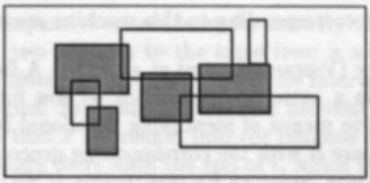

Figure 1.4: RP^B: a problem instance and a feasible solution (shaded).

Applications

Let us describe some applications of the problems JIP, JIS and RP^B.

Applications of JIP

Machine scheduling (Bar-Noy et al [8]). Given are k parallel (identical or unrelated) machines and a number of jobs. Each job may be scheduled on one of the machines in one of the prescribed time intervals and, if scheduled, yields a given profit, depending on when it was scheduled. In case of unrelated machines the intervals and profits are machine dependent. The goal is to maximize the profit by scheduling a subset of jobs to the machines.

The case of identical machines can be represented as JIP (from here the name "job interval packing") with column capacities all equal to k and row and interval capacities all equal to 1. Indeed, let the columns correspond to the end-points of the intervals and the rows to the machines. All the time intervals belonging to one job lie on one corresponding row. In a feasible solution to JIP at most one interval is selected for each job and at most k intervals intersect each column, which due to the construction of the columns implies that at most k time intervals overlap at each point in time. This may be easily translated into a feasible solution to the scheduling problem since one can assign now the jobs to the k machines by coloring the underlying interval graph in k colors, which can be done in polynomial time.

In case of unrelated machines we have a collection of intervals for each job - machine pair. This problem may be reduced to JIP with unit column, row and interval capacities as follows. The instance of JIP is constructed by the same principle as described above with the difference that the timeline of each of the k machines is projected into a separate part of the real line

and all the intervals corresponding to this machine are placed in this part.

Molecular biology (Veeramachaneni et al. [52]). A fundamental problem in biology is to gain a better understanding of how functions are encoded in genes. An effective means of identifying functional regions in a genomic sequence is to compare it with the corresponding genomic region of another species. The Consensus Sequence Reconstruction (CSR) problem is encountered as a subproblem: given two sets $\mathcal{H}$ and $\mathcal{M}$ of DNA fragments (say, one set taken from human DNA and the other from mouse DNA), determine as much information about the order and orientation of the fragments as possible. If $\mathcal{H}$ consists of a single (long) fragment H, the task is to align fragments in $\mathcal{M}$ with substrings of H such that different fragments are aligned with non-overlapping substrings. The goal is to maximize the sum of alignment scores. Instances of this problem can be viewed as instances of JIP with unit column, row and interval capacities: each fragment in $\mathcal{M}$ corresponds to a row, and the substrings of H with which the fragment can be aligned are the intervals.

PCB manufacturing (Spieksma [47]). In printed circuit board (PCB) manufacturing, preparing the production process requires placing so-called *feeders* on a feeder rack. The feeders deliver the components that are to be placed on prespecified locations on the board. Each feeder occupies a certain (small) number of consecutive slots in the rack. Every slot can hold at most one feeder. There are restrictions on the placement of feeders, i.e., each feeder can be placed only in a subset of all possible positions. Given a set of feeders, each with a list of admissible placements, it is desirable to place as many feeders as possible onto admissible positions on the feeder rack. If feeders differ in importance, it is also meaningful to assume that each feeder has a certain weight (priority) and to try to maximize the total weight of the feeders that are placed on the rack.

Viewed as an instance of JIP with unit capacities, the feeders correspond to rows and the admissible positions correspond to intervals.

Caching (Torng [50], Erlebach & Spieksma [21]). A cache is a small, fast memory that can temporarily store arbitrary memory items in order to allow the CPU to access them faster than in main memory. For the purpose of analyzing cache performance we view the execution of a program as a sequence of accesses to memory items. When a memory item x is accessed and does not yet reside in the cache, it can be (but does not have to be; we allow cache bypassing) brought into the cache. If x is still in the cache when it is accessed a second time later on, this is called a *cache hit*. With every cache

hit the cost for an expensive access to main memory is avoided. We view the period between two accesses to the same item x as an interval on the real line. So, at the time of an access at most one interval ends and at most one new interval begins. Selecting an interval i between two accesses to the same item x means that x is brought into (or remains in) the cache at the beginning of i and stays there until the end of i, which is the moment of the next access to x, thus yielding a cache hit.

Consider the case that there can be restrictions on the cache locations available to an item (e.g., as in t-way skewed associative cache, see Seznec [46]). For every memory item x, there is a number of admissible locations in the cache where the item can be stored. The problem of maximizing the number of cache hits can then be modeled as an instance of JIP with unit demands and unit weights as follows: project the timelines of all cache locations onto disjoint parts of the real line and add, for every period between consecutive accesses to x, a row and place on it an interval in those parts of the real line that correspond to cache locations that are admissible for x.

Applications of JIS

Applications of the job interval stabbing problem seem less abundantly present in literature; however, the following type of situation leads naturally to instances of JIS with unit interval demands. Each of m items (patients to receive treatment, products to undergo chemical processes, machines subject to inspection) has to undergo treatment on a regular basis. More precisely, for each item a set of intervals is given during which a treatment must take place. The treatment itself takes one time-unit and is provided by some kind of machine with unbounded capacity (that is, it can process any number of items), and consists of "turning the machine on" at some point in time, say q (this corresponds to selecting column q). Then the items corresponding to intervals that are stabbed by column q undergo the treatment. The objective is to minimize (a weighted combination of) the number of times the machine is turned on plus the number of items not processed (an item is not processed when at least one of its intervals has not undergone the treatment (this corresponds to selecting the row corresponding to that item)). An example of such a problem is described in [4], see also Hassin & Megiddo [31].

Load balancing (Gaur et al.[26]). Imagine a two-dimensional $m \times n$ array of identical processors connected as a mesh, i.e., each processor in the array is connected to the adjacent processors. Such configuration is sometimes used in

areas such as numeric computation and image processing. The computation is done over an array of tasks and may require information exchange between the adjacent tasks. For each task a computation load is given, i.e., a number representing the workload on a processor performing the task. The problem is to assign the tasks to the processors, so as to minimize the total load assigned to one processor (see e.g. [24]). A useful method to map the numbers (or computation loads) to the processors is to first perform a rectilinear partitioning of the two-dimensional array of numbers, in which a set of $n-1$ vertical lines and $m-1$ horizontal lines are drawn to subdivide the array of numbers. Each subrectangle (subarray of numbers enclosed by successive vertical and horizontal lines) is now assigned to a processor in the mesh. This type of mapping is particularly useful when communication between adjacent processors on the mesh is fast compared to communication between non-adjacent processors. The goal is to minimize the maximum load assigned to a processor. Hence the rectilinear partitioning problem arises, in which one aims to find a partition minimizing the maximum sum of the numbers in a subrectangle. One of the approaches to perform the rectilinear partitioning, proposed by Gaur, Ibaraki and Krishnamurti [26], is based on a reduction to the problem of stabbing a set of rectangles by the minimum number of lines parallel to the axes. So we come to the rectangle stabbing problem, in which one is given a set of axes-parallel rectangles in the plane, whose corners are located at the integer points and whose area is larger than 1. The problem is to stab all the rectangles by the minimum number of lines located at integer points, where a horizontal (or vertical) line is said to stab a rectangle if it goes through its interior. The special case, when the height of each rectangle is exactly 2 leads to JIS with unit weights and demands.

Applications of RP^B.

Database decision support (Fukuda et al. [22]) Consider database mining systems based on association rules for two numeric attributes and one Boolean attribute. For example, in a database of bank customers, "Age" and "Balance" are two numerical attributes, and "CardLoan" is a Boolean attribute. Taking the pair (Age, Balance) as a point in two-dimensional space, consider an association rule of the form

$$(Age \in [25, 45]) \cap (Balance \in [15, 40]) \Rightarrow (Cardloan = Yes),$$

which implies that bank customers whose ages and balances are in the specified ranges tend to use card loan with a high probability. So, each such association rule corresponds to a rectangle in two-dimensional space with coordinate axes associated with "Age" and "Balance". Database mining systems can now generate the set of all such rules for a given database, given thresholds on confidence and support, and tag each with a weight that shows their importance or value. Following that, database decision systems choose a subset of these rules for further development, such as marketing. A common formulation of this task is to choose a collection of disjoint rules of largest total gain or value. The situation when the ranges of one of the attributes of the generated association rules are approximately of the same size leads to the rectangle packing problem with bounded height ratio.

Map labeling (Agarwal et al. [2], Poon et al. [42]). One of the tasks of map labeling is to provide features on a map with non-overlapping labels. Very often, features that need to be labeled are points and the labels are assumed to be rectangles, often of a uniform height (a bounding box of the label text in a uniform typeface). Moreover, each label has to be placed in one of the admissible positions relative to the feature it labels. Such admissible positions can be for instance all the positions where the rectangular label touches its point feature with a corner. It is natural to assume that the labels may have different importance, which is expressed in different weights assigned to the labels. The problem where one aims to find a maximum weight collection of nonoverlapping rectangular labels of a uniform height, each touching its point feature with one of the corners is equivalent to RP^1. Indeed, consider all the admissible positions of all the labels as the given closed rectangles. Since all the admissible positions of a particular label overlap with the point representing the feature, and thus overlap with each other, a feasible solution to the rectangle packing problem can include at most one label for each feature, being thus a feasible solution to the map labeling problem.

Techniques

Let us sketch the techniques we use to design our approximation algorithms.

LP-rounding algorithms are particularly useful for problems that can be naturally formulated via integer linear programming. The main idea of an

LP-rounding algorithm is to transform the optimal (fractional) solution of an LP relaxation into a feasible integer solution. The approximation guarantee is derived from comparing both the solutions. Notice that an LP-rounding algorithm requires to solve an LP problem, which can be a time consuming (though polynomial) operation. The beginning of development of rounding techniques as a tool for proving approximation guarantees falls in the eighties (see e.g. Hochbaum [34], Raghavan [43].)

The primal-dual method also proves to be especially appropriate for combinatorial optimization problems admitting a natural integer programming formulation. In the primal-dual method for approximation algorithms, a feasible solution to the problem and a feasible solution to the dual of an LP relaxation are constructed simultaneously, guided in one or another way by the complementary slackness conditions. The performance guarantee is proved by comparing the values of both solutions. Although many algorithms can be understood in terms of the primal-dual method, the first truly primal-dual approximation algorithm is the algorithm of Bar-Yehuda and Even of 1981 [11] for the vertex cover problem. In recent years, the power of the method has been established by a sequence of papers developing this technique for a range of combinatorial problems (see Goemans & Williamson [27] for a survey on network design problems, Vazirani [51]).

Dynamic programming, introduced by Bellman in 1957 [13], has become an established tool in the design of approximation algorithms, where it is typically used as a routine for solving subsidiary problems to optimality (see Agarwal et al. [2], Chan [16], Calinescu et al. [17] for illustrations).

Outline of the Thesis

Chapter 2 describes how primal-dual approximation methods can be applied to JIP and JIS. These two problems are closely related to each other, because they can be formulated on the same input data and the LP relaxations of their natural ILP formulations constitute a primal-dual pair of LP problems. We consider two different special cases of the pair JIP-JIS, namely the case of unit (column, row and interval) capacities and the case of unit (interval) demands, and in each of the cases describe an efficient algorithm that computes a pair of feasible solutions, one to JIS and one to JIP, whose values are within a factor of 2 of each other. This implies that one of the algorithms is a 2 -

approximation for JIS and 1/2- approximation for JIP in the case of unit demands, and the other provides the same approximation ratios for JIS and JIP in the case of unit capacities. We note that among those 4 approximation algorithms, the 1/2-approximation algorithm for JIP with unit demands has been described before, although not in terms of the primal-dual method, by Berman and DasGupta [14] and Bar-Noy et al [9]. Our contribution is in presenting it as a primal-dual algorithm and deriving an approximation for JIS.

Chapters 3 and 4 focus on approximation algorithms based on LP-rounding for JIS and special cases of it. Our LP-rounding algorithms provide better approximation guarantees for several special cases of JIS than the primal-dual algorithms from Chapter 2.

In chapter 3 we describe a $(w_0 + 1)/w_0$ - approximation algorithm for JIS^{w_0}, which is a special case of JIS where the interval demands are bounded from below by w_0. This also implies a 2-approximation for the general case of JIS. Further we describe a $e/(e-1) \approx 1.582$ -approximation algorithm for JIS with unit interval demands, referred to as *STAB*. We also show that the ratio between the optimal value of JIS with unit demands and the optimal value of the LP relaxation, i.e., the integrality gap factor, can achieve the value of $e/(e-1)$. This suggests that it is unlikely to improve over this approximation factor using an LP-rounding algorithm based on the same LP relaxation.

In chapter 4 we investigate the problem JIS_k - a special case of JIS, where at most k intervals are allowed to share a row. We prove that the algorithm *STAB* provides $\frac{1}{(1-(1-1/k)^k}$ - approximation guarantee for this special case. This approximation ratio is equal to the size of the integrality gap factor for $k = 2$. We also present an analysis of the size of the integrality gap factor for $k > 2$.

Furthermore, in chapter 4 we give a non-approximability result, showing that no polynomial-time approximation scheme is possible for JIS_2 (and thus JIS in general) unless $\mathcal{P} = \mathcal{NP}$.

Chapter 5 gives an illustration of using dynamic programming in an approximation algorithm. We focus on the problem RP^B, for which a polynomial time approximation scheme (PTAS) can be obtained by straightforward generalization of the PTAS by Agarwal et al. [2]. We show how one can achieve an improvement in the running time and memory consumption of this PTAS by refining a dynamic programming subroutine.

Preliminaries

In this thesis we use the following terminology and notation:

- A *δ-approximation algorithm* for a maximization (minimization) problem $\mathcal{P}$ is a polynomial time algorithm that delivers a feasible solution with the value $v(\mathcal{I}) \geq \delta \cdot OPT(\mathcal{I})$ $(v(\mathcal{I}) \leq \delta \cdot OPT(\mathcal{I}))$ for each instance $\mathcal{I}$ of $\mathcal{P}$, where $OPT(\mathcal{I})$ is the optimal value of $\mathcal{P}$ for $\mathcal{I}$.
- A *polynomial-time approximation scheme* (PTAS) for a problem $\mathcal{P}$ is a family of δ-approximation algorithms, for each $\delta < 1$ if $\mathcal{P}$ is a maximization problem, and for each $\delta > 1$ if $\mathcal{P}$ is a minimization problem. Note, that the running time of a δ-approximation algorithm is allowed to be exponential in δ.
- A *combinatorial* algorithm is a polynomial algorithm whose running time does not depend on the numerical values in the input, provided that all the basic arithmetic operations on these values are considered as constant time operations. No combinatorial algorithm is known for solving an arbitrary linear programming problem.
- We use the abbreviation *ILP* for *integer linear programming*, and *LP* for *linear programming*.
- The *LP relaxation* of an ILP formulation is an LP problem which arises when the integrality constraints of the ILP formulation are relaxed.
- Often we refer to a solution of the LP relaxation of an ILP formulation of some problem as an *LP solution* of the problem. Since we consider for each problem only one ILP formulation, there can not be any ambiguity.
- The *integrality gap factor* of an ILP formulation is a maximum ratio over all instances between the optimum value of the formulation and the optimum value of its LP relaxation.
- We use notation $[a, b]$ for the set of integers $\{a, a+1, ..., b\}$.

Chapter 2

Primal-dual approximation algorithms for the job interval packing and job interval stabbing problem.

2.1 Introduction

In this chapter we describe two primal-dual approximation algorithms for two different special cases of the pair JIP-JIS. Each of the algorithms returns two feasible solutions, one for JIP and one for JIS, whose values are within a factor of 2 of each other.

Preliminaries. Let us give more formal definitions of JIP and JIS.

We refer to the family of inputs described as follows as JI:
given is a grid consisting of t columns, numbered consecutively from left to right, and m numbered rows. Furthermore, given is a set of intervals $I = \{1, 2, ..., n\}$ lying on the rows of the grid. An interval i is specified by the triple (l_i, r_i, ρ_i), where l_i, r_i are the indices of the left- and the right-most columns intersecting interval i and ρ_i is the index of the row where interval i lies. For each column c, row r and interval i we are given positive integral parameters v_c, u_r and p_i respectively, referred to as the column, row and interval capacities. For each interval i we are given a positive integral parameter w_i referred to as the interval demand. We assume that the intervals are ordered according to nondecreasing r_i.

We use the following terminology: column c is said to stab interval i if column c intersects interval i, i.e., $l_i \le c \le r_i$. Also, row ρ_i is said to stab interval i if interval i lies on row ρ_i.

For each instance $\mathcal{I}$ of JI we formulate the following problems:

Job interval packing problem (JIP):
for each interval $i \in I$ specify an integral multiplicity x_i such that:
- it does not exceed the interval capacity p_i,
- for each column c the sum of multiplicities of the intervals stabbed by column c does not exceed the column capacity v_c,
- for each row r the sum of multiplicities of the intervals stabbed by row r does not exceed the row capacity u_r,
- the total demand $\sum w_i x_i$ is maximized.

Job interval stabbing problem (JIS):
for each column c, row r and interval i specify integral multiplicities y_c, z_r and s_i respectively, such that:
- for each interval i the sum of the multiplicities of the columns stabbing interval i plus the multiplicity of the row stabbing interval i plus the multiplicity of interval i equals at least its demand w_i,
- the total capacity $\sum_{c=1}^{t} v_c y_c + \sum_{r=1}^{m} u_r z_r + \sum_{i=1}^{n} p_i s_i$ is minimized.

One may interpret the interval demands w_i as interval weights in JIP and the column, row and interval capacities v_c, u_r, p_i as column, row and interval weights in JIS. However in this chapter we refer to these parameters as demands and capacities to keep the terminology uniform for both problems.

In this chapter we consider two special cases of JIP and JIS. The first special case restricts to the instances where all the column, row and interval capacities are unit, i.e., $v_c = u_r = p_i = 1$, $\forall c, r, i$. We refer to the family of inputs satisfying this condition as *JI with unit capacities*. The second special case assumes that all the interval demands are unit, i.e., $w_i = 1$, $\forall i = 1, ..., n$. The family of these inputs is referred to as *JI with unit demands*.

Both problems JIP and JIS are NP-hard and, moreover, MAX SNP-hard even if all the parameters are unit (see Spieksma [47] and Section 4.3 of this thesis). This implies that there is no polynomial time approximation scheme for each of these problems unless $\mathcal{P} = \mathcal{NP}$.

Below we give natural ILP formulations of JIP and JIS:

JIP:

$$\begin{aligned}
&\text{Maximize} && \textstyle\sum_{i=1}^{n} w_i x_i && && (2.1.1)\\
&\text{subject to} && \textstyle\sum_{i:p_i=r} x_i \le u_r && \forall r = 1,..,m && (2.1.2)\\
& && \textstyle\sum_{i:c\in[l_i,r_i]} x_i \le v_c && \forall c = 1,..,t && (2.1.3)\\
& && x_i \le p_i && \forall i = 1,..,n && (2.1.4)\\
& && x_i \in \mathbb{Z}_+^1 && \forall i = 1,..,n && (2.1.5)
\end{aligned}$$

JIS:

$$\begin{aligned}
&\text{Minimize} && \textstyle\sum_{c=1}^{t} v_c y_c + \sum_{r=1}^{m} u_r z_r + \sum_{i=1}^{n} p_i s_i && && (2.1.6)\\
&\text{subject to} && z_{p_i} + \textstyle\sum_{c\in[l_i,r_i]} y_c + s_i \ge w_i && \forall i = 1,..,n && (2.1.7)\\
& && z_r, y_c, s_i \in \mathbb{Z}_+^1 && \forall r, c, i. && (2.1.8)
\end{aligned}$$

Observe, that for any instance $\mathcal{I}$ of JI the LP relaxations of these ILP formulations, which arise when we substitute the integrality constraints (2.1.5) and (2.1.8) by nonnegativity constraints $x_i \ge 0$, $\forall i = 1,..,n$ and $z_r, y_c, s_i \ge 0$, $\forall r, c, i$ respectively, constitute a primal-dual pair of LP problems. From the strong duality theorem for linear programming follows, that for any instance $\mathcal{I}$, the LP relaxations of JIP and JIS have the same optimum value, which we denote by $LP(\mathcal{I})$, and it holds:

$$JIP(\mathcal{I}) \le LP(\mathcal{I}) \le JIS(\mathcal{I}), \tag{2.1.9}$$

where $JIP(\mathcal{I})$ and $JIS(\mathcal{I})$ are the optimum values of JIP and JIS for $\mathcal{I}$ respectively.

We say that problems JIP and JIS are weakly dually related and establish the following result:

Lemma 2.1.1. *(Weak duality lemma for JIP and JIC)*
For any instance $\mathcal{I}$ of JI, for any feasible solution to JIP with value $JIP^{feas}(\mathcal{I})$ and any feasible solution to JIS with value $JIS^{feas}(\mathcal{I})$ holds:

$$JIP^{feas}(\mathcal{I}) \le JIS^{feas}(\mathcal{I}).$$

Proof. Follows from (2.1.9). □

Previous research. JIP and its special cases have received a great deal of attention in the literature. A greedy $1/2$ - approximation algorithm for JIP with unit capacities and demands (i.e., $u_r = v_e = p_i = w_i = 1,\ \forall e, r, i$) is described in Spieksma [47]. A better approximation guarantee for this case has been found by Chuzhoy et al. [18]. They present an algorithm with performance guarantee $(e-1)/e - \varepsilon \approx 0.63 - \varepsilon$, for any $\varepsilon > 0$, exploiting a sophisticated technique involving randomized LP-rounding. Bar-Noy et al. [8] describe a $(1/2-\varepsilon)$-approximation algorithm for JIP with unit capacities (i.e., $u_r = v_e = p_i = 1,\ \forall e, r, i$) using an LP-rounding technique. One can easily generalize this algorithm to JIP (with arbitrary capacities and demands) and, using the ideas introduced by Calinescu et al. [17], improve the approximation factor to $1/2$. The resulting $1/2$ - approximation algorithm (which involves solving an LP problem) has to our knowledge so far the highest approximation factor for JIP. Combinatorial $1/2$-approximation algorithms for JIP with unit capacities are described by Berman and DasGupta [14] and Bar-Noy et al [9]. In Erlebach & Spieksma [21] the performance of greedy algorithms for JIP with constant column capacity and unit row capacity (i.e., $v_e = v,\ \forall e,\ u_r = 1,\ \forall r$) is investigated using competitive analysis.

So far JIS has not been as extensively studied as JIP. Hassin and Megiddo [31] describe a combinatorial $2 - \frac{1}{K+1}$ - approximation algorithm for the case of JIS with unit capacities and demands, where each interval is intersected by exactly K columns, and a 2-approximation for a slightly more general case, when each interval is intersected by the same number of columns.

For JIS with unit demands a 2-approximation algorithm based on LP-rounding is implied by the results of Gaur et al. [26]. They consider the so-called *rectangle stabbing* problem, which generalizes JIS with unit demands to the case when instead of intervals we are given rectangles which can intersect several rows of the grid.

Outline of the chapter and our contribution. This chapter is based on the results presented in [39].

Section 2.2 is devoted to JI with unit capacities. Notice that JIP with unit capacities is a special case of the weighted set packing problem. For a better understanding of the primal-dual principles we first describe a generic primal-dual algorithm, called *Local Covering*, for the weighted set packing problem (Subsection 2.2.1) and then specify a set-up of *Local Covering* for JIP with unit capacities, yielding algorithm *ALG1* (Subsection 2.2.2). We also give conditions which guarantee that a particular set-up of *Local Covering* yields a constant factor approximation algorithm. This algorithm is in fact a primal-

dual interpretation of the *Opportunity Cost* algorithm described by Akcoglu et al. in [1], which is in turn based on the local-ratio technique, introduced by Bar-Yehuda and Even [12] and later elaborated by Bar-Noy et al. [9].

A distinctive feature of the primal-dual algorithm *ALG1* described in Subsection 2.2.2 is that it simultaneously delivers two feasible solutions, one for JIP and one for JIS with unit capacities. Moreover, we show that their values are within a factor of 2 of each other. The weak duality relation between JIP and JIS implies then that these solutions are respectively a $1/2$ - approximation for JIP with unit capacities and a 2 - approximation for JIS with unit capacities. We also show that our analysis is tight, i.e., the approximation guarantees can not be improved by carrying out a better analysis of the algorithm.

We note that if viewed as merely a $1/2$-approximation algorithm for JIP, *ALG1* behaves similar to the algorithms described previously by Berman and DasGupta [14] and Bar-Noy et al [9]. We stress that our contribution is in the 2-approximation algorithm for JIS with unit capacities.

In Section 2.3 we consider JI with unit demands. It is easy to see that JIS with unit demands is a special case of the the weighted hitting set problem. Following the framework of the generic *primal-dual algorithm with reverse delete step* described in [27] for the weighted hitting set problem, we develop a primal-dual approximation algorithm *ALG2*. Similar to *ALG1* it returns two feasible solutions, one for JIS and one for JIP with unit demands. Again, we show that their values are within a factor of 2 of each other. This makes *ALG2* a 1/2-approximation algorithm for JIP and a 2-approximation algorithm to JIS with unit demands. We conclude by showing tightness of the analysis.

Notice that all the algorithms described in this chapter are combinatorial in contrast to LP-rounding algorithms.

2.2 The Case of Unit Capacities

2.2.1 The Local Covering algorithm for the Weighted Set Packing Problem

Consider the *weighted set packing* problem (WSP):
given is a collection $\mathcal{S}$ of subsets of a finite ground set $\mathcal{E}$. For each subset $s \in \mathcal{S}$ a nonnegative weight w_s is given. Find a maximum-weight collection

A of disjoint subsets from $\mathcal{S}$.

Below we give a natural ILP formulation of WSP, where we use the characteristic vector of A, $\chi \in \{0,1\}^{|\mathcal{S}|}$, as a vector of decision variables:

WSP:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{s\in\mathcal{S}} w_s\chi_s && (2.2.1)\\ \text{subject to} \quad & \textstyle\sum_{s:e\in s} \chi_s \le 1 \quad \forall e \in \mathcal{E} && (2.2.2)\\ & \chi_s \in \{0,1\} \quad \forall s \in \mathcal{S} && (2.2.3) \end{aligned}$$

By substituting nonnegativity constraints $\chi_s \ge 0, \quad \forall s \in \mathcal{S}$ for integrality constraints (2.2.3) we obtain the LP relaxation of this formulation.

Let us also introduce the dual linear problem to this LP relaxation: (here we use $\gamma \in \mathbb{R}_+^{|\mathcal{E}|}$ for a vector of dual variables)

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum_{e\in\mathcal{E}} \gamma_e && (2.2.4)\\ \text{subject to} \quad & \textstyle\sum_{e\in s} \gamma_e \ge w_s \quad \forall s \in \mathcal{S} && (2.2.5)\\ & \gamma_e \ge 0 \quad \forall e \in \mathcal{E} && (2.2.6) \end{aligned}$$

For ease of discussion let us introduce some terminology. We say that:

- an element $e \in \mathcal{E}$ and a subset $s \in \mathcal{S}$ are *incident* to each other if $e \in s$;
- the *coverage* of a subset $s \in \mathcal{S}$ by a vector $\gamma \in \mathbb{R}_+^{|\mathcal{E}|}$ is a value equal to the sum of the elements of γ corresponding to the elements e belonging to s, i.e., $\sum_{e\in s} \gamma_e$;
- a subset $s \in \mathcal{S}$ is *covered* if its coverage equals at least w_s. Otherwise, we say that s is *violated*;
- a subset s_1 is a *neighbor* of a subset s_2 if they share a common element;
- $N(s)$ is the set of all the neighbors of s. Note that $s \in N(s)$;
- a feasible solution χ to the ILP formulation (2.2.1)-(2.2.3) as well as the corresponding set $A \in \mathcal{S}$ is a *feasible packing*;
- a feasible solution γ to the LP formulation (2.2.4)-(2.2.6) is a *feasible covering*.

Definition 2.2.1. For a given vector $\gamma \in \mathbb{R}_+^{|E|}$ and a subset s^* let us call a vector $\delta \in \mathbb{R}_+^{|E|}$ satisfying: $\sum_{e \in s} \delta_e \geq \min(w_s - \sum_{e \in s} \gamma_e, w_{s^*} - \sum_{e \in s^*} \gamma_e)$, $\forall s \in N(s^*)$, a *local covering* for γ in s^*.

Let us now describe a generic primal-dual algorithm *Local Covering* for WSP. The framework of the algorithm is the following: initially all the dual variables γ_e are zero and A is empty. Until γ becomes a feasible covering do:

- select (some) subset s, violated by the current γ, and push it on a stack,
- construct a local covering δ for γ in s,
- increment vector γ by the the values of vector δ.

When γ becomes a feasible covering, pop the subsets from the stack iteratively and each time add a subset to A if this does not violate the feasibility of A.

Figure 2.1 gives a formal description of *Local Covering* (notice that $\delta \in \mathbb{R}_+^{|E|}$, $\Delta \in \mathbb{R}_+$, $l \in \mathbb{N}$).

1. $\gamma \leftarrow \mathbf{0}$
2. $A \leftarrow \emptyset$
3. $l \leftarrow 0$
4. Until γ is a feasible covering
   $l \leftarrow l + 1$
   Select s^l such that $\sum_{e \in s^l} \gamma_e < w_{s^l}$
   $\Delta \leftarrow w_{s^l} - \sum_{e \in s^l} \gamma_e$
   Find $\delta \in \mathbb{R}_+^{|E|}$ s. t.: $\sum_{e \in s} \delta_e \geq \min(w_s - \sum_{e \in s} \gamma_e,\ \Delta), \quad \forall s \in N(s^l)$
   $\gamma \leftarrow \gamma + \delta$
5. For l downto 1
   if $A \cup \{s^l\}$ is feasible then $A \leftarrow A \cup \{s^l\}$
6. Return A (and γ)

Figure 2.1: Algorithm *Local Covering*.

Observe that it is not exactly specified in the algorithm how s^l and δ are selected. The description of these selection procedures is left to a particular set up.

It does not make sense to discuss implementation and efficiency of the algorithm in such a general setting. Let us only establish the following result:

Theorem 2.2.1. *For any instance $\mathcal{I}$ of WSP, the set A and the vector γ returned by Local Covering are a feasible packing and a feasible covering respectively. Moreover, if Δ and δ found by the algorithm at each iteration satisfy:*

$$\sum_{e \in \mathcal{E}} \delta_e \leq \beta \cdot \Delta,$$

then

$$\sum_{e \in \mathcal{E}} \gamma_e \leq \beta \sum_{s \in A} w_s.$$

Proof. The feasibility of A and γ is obvious. Let us establish the relation between their values.

Let p be the number of iterations made by the algorithm at step 4. Observe that p is at most $|\mathcal{S}|$ since at each iteration the number of violated subsets decreases by at least 1 and the iterations stop when there is no more violated subset.

Let Δ^l, δ^l and γ^l $(l = 1, ..., p)$ be the values of Δ, vector δ and vector γ respectively at the end of the l-th iteration of step 4. Let γ^0 be a zero vector. So we have $\gamma^l = \gamma^{l-1} + \delta^l$, $\forall l = 1, ..., p$. The condition of the theorem can be written as

$$\sum_{e \in \mathcal{E}} \delta_e^l \leq \beta \cdot \Delta^l \quad \forall l = 1, .., p.$$

Further, let A^q be the state of set A at the end of the loop of the cycle at step 5, corresponding to $l = q$. Let A^{p+1} be $\emptyset$. Then we have $\emptyset = A^{p+1} \subseteq A^p \subseteq ... \subseteq A^1$.

We show that for each $l = 1, ..., p+1$

$$\sum_{e \in \mathcal{E}} (\gamma_e^p - \gamma_e^{l-1}) \leq \beta \sum_{s \in A^l} (w_s - \sum_{e \in s} \gamma_e^{l-1}). \tag{2.2.7}$$

Then for $l = 1$, since γ^0 is a zero vector and A^1 is the set A returned by the algorithm, we obtain the result of the theorem.

We use induction on $l = p+1, ..., 1$. The basis of induction is $l = p+1$. Then inequality (2.2.7) trivially holds, since $A^{p+1} = \emptyset$. Suppose the inequality is

proven for $l = j+1$. Let us prove it for $l = j$. So, we have:

$$\sum_{e \in \mathcal{E}} (\gamma_e^p - \gamma_e^j) \leq \beta \sum_{s \in A^{j+1}} (w_s - \sum_{e \in s} \gamma_e^j), \tag{2.2.8}$$

and the theorem will be established if we show that

$$\sum_{e \in \mathcal{E}} (\gamma_e^p - \gamma_e^{j-1}) \leq \beta \sum_{s \in A^j} (w_s - \sum_{e \in s} \gamma_e^{j-1}). \tag{2.2.9}$$

First let us show that the following inequality holds:

$$\sum_{s \in A^j} (w_s - \sum_{e \in s} \gamma_e^{j-1}) \geq \sum_{s \in A^{j+1}} (w_s - \sum_{e \in s} \gamma_e^j) + \Delta^j. \tag{2.2.10}$$

There are two cases possible: either $A^j = A^{j+1}$ or $A^j = A^{j+1} \cup \{s^j\}$.

Consider the first case. Suppose $A^j = A^{j+1}$. Clearly, this case is only possible if A^{j+1} contains a neighbor of subset s^j, say subset s^q, i.e., $s^q \in N(s^j)$. Consider γ^j, it is equal to $\gamma^{j-1} + \delta^j$, where δ^j satisfies:

$$\sum_{e \in s} \delta_e^j \geq \min(w_s - \sum_{e \in s} \gamma_e^{j-1}, \ \Delta^j), \quad \text{for all } s \in N(s^j),$$

and in particular for s^q, i.e., $\sum_{e \in s^q} \delta_e^j \geq \min(w_{s^q} - \sum_{e \in s^q} \gamma_e^{j-1}, \ \Delta^j)$. Assume,

$$\sum_{e \in s^q} \delta_e^j \geq \min(w_{s^q} - \sum_{e \in s^q} \gamma_e^{j-1}, \ \Delta^j) = w_{s^q} - \sum_{e \in s^q} \gamma_e^{j-1}.$$

Then

$$\sum_{e \in s^q} \gamma_e^j = \sum_{e \in s^q} \gamma_e^{j-1} + \sum_{e \in s^q} \delta_e^j \geq w_{s^q}.$$

This means that subset s^q is covered by γ^j and therefore also by $\gamma^{j+1}, ..., \gamma^p$, since $\gamma^j \leq \gamma^{j+1} \leq ... \leq \gamma^p$. Therefore subset s^q is not violated at iteration j or later and can not be selected and pushed on the stack during iterations $j+1, .., p$ of step 4, which contradicts with $s^q \in A^{j+1}$. Therefore, the only possible case is that

$$\sum_{e \in s^q} \delta_e^j \geq \min(w_{s^q} - \sum_{e \in s^q} \gamma_e^{j-1}, \ \Delta^j) = \Delta^j.$$

Now, using the assumption $A^j = A^{j+1}$, the facts that $\gamma^{j-1} = \gamma^j - \delta^j$ and that there exists $s^q \in A^{j+1}$ such that the above inequality holds, we can rewrite the left-hand side of (2.2.10) as:

$$\sum_{s \in A^{j+1}} \left(w_s - \sum_{e \in s} \gamma_e^{j-1} \right) = \sum_{s \in A^{j+1}} \left(w_s - \sum_{e \in s} \gamma_e^j + \sum_{e \in s} \delta_e^j \right) =$$

$$= \sum_{s \in A^{j+1}} \left(w_s - \sum_{e \in s} \gamma_e^j \right) + \sum_{s \in A^{j+1}} \sum_{e \in s} \delta_e^j \geq \sum_{s \in A^{j+1}} \left(w_s - \sum_{e \in s} \gamma_e^j \right) + \Delta^j,$$

which proves (2.2.10) in the case $A^j = A^{j+1}$.

Consider now the case $A^j = A^{j+1} \cup \{s^j\}$.
Using the fact that $\Delta^j = w_{s^j} - \sum_{e \in s^j} \gamma_e^{j-1}$ and $\gamma^{j-1} \leq \gamma^j$, rewrite the left-hand side of (2.2.10) as:

$$\sum_{s \in A^{j+1}} \left(w_s - \sum_{e \in s} \gamma_e^{j-1} \right) + (w_{s^j} - \sum_{e \in s^j} \gamma_e^{j-1}) \geq \sum_{s \in A^{j+1}} \left(w_s - \sum_{e \in s} \gamma_e^j \right) + \Delta^j,$$

which proves (2.2.10) in the case $A^j = A^{j+1} \cup \{s^j\}$.

Now, with (2.2.10) established, we rewrite it by multiplying both sides by β:

$$\beta \sum_{s \in A^j} (w_s - \sum_{e \in s} \gamma_e^{j-1}) \geq \beta \sum_{s \in A^{j+1}} (w_s - \sum_{e \in s} \gamma_e^j) + \beta \Delta^j.$$

Next, using the condition of the theorem, i.e., $\sum_{e \in \mathcal{E}} \delta_e^j \leq \beta \Delta^j$, the induction hypothesis (2.2.8) and the fact $\gamma^j - \delta^j = \gamma^{j-1}$, we can bound the last expression from below by:

$$\sum_{e \in \mathcal{E}} (\gamma_e^p - \gamma_e^j) + \sum_{e \in \mathcal{E}} \delta_e^j = \sum_{e \in \mathcal{E}} (\gamma_e^p - \gamma_e^{j-1}).$$

Thus we establish inequality (2.2.9), which proves the theorem. □

The weak duality theorem for linear programming implies

$$OPT(\mathcal{I}) \leq \sum_{e \in \mathcal{E}} \gamma_e(\mathcal{I}),$$

where $OPT(\mathcal{I})$ is the optimum value of WSP for instance $\mathcal{I}$ and $\gamma(\mathcal{I})$ is feasible covering γ returned by *Local Covering* for $\mathcal{I}$. Let $A(\mathcal{I})$ be the set A returned by the algorithm for $\mathcal{I}$, then from Theorem 2.2.1 we have

$$OPT(\mathcal{I}) \leq \sum_{e \in \mathcal{E}} \gamma_e(\mathcal{I}) \leq \beta \sum_{s \in A(\mathcal{I})} w_s.$$

Corollary 2.2.2. *If (a particular set-up of) the Local Covering algorithm runs in polynomial time and the condition of Theorem 2.2.1 is satisfied, then the Local Covering is a $\frac{1}{\beta}$-approximation algorithm.*

2.2.2 Algorithm *ALG1*

Suppose we are given an instance $\mathcal{I}$ of JI with unit capacities. Recall that this is a grid consisting of t columns, numbered consecutively from left to right, and m numbered rows together with a set of intervals $I = \{1, 2, ..., n\}$ lying on the rows of the grid. An interval i is specified by the triple (l_i, r_i, ρ_i), where l_i, r_i are the indexes of the left- and the right-most columns intersecting the interval and ρ_i is the index of the row where it lies. For each interval i we are given a positive integral parameter w_i referred to as the interval demand. We assume that the intervals are ordered according to nondecreasing r_i.

Consider the job interval packing problem (JIP) with unit capacities, which can be formulated as follows: *select a maximum weight subset of intervals A such that no two intervals share a column or row.*

Observe that JIP with unit capacities is a special case of the weighted set packing problem (WSP), considered in the previous section. Indeed, let the ground set $\mathcal{E}$ be the set of all the columns and rows of the grid and the collection $\mathcal{S}$ be $\{s_1, ..., s_n\}$, where s_i is the subset of columns and the row stabbing interval i, i.e. $s_i = \{column\ l_i, ..., column\ r_i,\ row\ \rho_i\}$. The weights of the subsets are equal to the corresponding interval weights: $w_{s_i} = w_i$, $\forall i = 1, ..., n$. It is easy to see that any feasible packing corresponds to a feasible solution to JIP with unit capacities of the same value.

Below we give an ILP formulation of JIP with unit capacities, where we use a characteristic vector x of A as a vector of decision variables:

$$\begin{array}{lll} \text{JIP:} \quad \text{Maximize} & \sum_{i=1}^{n} w_i x_i & (2.2.11) \\ \text{subject to} & \sum_{i:\rho_i=r} x_i \leq 1 \quad \forall r = 1,..,m & (2.2.12) \\ & \sum_{i:c\in[l_i,r_i]} x_i \leq 1 \quad \forall c = 1,..,t & (2.2.13) \\ & x_i \in \mathbf{Z}_+^1 \quad \forall i = 1,..,n. & (2.2.14) \end{array}$$

The dual to its LP relaxation (the dual variables $z \in \mathbb{R}^m$ and $y \in \mathbb{R}^t$ correspond to the constraints (2.2.12) and (2.2.13) respectively) looks as follows:

$$\begin{array}{lll} \text{Dual:} \quad \text{Minimize} & \sum_{c=1}^{t} y_c + \sum_{r=1}^{m} z_r & (2.2.15) \\ \text{subject to} & z_{\rho_i} + \sum_{c\in[l_i,r_i]} y_c \geq w_i \quad \forall i = 1,..,n & (2.2.16) \\ & z_r, y_c \geq 0 \quad \forall r, c. & (2.2.17) \end{array}$$

Notice that by replacing in this formulation the nonnegativity constraints (2.2.17) with integrality constraints

$$z_r, y_c \in \mathbf{Z}_+, \ \forall r, c, \tag{2.2.18}$$

we obtain an ILP formulation of the job interval stabbing (JIS) problem with unit capacities: *for each column c and row r specify integral multiplicities y_c and z_r respectively such that:*

- for each interval i the sum of multiplicities of the columns and the row stabbing interval i is at least the demand w_i,

- the sum of the multiplicities is minimum.

Notice that in the case of JIS with unit capacities the interval multiplicities can be fixed to zero without loss in the optimum value, since, given any feasible solution, one can obtain a feasible solution of the same value by decreasing interval multiplicities to zero and increasing row multiplicities so as to preserve the feasibility.

We describe now a set up of the generic algorithm *Local Covering* for JIP with unit capacities, yielding a primal-dual algorithm called *ALG1*.

Let us reproduce here the step 4 of the algorithm *Local Covering* (see Figure 2.2) and translate it into the terms of JIP with unit capacities.

Until γ is a feasible covering
$l \leftarrow l+1$
(1) Select s^l such that $\sum_{e \in s^l} \gamma_e < w_{s^l}$
(2) $\Delta \leftarrow w_{s^l} - \sum_{e \in s^l} \gamma_e$
(3) Find $\delta \in \mathbb{R}_+^{|\mathcal{E}|}$ s. t.: $\sum_{e \in s} \delta_e \geq \min(w_s - \sum_{e \in s} \gamma_e,\ \Delta),\ \ \forall s \in N(s^l)$
(4) $\gamma \leftarrow \gamma + \delta$

Figure 2.2: Step 4 of *Local Covering*.

Line (1), i.e., selecting a violated subset $s^l = s_i$ corresponds in our context to selecting interval i such that $\sum_{c \in [l_i, r_i]} y_c + z_{\rho_i} < w_i$. When more then one index i satisfies this condition we select the smallest of them.

Line (2) should be translated as $\Delta \leftarrow w_i - \sum_{c \in [l_i, r_i]} y_c - z_{\rho_i}$.
In line (3) we have to find $\delta = (\delta_{col\ 1}, ..., \delta_{col\ t},\ \delta_{row\ 1}, ..., \delta_{row\ m})$ s.t.

$$\sum_{e \in s_j} \delta_e \geq \min(w_j - \sum_{c \in [l_j, r_j]} y_c - z_{\rho_j},\ \Delta),\ \ \forall j \text{ s.t. } s_j \in N(s_i), \qquad (2.2.19)$$

where i is the number selected in line (1). We assign the value of Δ to the elements of δ corresponding to the right-most column stabbing interval i and to its row, and 0 to all the other elements.

Lemma 2.2.3. *If vectors z and y and index i are such that $w_j - \sum_{c \in [l_j, r_j]} y_c - z_{\rho_j} \leq 0$ for all $j = 1, ..., i-1$, then vector δ defined as:*

$$\delta_e = \begin{cases} \Delta, & \text{if } e = col\ r_i, \\ \Delta, & \text{if } e = row\ \rho_i, \\ 0, & \text{otherwise} \end{cases}$$

satisfies (2.2.19) for any $\Delta > 0$.

Proof. The condition of the lemma guarantees that (2.2.19) is satisfied for all $s_j \in N(s_i)$ such that $j < i$.

Consider the other neighbors of s_i, i.e. all $s_j \in N(s_i)$ such that $j \geq i$. These subsets correspond to the intervals that either lie on the same row or share a column with interval i and whose right-most stabbing column has index at least equal to r_i. Then sharing a column with interval i implies

sharing the column r_i. This means that each subset $s_j \in N(s_i)$, $j \geq i$ includes either row ρ_i or column r_i. Thus:

$$\forall s_j \in N(s_i), \text{ s.t. } j \geq i: \sum_{e \in s_j} \delta_e \geq \min(\delta_{col\, r_i}, \delta_{row\, \rho_i}) = \Delta.$$

This implies the lemma. □

Figure 2.3 shows the formal description the algorithm *ALG1*. Here A is a set of interval indices.

1. $y \leftarrow 0, z \leftarrow 0$
2. $A \leftarrow \emptyset$
3. $l \leftarrow 0$
4. For $i \leftarrow 1$ to n
   $\Delta \leftarrow w_i - \sum_{c \in [i, r_i]} y_c - z_{\rho_i}$,
   if $\Delta > 0$ then:
   $l \leftarrow l + 1,\ y_{r_i} \leftarrow y_{r_i} + \Delta,\ z_{\rho_i} \leftarrow z_{\rho_i} + \Delta,\ i_l \leftarrow i$
5. For l downto 1
   if $A \cup \{i_l\}$ is feasible to JIP then $A \leftarrow A \cup \{i_l\}$
6. Return A (and y, z)

Figure 2.3: Algorithm *ALG1*

Theorem 2.2.4. *For any instance $\mathcal{I}$ of JI with unit capacities, set A and vectors (y, z) returned by the algorithm* ALG1 *describe feasible solutions to JIP and JIS respectively, and their values are related as:*

$$value(y, z) \leq 2 \cdot value(A).$$

Proof. Obviously A is a feasible solution to JIP with unit capacities. Consider (z, y). According to Theorem 2.2.1 it is a feasible covering, i.e., it is a feasible solution to the LP formulation (2.2.15)-(2.2.17). Obviously the vectors z and y are integral since w_i is integral for all $i = 1, ..., n$. Thus (z, y) is a feasible solution to the ILP formulation (2.2.15),(2.2.16),(2.2.18).

Let us establish the ratio of 2 between the values of the solutions. Observe, that the conditions of Theorem 2.2.1 are satisfied with $\beta = 2$. Indeed, at each iteration of step 4 of the algorithm $\sum_{e \in \mathcal{E}} \delta_e = 2\Delta$. Theorem 2.2.1 implies that

$$\sum_{r=1}^{m} z_r + \sum_{c=1}^{t} y_c \leq 2 \cdot \sum_{i \in A} w_i.$$

□

Theorem 2.2.5. *Algorithm* ALG1 *can be implemented to run in* $O(n \log n)$ *time.*

Proof. It costs at most $O(n)$ time to maintain and update y and z since $z \in \mathbb{R}^m$, where m is at most n (we do not consider empty rows of the grid) and $y \in \mathbb{R}^t$ where all the elements except $y_{\rho_1}, y_{\rho_2}, ..., y_{\rho_n}$ stay zero throughout the algorithm and thus do not need to be maintained.

In step 4 the only time consuming part is calculating $\sum_{c \in [l_i, r_i]} y_c$. Let us explain how to do it in $O(\log n)$ time. Consider the i-th iteration of the for-cycle in step 4. Suppose that at the beginning of it we have some (z, y) and suppose we know the value of $Sum_j \equiv \sum_{c \leq j} y_c$ for each $j = r_1, ..., r_{i-1}$. Since at this moment only $y_{\rho_1}, ..., y_{\rho_{i-1}}$ can be positive and hence $y_c = 0$ for all $c > r_{i-1}$, $\sum_{c \in [l_i, r_i]} y_c$ equals the difference $(Sum_{r_{i-1}} - Sum_{c^*})$, where $c^* = \max\{c : c \in \{r_1, ..., r_n\}, c < l_i\}$. Using binary search, c^* can be found in $O(\log n)$ time.

It is easy to maintain these values of Sum_j. For that purpose let us maintain an auxiliary variable Sum which at each moment in time equals $\sum_{c \in \{r_1, ..., r_n\}} y_c$ for the current value of vector y. At the beginning of the algorithm $Sum = 0$ and at every iteration $i = 1, ..., n$ of step 4 such that $\Delta > 0$ we increment the value of Sum by the same value as y_{r_i} i.e. by Δ, and assign it to Sum_{r_i}. So, at the beginning of the i-th iteration the values Sum_c for all $c \in \{r_1, ..., r_{i-1}\}$ are known.

Thus each iteration in the 4th step takes $O(\log n)$ time, which implies the running time $O(n \log n)$ for all the iterations.

The most complicated operation at step 6 is the feasibility check. We use a mechanism of indicators and pointers to be able to check whether the place necessary to insert a candidate interval is occupied. We maintain an indicator for each row, which is set initially into position "free", and a pointer that at each moment points to the left endpoint of the interval, added

to A last. When an interval i is added to A the indicator attached to the row ρ_i is set into position "occupied" and the pointer is set to point onto the column l_i. Recall that at step 6 we consider the intervals in the order of decreasing numbers. This means that the right endpoint of each next candidate interval, say i, lies to the left or coincides with any of the right endpoints of the intervals added to A before: $r_i \leq r_j$, $\forall j \in A$. Thus the interval i does not share a column with any of the intervals in A if and only if r_i is less than the index of the left endpoint of the interval added to A last, which is at any time pointed by the pointer. Thus to check whether insertion of an interval i preserves feasibility of A it is enough to make sure that the row ρ_i is indicated free and the column number c pointed by the pointer is bigger then r_i. This can be implemented in a constant time. □

From the weak duality relation between JIP and JIS and from Theorem 2.2.4 we have that for any instance $\mathcal{I}$ of JI with unit capacities:

$$JIP(\mathcal{I}) \leq value(y(\mathcal{I}), z(\mathcal{I})) \leq 2 \cdot value(A(\mathcal{I})) \leq 2JIS(\mathcal{I}),$$

where $JIP(\mathcal{I})$ and $JIS(\mathcal{I})$ are the optimal values of JIP and JIS on $\mathcal{I}$ and $y(\mathcal{I}), z(\mathcal{I})$ and $A(\mathcal{I})$ are the solutions returned by the algorithm *ALG1* applied to $\mathcal{I}$.

Corollary 2.2.6. *Algorithm* ALG1 *is a 1/2-approximation algorithm for JIP with unit capacities and 2-approximation for JIS with unit capacities.*

Tightness. The results stated in Corollary 2.2.6 are tight, i.e. the analysis of the algorithm's performance can not be improved to provide a better factor.

Example 1 shows that the ratio between the value of the solution returned by the algorithm and the optimal value of JIP with unit capacities can be indeed 1/2. Example 2 shows a similar fact for JIS with unit capacities.

Example 1. Consider an instance $\mathcal{I}$ of JI with unit capacities depicted in

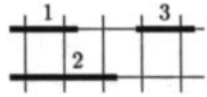

Figure 2.4: Example 1.

Figure 2.4. All the interval weights w_i, $i = 1, 2, 3$, are unit. Solution A

returned by the algorithm *ALG1* is $\{1\}$, while the optimal solution to JIP is $\{2,3\}$. Thus the value of A is $1/2$ times the optimal value.

Example 2. Consider an instance in Figure 2.5, consisting of k simple

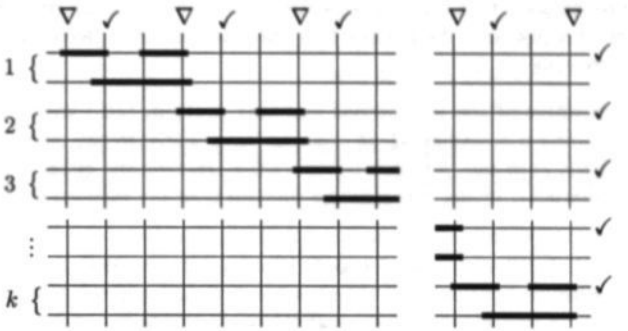

Figure 2.5: Example 2

repeating configurations, where all the intervals have unit weight. In this case the solution (z, y) returned by *ALG1* as well as the optimal solution to JIS (z^{opt}, y^{opt}) are 0,1-vectors. The elements corresponding to 1 (columns or rows) in (z, y) are marked in the picture with "✓" and the elements corresponding to 1 in (z^{opt}, y^{opt}) are marked with "▽". It is easy to see that $value(z, y) = 2k$, $value(z^{opt}, y^{opt}) = k+1$. When k tends to infinity, the ratio tends to 2.

2.3 The Case of Unit Demands: Algorithm *ALG2*

In this section we focus on JI with unit demands. This special case of JI can be described as follows: given is a grid consisting of t columns, numbered consecutively from left to right, and m numbered rows together with a set of intervals $I = \{1, 2, ..., n\}$ lying on the rows of the grid. An interval i is specified by the triple (l_i, r_i, ρ_i), where l_i, r_i are the indexes of the left- and the right-most columns intersecting the interval and ρ_i is the index of the row where it lies. For each column c, row r and interval i, we are given positive integral parameters v_c, u_r and p_i respectively, referred to as the column, row and interval capacities. We assume that the intervals are ordered according to nondecreasing r_i.

The objective of the job interval stabbing problem (JIS) with unit demands is: *find a collection H of columns, rows and intervals of minimum total capacity, such that for each interval $i \in I$, H contains either a column stabbing interval i, or the row stabbing interval i, or the interval i itself.*

Notice that JIS with unit demands is a special case of the well known *weighted hitting set* problem (WHS): *given is a finite weighted ground set $\mathcal{E}$, each element $e \in \mathcal{E}$ having a nonnegative weight w_e, and a collection of its subsets $\mathcal{S}$. Find a minimum weight subset $H \subseteq \mathcal{E}$ such that $H \cap s \neq \emptyset$ for any $s \in \mathcal{S}$.*

Indeed, let the set of all the columns, rows and intervals with their weights constitute a weighted ground set $\mathcal{E}$. Let $\mathcal{S}$ be $\{s_1, s_2, .., s_n\}$, where subset s_i contains the interval i together with the subset of the columns and the row stabbing it. Then any feasible hitting set corresponds to a feasible solution to JIS with unit demands of the same value.

In the spirit of WHS we say that an interval i is *hit* by a subset H of columns, rows and intervals if H contains either column or row stabbing interval i, or interval i itself.

Below is given an ILP formulation of JIS with unit demands (for the decision variables we use here characteristic vector (y, z, s) of H, where $y \in \mathbb{Z}^t$ is associated with the set of columns, $z \in \mathbb{Z}^m$ with the set of rows and $s \in \mathbb{Z}^n$ with the set of intervals)

$$\begin{array}{lll} \text{JIS: Minimize} & \sum_{c=1}^{t} v_c y_c + \sum_{r=1}^{m} u_r z_r + \sum_{i=1}^{n} p_i s_i & (2.3.1) \\ \text{subject to} & z_{\rho_i} + \sum_{c \in [l_i, r_i]} y_c + s_i \geq 1 \quad \forall i, & (2.3.2) \\ & z_r, y_c, s_i \in \{0,1\} \quad \forall r, c, i. & (2.3.3) \end{array}$$

The dual to its LP relaxation is

$$\begin{array}{lll} \text{Dual: Maximize} & \sum_{i=1}^{n} x_i & (2.3.4) \\ \text{subject to} & \sum_{i:\rho_i = r} x_i \leq u_r \quad \forall r = 1, .., m & (2.3.5) \\ & \sum_{i: c \in [l_i, r_i]} x_i \leq v_c \quad \forall c = 1, .., t & (2.3.6) \\ & x_i \leq p_i \quad \forall i = 1, .., n & (2.3.7) \\ & x_i \geq 0 \quad \forall i = 1, .., n & (2.3.8) \end{array}$$

Notice that when replacing the nonnegativity constraints (2.3.8) with integrality constraints $x_i \in \mathbb{Z}\ \forall i = 1, .., n$, we obtain an ILP formulation of the job interval packing problem (JIP) with unit demands:

specify an integral multiplicity for each interval i, not exceeding its capacities p_i, such that
- for each column c or row r the sum of the multiplicities of the intervals sharing it does not exceed the capacity v_c or u_r respectively,
- the sum of the multiplicities is minimum.

Following the framework of the generic *primal-dual algorithm with reverse delete step* described for WHS by Goemans & Williamson [27], we develop a primal-dual algorithm for JIS with unit demands, called *ALG2*.

We use auxiliary variables $\hat{v} \in \mathbb{R}^t$, $\hat{u} \in \mathbb{R}^m$ and $\hat{p} \in \mathbb{R}^n$ which are in fact slack variables for the constraints (2.3.5)-(2.3.7) i.e., at each moment in time:

$$\hat{v}_c = v_c - \sum_{i: c \in [l_i, r_i]} x_i \ \ \forall c, \qquad \hat{u}_r = u_r - \sum_{i: \rho_i = r} x_i \ \ \forall r, \qquad \hat{p}_i = p_i - x_i \ \ \forall i \quad (2.3.9)$$

for some current values of x_i, $i = 1, ..., n$.

Initially the dual vector x is zero, the set H is empty and the slack variables $\hat{v}$, $\hat{u}$, $\hat{p}$ are equal to v, u and p respectively. For each interval i from 1 to n we check whether it is already hit by H, if not we do the following:

- assign to the dual variable x_i the minimum of the values of slack variables corresponding to the following elements: the columns stabbing interval i, its row and interval i itself, i.e., $x_i \leftarrow \min\{\hat{v}_{l_i}, .., \hat{v}_{r_i}, \hat{u}_{\rho_i}, \hat{p}_i\}$;

- update the slack variables (since the value of x_i changed). Due to the way x_i was updated, at least one of the slack variables $\hat{v}_{l_i}, .., \hat{v}_{r_i}, \hat{u}_{\rho_i}, \hat{p}_i$ has to be zero now;

- add to H the elements (columns $l_i, ..., r_i$, row ρ_i or interval i) whose corresponding slack variables are zero. Notice that at least one of these element has to be added to H and thus interval i becomes hit.

Clearly, after all the n intervals are processed as above, H is a feasible solution to JIS with unit demands, since all the intervals are hit. At the next stage we try to remove elements from H. For that we consider each element in H and remove it if feasibility of H is preserved. The order of considerations plays a role here. First we check the columns in the order reverse to the order they were added to H. Then all the other elements, i.e., rows and intervals, in an arbitrary order.

The formal description of algorithm *ALG2* is shown in Figure 2.6. We use three index sets to represent set H, the set of column, row and interval indexes H^{col}, H^{row} and H^{int} respectively.

Theorem 2.3.1. *For any instance $\mathcal{I}$ of JI with unit demands the sets $(H^{col}, H^{row}, H^{int})$ and vector x returned by the algorithm* ALG2 *describe feasible solutions to JIS and JIP respectively, and their values are related as follows:*

$$value(H^{col}, H^{row}, H^{int}) \leq 2\ value(x).$$

To prove it we need a preliminary lemma. This is a result for the weighted hitting set problem that can be also found in [27]:

Lemma 2.3.2. *Consider an instance of WHS. If a set $H \subset \mathcal{E}$, vector $\chi \in \mathbb{R}^{|\mathcal{S}|}$ and $\beta \geq 0$ satisfy:*

$$\forall e \in H: \sum_{s \ni e} \chi_s = w_e \quad \text{and} \quad \forall s \in \mathcal{S},\ \text{such that } \chi_s > 0: \ |s \cap H| \leq \beta$$

then

$$\sum_{e \in H} w_e \leq \beta \sum_{s \in \mathcal{S}} \chi_s.$$

Proof. Using the conditions of the Lemma we have:

$$\sum_{e \in H} w_e = \sum_{e \in H} \sum_{s \ni e} \chi_s = \sum_{s \in \mathcal{S}} |s \cap H| \chi_s \leq \beta \sum_{s \in \mathcal{S}} \chi_s. \qquad \square$$

1. $x \leftarrow \mathbf{0}$
2. $H^{col} \leftarrow \emptyset$, $H^{row} \leftarrow \emptyset$, $H^{int} \leftarrow \emptyset$
3. $\hat{v} \leftarrow v$, $\hat{u} \leftarrow u$, $\hat{p} \leftarrow p$
4. $l \leftarrow 0$
5. For $i \leftarrow 1$ to n
   If $([l_i, r_i] \cap H^{col} = \emptyset)$ *AND* $(\{\rho_i\} \cap H^{row} = \emptyset)$ then
   $x_i \leftarrow \min\{\hat{v}_{l_i}, \ldots, \hat{v}_{r_i}, \hat{u}_{\rho_i}, \hat{p}_i\}$;
   For $c \leftarrow l_i$ to r_i if $(\hat{v}_c \leftarrow \hat{v}_c - x_i) = 0$ then
   $l \leftarrow l + 1$, $c_l \leftarrow c$, $H^{col} \leftarrow H^{col} \cup \{c\}$
   If $(\hat{u}_{\rho_i} \leftarrow \hat{u}_{\rho_i} - x_i) = 0$ then $H^{row} \leftarrow H^{row} \cup \{\rho_i\}$
   If $(\hat{p}_i \leftarrow \hat{p}_i - x_i) = 0$ then $H^{int} \leftarrow H^{int} \cup \{i\}$
6. For $j \leftarrow l$ downto 1
   if $H^{col} - \{c_j\}$, together with H^{row} and H^{int}, is feasible
   then $H^{col} \leftarrow H^{col} - \{c_j\}$
7. For all $i \in H^{int}$
   if $H^{int} - \{i\}$ together with H^{col} and H^{row} is feasible
   then $H^{row} \leftarrow H^{row} - \{i\}$
8. For all $r \in H^{row}$
   if $H^{row} - \{r\}$, together with H^{col} and H^{int}, is feasible
   then $H^{row} \leftarrow H^{row} - \{r\}$
9. Return $H^{col}, H^{row}, H^{int}$ (and x)

Figure 2.6: Algorithm *ALG2*.

Proof.(of the Theorem) Obviousely, by construction the sets $(H^{col}, H^{row}, H^{int})$ describe a feasible solution to JIS with unit demands and the vector x is feasible to the dual LP formulation (2.3.5)-(2.3.7). Notice, that the integrality of the input data implies integrality of x. Thus x is feasible to JIP with unit demands.

Let us establish the relation between the solution values. Recall the representation of JIS with unit demands as a special case of WHS, described earlier in this section. We use the result of the lemma with $(H^{col}, H^{row}, H^{int})$ representing H, x representing χ, β equal to 2.

Let us show that the first condition of the lemma is satisfied. Recall that an index of an element (be that column, row or interval) is added to one of $(H^{col}, H^{row}, H^{int})$ only when the corresponding slack variable (2.3.9) becomes zero. After a slack variable becomes zero, it is not changed by the algorithm anymore. Thus at the end of the algorithm we have that all the slack variables (2.3.9) corresponding to the elements in the solution are zero, and thus the first condition of the lemma follows.

Let us establish the second condition, i.e., show that for $(H^{col}, H^{row}, H^{int})$ returned by the algorithm and for any $i = 1, ..., n$, such that $x_i > 0$ holds:

$$|\{l_i, .., r_i\} \cap H^{col}| + |\{\rho_i\} \cap H^{row}| + |\{i\} \cap H^{int}| \leq 2.$$

Take i s.t. $x_i > 0$. If we show that $|\{l_i, .., r_i\} \cap H^{col}| \leq 1$, the above bound follows easily from the fact that, due to the minimality of solution $(H^{col}, H^{row}, H^{int})$, accomplished in steps 6,7,8, $|\{i\} \cap H^{int}| = 0$ for any i, for which $|\{l_i, .., r_i\} \cap H^{col}| + |\{\rho_i\} \cap H^{row}| \geq 1$.

Suppose $|\{l_i, .., r_i\} \cap H^{col}| \geq 2$, i.e. H^{col} contains at least 2 columns incident to the interval i, say, columns c_1 and c_2, $c_1 < c_2$. Consider the moment right before x_i became positive, i.e. the beginning of the ith iteration at step 5 of the algorithm. All the previously considered intervals j, $j < i$, are already covered by this moment and nor c_1, neither c_2 are yet added to H^{col} (otherwise, $\{l_i, l_i + 1, .., r_i\} \cap H^{col} = \emptyset$ would not hold). Look now at step 6, the moment when we are considering column c_1 as a candidate for removal from H^{col}. We claim that all intervals j, $j < i$, are currently covered by other elements (columns, rows or intervals). This is due to the fact that no element, added to the solution during step 5 before column c_1, can be considered for removal in step 6 before c_1. Further, it is not difficult to see that all intervals l, $l \geq i$, stabbed by column c_1, have to be stabbed by column c_2 as well by the ordering of the intervals (see Figure 2.7). Therefore nothing can prevent us from removing c_1 from H^{col} in step 6.

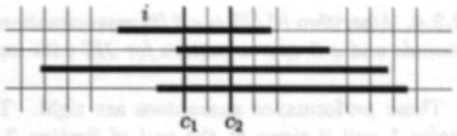

Figure 2.7: All the intervals l, $l \geq i$, incident to column c_1, have to be incident to column c_2 as well.

□

Theorem 2.3.3. *Algorithm* ALG2 *can be implemented to run in* $O(nt)$ *time.*

Proof. The initialization of the variables takes $O(n+t+m)$ time. Assuming that there is no row without intervals in the grid, and thus $m \leq n$, $O(n+t+m) = O(n+t)$.

The operations of taking the minimum at step 5 and updating the slack variables takes at most $O(t)$ time, thus the 5th step takes at most $O(nt)$ time in total.

Performing the feasibility check at step 6 requires some additional data structure. First, it is convenient to have indicators for each column, row and interval, showing whether the element is currently present in H. It takes only a constant time to update these indicators when adding and removing elements from H. Second, for each interval we maintain a counter, at each moment equal to the number of selected elements hitting that interval. We can establish those counters right after step 5 in $O(nt)$ time by checking for each interval the corresponding indicators. Then we can check whether removal of a column $c \in A^{col}$ does not violate the feasibility as follows: if all the values of the counters corresponding to the intervals incident to column c are greater than 1, then we can remove the column and update the counters. For all the columns $c \in A^{col}$ it takes at most $O(nt)$ time.

Step 7, with the help of the counters, is easy to implement in $O(n)$ time .

In step 8 let us perform for each interval the following check: if its counter equals 1 and its row is currently present in H, put a mark on this row. This means that the row can not be removed from H without disturbing feasibility. After all the intervals are checked remove from H^{row} the numbers of the rows which are not marked. This procedure takes at most $O(n)$ time. □

Theorem 2.3.1 together with the weak duality relation between JIP and JIS has the following consequence:

Corollary 2.3.4. *Algorithm* ALG2 *is a 1/2-approximation algorithm for JIP with unit demands and a 2-approximation for JIS with unit demands.*

Tightness. These performance guarantees are tight. To see that return to the Examples 1 and 2 given at the end of Section 2.2.2. Since in the instances described in those examples all the data (capacities and demands) are unit, we can apply to them algorithm *ALG2*. It turns out that the solutions returned by *ALG2* coincide with those produced by *ALG1*, which are described in the examples. Thus, the solution for JIP is exactly 1/2 times the optimum and the solution for JIS tends to be 2 times the optimum. This means that the performance guarantees stated in 2.3.4 can not be improved by a better analysis.

Remark 2.3.1. Notice that both algorithms *ALG2* and *ALG1* can be applied to instances of JI with unit capacities and demands. It is not difficult to verify, that the solutions for JIP (subsets of intervals) returned by the two algorithms coincide, while the solutions to JIS (subsets of columns and rows) can be different.

Chapter 3

Approximation of the job interval stabbing problem (JIS)

3.1 Introduction

This chapter is devoted entirely to approximation algorithms for the job interval stabbing problem based on LP-rounding.

Preliminaries. Let us give a formulation of the job interval stabbing problem (JIS): *given is a grid consisting of t columns, numbered consecutively from left to right, and m numbered rows. Further given is a set of intervals $I = \{1, 2, ..., n\}$ lying on the rows of the grid. An interval i is specified by the triple (l_i, r_i, ρ_i), where l_i, r_i are the indices of the left- and the right-most column stabbing interval i and ρ_i is the index of the row stabbing interval i. For each column c, row r and interval i, given are positive integral weights, v_c, u_r and p_i respectively. For each interval i we are given a positive integral parameter w_i referred to as the interval demand. We assume that the intervals are ordered according to nondecreasing r_i. The task is to specify for each column c, row r and interval i integral multiplicities y_c, z_r and s_i respectively, such that:*

- for each interval i the sum of multiplicities of the columns and the row stabbing interval i and the multiplicity of interval i equals at least its demand w_i,

- the total weight $\sum_{c=1}^{t} v_c y_c + \sum_{r=1}^{m} u_r z_r + \sum_{i=1}^{n} p_i s_i$, is minimized.

Notice that here, in contrast to the previous chapter, we refer to parameters v_c, u_r and p_i as weights (and not as capacities).

JIS is MAX SNP-hard even if all the column, row and interval weights and interval demands are equal to 1: $v_c, u_r, p_i, w_i = 1$, $\forall c, r, i$ (see Section 4.3 of this thesis). This implies that no polynomial time approximation scheme (PTAS) can exists for this special case unless $\mathcal{P} = \mathcal{NP}$.

Previous research. In [31] Hassin and Megiddo present an investigation of a range of special cases of the following problem: given n compact subsets of $\mathbb{R}^d$, find a set of straight lines of minimum cardinality so that each of the given subsets is hit by at least one line. They introduce an approximation algorithm with a performance guarantee of at most $2 - \frac{1}{K+1}$ for a problem, which in our context may be described as a special case of JIS, where each of the intervals is stabbed by exactly K columns and all the numerical parameters v_c, u_r, p_i, w_i, $\forall c, r, i$ are unit. For a slightly more general case, when each of the intervals is stabbed by the same number of columns, a 2-approximation algorithm is given.

In [26] Gaur et al. describe a 2-approximation algorithm for the so-called *rectangle stabbing* problem. This problem appears when instead of intervals we are given rectangles, each of which may be stabbed by several rows as well as by several columns. The task is to stab all the rectangles by a minimum weight subset of columns and rows. This problem is a generalization of JIS with unit demands.

Our contribution and outline of the chapter. This chapter improves the results of [39], where we describe a $(2+\epsilon)$-approximation algorithm for JIS, based on an LP-rounding technique, using a coloring idea, initially introduced by Bar-Noy et al. [8].

In section 3.2 we show that, when the vector of unit multiplicities z is fixed in advance, JIS can be formulated as a minimum cost flow problem. This observation is used in Section 3.3, which focuses on the problem JIS^{w_0}, which is JIS with all the demands greater or equal to w_0: $w_i \geq w_0$, $\forall i = 1, ..., n$. We describe an LP-rounding $(w_0+1)/w_0$-approximation algorithm for JIS^{w_0}. Notice that this implies a 2-approximation algorithm for the general version of JIS, since all the demands are positive integral numbers, and thus at least 1. This algorithm generalizes the result of Gaur et al. [26] (implied for intervals) to the case of arbitrary demands. In addition it requires less computational effort since we observe, that some of the LP problems that have to be solved in the algorithm of [26], may be solved combinatorially via a minimum cost flow algorithm.

In Section 3.4 we consider JIS with unit demands and describe an LP-

rounding $e/(e-1) \approx 1.582$-approximation algorithm. We also show that the integrality gap factor of the natural ILP formulation of JIS with unit demands is arbitrarily close to $e/(e-1)$, making existence of an LP-rounding algorithm for this problem with a better approximation ratio unlikely.

3.2 Reduced Problem JIS($\bar{z}$)

In this section we consider the problem JIS given that the vector of the row multiplicities z is fixed in advance: $z = \bar{z}$, $\bar{z} \in \mathbf{Z}_+^m$. We denote this problem by JIS($\bar{z}$) and refer to it as the *reduced problem*. We show that JIS($\bar{z}$) can be solved to optimality via a minimum cost flow algorithm.

Consider a straightforward ILP formulation of JIS($\bar{z}$):

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum_{c=1}^{t} v_c y_c + \sum_{i=1}^{n} p_i s_i && && (3.2.1)\\ \text{subject to} \quad & \textstyle\sum_{c \in [l_i, r_i]} y_c + s_i \geq w_i - \bar{z}_{p_i} && \forall i = 1, .., n && (3.2.2)\\ & y_c, s_i \in \mathbf{Z}_+^1 && \forall c, i. && (3.2.3) \end{aligned}$$

The LP relaxation is obtained by replacing constraints (3.2.3) with

$$y_c \geq 0, s_i \geq 0, \ \forall c, i. \tag{3.2.4}$$

Reformulating (3.2.1)-(3.2.3) in matrix notation, using $b_i = w_i - \bar{z}_{p_i}$, $\forall i = 1, ..., n$, yields:

$$\begin{aligned} \text{Minimize} \quad & vy + ps \\ \text{subject to} \quad & Ay + Is \geq b \\ & y \in \mathbf{Z}_+^t, s \in \mathbf{Z}_+^n \end{aligned} \tag{3.2.5}$$

Here $I \in \{0,1\}^{n \times n}$ is an identity matrix, $A \in \{0,1\}^{n \times t}$, $b \in \mathbf{Z}_+^n$. Notice that the matrix A has the so called "consecutive ones" property, i.e., in the columns of A '1'-s are positioned consecutively, without interfering '0'-s. This property implies that A is a totally unimodular matrix (see [45]). Since a totally unimodular matrix combined with an identity matrix retains the property of total unimodularity (see e.g. [45]) we conclude that the constraint matrix of formulation (3.2.1)-(3.2.3) is totally unimodular and therefore the associated LP relaxation is integral, i.e., describes an integral polyhedron.

Proposition 3.2.1. *The linear programming problem described by (3.2.1), (3.2.2) and (3.2.4) is integral.*

Of course, this implies that the problem JIS($\bar{z}$) can be solved in polynomial time by solving the LP relaxation of its ILP formulation. Now we explain how its special structure allows us to solve it more efficiently than a general linear programming problem.

Lemma 3.2.2. *The problem JIS($\bar{z}$) can be solved in time $O(MCF(t, n+t))$, where $MCF(t,p)$ is the time needed to solve a minimum cost flow problem on a network with t nodes and p arcs.*

Proof. Let us formulate the dual to the LP relaxation of (3.2.5), where we use a dual vector $x \in \mathbb{R}^n_+$:

$$\begin{array}{ll} \text{Maximize} & bx \\ \text{subject to} & A^T x \le v \\ & x \le p \\ & x \ge 0 \end{array}$$

Using a vector of auxiliary variables $q \in \mathbb{R}^t_+$ we can rewrite it as follows:

$$\begin{array}{ll} \text{Maximize} & bx \\ \text{subject to} & A^T x + Iq = v \\ & x \le p \\ & x, q \ge 0 \end{array}$$

Consider the composite matrix $A^T|I$. Denote its rows by $a_1, a_2, ..., a_t$ and transform the matrix $A^T|I$ as follows (this transformation is also shown in in [3], see pages 304-306):

$$\begin{pmatrix} -a_1 \\ a_1 - a_2 \\ \dots \\ a_{t-1} - a_t \\ a_t \end{pmatrix} \tag{3.2.6}$$

Notice that this matrix has $t+1$ rows. Now we can rewrite our LP in the following equivalent form:

$$\begin{aligned} &\text{Minimize} && -bx \\ &\text{subject to} && \begin{pmatrix} -a_1 \\ a_1 - a_2 \\ \dots \\ a_{t-1} - a_t \\ a_t \end{pmatrix} \begin{pmatrix} x \\ q \end{pmatrix} = \begin{pmatrix} -v_1 \\ v_1 - v_2 \\ \dots \\ v_{t-1} - v_t \\ v_t \end{pmatrix} \\ & && x \leq p \\ & && x, q \geq 0 \end{aligned} \tag{3.2.7}$$

This is a minimum cost flow formulation. Indeed, the matrix (3.2.6) has exactly one '-1' and one '+1' in each column; this follows from the fact that matrix $A^T|I$ has consecutive ones in columns. Therefore we can solve this problem via a minimum cost flow algorithm.

Let us write down the dual of (3.2.7). There are $t+1$ dual variables $y'_0, \dots, y'_t$ and n dual variables $s_1, \dots, s_n$:

$$\begin{aligned} &\text{Maximize} && \begin{pmatrix} -v_1 \\ v_1 - v_2 \\ \dots \\ v_{t-1} - v_t \\ v_t \end{pmatrix} y' - ps \\ &\text{subject to} && y'_{l_i - 1} - y'_{r_i} + s_i \geq b_i, \quad \forall i = 1, \dots, n \\ & && y'_{c-1} - y'_c \geq 0, \quad \forall c = 1, \dots, t \\ & && s \geq 0 \end{aligned} \tag{3.2.8}$$

Given an optimal solution to the minimum cost flow problem, one can easily obtain (in time dominated by the time necessary to solve a min-cost flow problem) the optimal values of the dual variables $y'_0, \dots, y'_t$ and $s_1, \dots, s_n$ (see [3]).

It is easy to verify that by substituting $\sum_{i=c+1}^{t} y_i$ for y'_c, $\forall c = 0, \dots, t$ in (3.2.8) we obtain formulation (3.2.1),(3.2.2),(3.2.5), where $w_i - \bar{z}_{\rho_i} = b_i$. Therefore (y, s) is an optimum solution to JIS($\bar{z}$) where

$$y_c = y'_{c-1} - y'_c, \ \forall c = 1, .., t.$$

□

3.3 Approximation of JIS^{w_0}

Let us consider a more specific version of JIS: let JIS^{w_0} be JIS, where the interval demands are greater or equal to w_0, i.e., $w_i \geq w_0$ for all $i = 1, ..., n$. In this section we describe an $(w_0+1)/w_0$-approximation algorithm for JIS^{w_0}.

Let us give a straightforward ILP formulation of JIS^{w_0}:

$$\text{Minimize} \quad \sum_{c=1}^{t} v_c y_c + \sum_{r=1}^{m} u_r z_r + \sum_{i=1}^{n} p_i s_i \qquad (3.3.1)$$

$$\text{subject to} \quad z_{\rho_i} + \sum_{c \in [l_i, r_i]} y_c + s_i \geq w_i \qquad \forall i = 1, .., n \quad (3.3.2)$$

$$z_r, y_c, s_i \in \mathbf{Z}_+^1 \qquad \forall r, c, i. \qquad (3.3.3)$$

Its LP relaxation can be obtained by replacing integrality constraints (3.3.3) with nonnegativity constraints

$$z_r \geq 0, y_c \geq 0, s_i \geq 0, \ \forall r, c, i. \qquad (3.3.4)$$

The algorithm *ROUND* for the problem JIS^{w_0} can informally be described as follows. Given an instance of JIS^{w_0}, it solves the LP relaxation of formulation (3.3.1)-(3.3.3), obtaining an optimal LP solution z^{lp}, y^{lp}, s^{lp}. Then, it multiplies the values of the z-variables by $(w_0+1)/w_0$ and rounds all of them down. Denote the resulting integral vector by $\bar{z}$. Finally, find an optimal solution of JIS($\bar{z}$) $\bar{y}, \bar{s}$ and return $\bar{z}, \bar{y}, \bar{s}$.

The formal description of the algorithm may be found in Figure 3.1.

1. solve the LP relaxation of (3.3.1)-(3.3.3) and obtain its optimal solution y^{lp}, z^{lp}, s^{lp};
2. for all $i = 1$ to m $\quad \bar{z}_i \leftarrow \lfloor (w_0 + 1)/w_0 \cdot z_i^{lp} \rfloor$,
3. solve JIS($\bar{z}$), obtain $\bar{y}, \bar{s}$;
4. return $\bar{y}, \bar{z}, \bar{s}$

Figure 3.1: Algorithm *ROUND*.

Theorem 3.3.1. *For any instance of JIS^{w_0} algorithm* ROUND *returns a feasible solution with a value of at most $(w_0 + 1)/w_0$ times the optimum.*

Proof. Consider an instance $\mathcal{I}$ of JIS^{w_0}. The feasibility of the solution returned by *ROUND* for $\mathcal{I}$ is obvious, since for any integer z an optimal

solution to JIS(z), together with z, constitutes by construction a feasible solution to JIS (see section 3.2).

To establish the performance guarantee we compare the value of the returned solution $(\bar{y}, \bar{z}, \bar{s})$ with the value of the optimum LP solution (y^{lp}, z^{lp}, s^{lp}). The crucial part here is to establish that $\left(\frac{w_0+1}{w_0}\, y^{lp},\ \frac{w_0+1}{w_0}\, s^{lp}\right)$ is a feasible solution to the LP relaxation of JIS($\bar{z}$). Indeed, consider any interval i. From the feasibility of z^{lp}, y^{lp}, s^{lp} we have:

$$z^{lp}_{\rho_i} + \sum_{c=l_i}^{r_i} y^{lp}_c + s^{lp}_i \geq w_i.$$

Multiplying both sides of the inequality by $\frac{(w_0+1)}{w_0}$ gives:

$$\frac{(w_0+1)}{w_0}\, z^{lp}_{\rho_i} + \frac{(w_0+1)}{w_0} \sum_{c=l_i}^{r_i} y^{lp}_c + \frac{(w_0+1)}{w_0} s^{lp}_i \geq \frac{(w_0+1)}{w_0} w_i \geq w_i + 1,$$

where the last inequality holds because $w_i \geq w_0$. Rounding down one of the terms decreases the left-hand side by at most 1:

$$\left\lfloor \frac{(w_0+1)}{w_0}\, z^{lp}_{\rho_i} \right\rfloor + \frac{(w_0+1)}{w_0} \sum_{c=l_i}^{r_i} y^{lp}_c + \frac{(w_0+1)}{w_0} s^{lp}_i \geq w_i.$$

Recall that $\left\lfloor \frac{(w_0+1)}{w_0} \cdot z^{lp}_i \right\rfloor = \bar{z}_i,\ \forall i = 1, .., m$. Thus:

$$\frac{(w_0+1)}{w_0} \sum_{c=l_i}^{r_i} y^{lp}_c + \frac{(w_0+1)}{w_0} s^{lp}_i \geq w_i - \bar{z}_{\rho_i}.$$

From this we conclude that $\left(\frac{w_0+1}{w_0}\, y^{lp},\ \frac{w_0+1}{w_0}\, s^{lp}\right)$ is a feasible solution to the LP relaxation of JIS($\bar{z}$) for $\mathcal{I}$.

Proposition 3.2.1 implies that the optimal value of JIS($\bar{z}$) is less or equal to the value of any feasible solution to its LP relaxation. Therefore it follows:

$$v\bar{y} + p\bar{s} \leq v \cdot \frac{(w_0+1)}{w_0} y^{lp} + p \cdot \frac{(w_0+1)}{w_0} s^{lp} = \frac{(w_0+1)}{w_0} (vy^{lp} + ps^{lp}).$$

Step 2 of the algorithm (see Figure 3.1) implies that $\bar{z}_i \leq \frac{(w_0+1)}{w_0} z^{lp}_i,\ \forall i$. Thus:

$$u\bar{z} \leq \frac{(w_0+1)}{w_0}\, uz^{lp}.$$

Then the value of the solution output by *ROUND* is

$$v\bar{y} + p\bar{s} + u\bar{z} \leq \frac{(w_0+1)}{w_0}(vy'^{p} + ps'^{p} + uz'^{p}) \leq \frac{(w_0+1)}{w_0} OPT,$$

where OPT is the optimum value of JIS^{w_0} for $\mathcal{I}$. □

Lemma 3.3.2. *The running time of algorithm* ROUND *is* $O(ULP(n, t+n+m) + MCF(n, t+n))$, *where* $ULP(n, t+n+m)$ *is the time needed to solve a linear programming problem with* $n \times (t+n+m)$ *matrix with 0,1 entries*[1] , *and* $MCF(t, t+n)$ *is the time needed to solve a minimum cost flow problem on a network with* t *nodes and* $t+n$ *arcs.*

Proof. Obvious. □

Corollary 3.3.3. *Algorithm* ROUND *is a* $(w_0+1)/w_0$*-approximation algorithm.*

Tightness. So far we have not found an instance proving the tightness of the result of Theorem 3.3.1, namely, an instance providing the ratio of $(w_0+1)/w_0$ between the optimum value of JIS^{w_0} and the value of the solution returned by the algorithm.

However, we can present an instance of JIS^1 and a feasible LP solution for it, very close to the optimum LP solution, whose value is increased with a factor arbitrarily close to 2 by the rounding procedure of the algorithm. This implies that the steps 2, 3 and 4 of *ROUND* (see Figure 3.1) can indeed double the value of a given fractional solution.

Consider the instance given in Figure 3.2, assuming that all the parameters are unit. An optimal LP solution for JIS^1 consists of values of 1/2 attached to the first row and all the columns of the grid. We perturb this fractional solution by adding an arbitrary small positive ε to the y-values and subtracting ε from the positive z-variable (see the brackets in Figure 3.2). Notice that it maintains the feasibility of the solution.

The rounding procedure of the algorithm (steps 2-4), applied to this slightly altered fractional solution, rounds the positive z-variable down to 0 and finds the optimum solution $(\bar{y}, \bar{s})$, that consists of values 1 attached to each column. So the value of this rounded solution is $2k$, whereas the value

[1] In view of the result of Tardos [48] this time only depends on the dimensions of the constraint matrix and does not depend on the input size of the right-hand side and objective vectors.

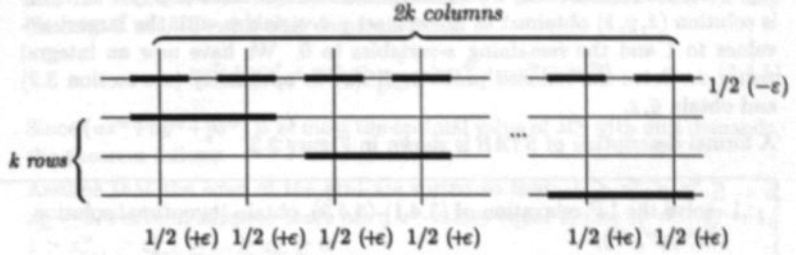

Figure 3.2: Example.

of the initial fractional solution is $(2k(1/2+\varepsilon)+1/2-\varepsilon)$. If we increase k and correspondingly decrease ε, the ratio between them tends to 2.

The discussion above suggests that it might be possible to guarantee a stronger performance of the algorithm when using a certain structure of an optimal LP solution in the analysis.

3.4 Approximation of JIS with unit demands

In this section we present an $e/(e-1) \approx 1.582$-approximation algorithm for JIS with unit demands, called *STAB*.

Recall that JIS with unit demands is JIS where all the interval demands w_i equal 1. Let us give its ILP formulation:

$$\begin{array}{lll} \text{Minimize} & \sum_{c=1}^{t} v_c y_c + \sum_{r=1}^{m} u_r z_r + \sum_{i=1}^{n} p_i s_i & (3.4.1) \\ \text{subject to} & z_{\rho_i} + \sum_{c \in [l_i, r_i]} y_c + s_i \geq 1 \qquad \forall i = 1, .., n & (3.4.2) \\ & z_r, y_c, s_i \in \mathbb{Z}_+^1 \qquad \forall r, c, i. & (3.4.3) \end{array}$$

The LP relaxation arises when we replace the integrality constraints (3.4.3) by

$$z_r \geq 0, y_c \geq 0, s_i \geq 0, \ \forall r, c, i.$$

Informally algorithm *STAB* can be described as follows. Given an instance of *JIS* with unit demands, it solves the LP relaxation of formulation (3.4.1)-(3.4.3) and obtains an optimal LP solution (z^{lp}, y^{lp}, s^{lp}). It outputs the best of

$m+1$ candidate solutions, where candidate solution number j ($j = 0, ..., m$) is solution $(\bar{z}, \bar{y}, \bar{s})$ obtained as follows: set j z-variables with the largest z^{lp}-values to 1 and the remaining z-variables to 0. We have now an integral vector $\bar{z}$. Solve the reduced problem JIS($\bar{z}$) to optimality (see section 3.2) and obtain $\bar{y}, \bar{s}$.

A formal description of *STAB* is shown in Figure 3.3.

1. solve the LP relaxation of (3.4.1)-(3.4.3), obtain its optimal solution (z^{lp}, y^{lp}, s^{lp});
2. reindex the rows of the grid so that $z_1^{lp} \geq z_2^{lp} \geq ... \geq z_m^{lp}$;
3. $V \leftarrow \infty$;
4. for $j = 0$ to m
   for $i = 1$ to j $\bar{z}_i \leftarrow 1$,
   for $i = j+1$ to m $\bar{z}_i \leftarrow 0$.
   solve JIS($\bar{z}$), obtain $\bar{y}, \bar{s}$;
   if $u\bar{z} + v\bar{y} + p\bar{s} < V$ then $V \leftarrow u\bar{z} + v\bar{y} + p\bar{s}$, $z^* \leftarrow \bar{z}, y^* \leftarrow \bar{y}, s^* \leftarrow \bar{s}$
5. return y^*, z^*, s^*

Figure 3.3: Algorithm *STAB*.

Theorem 3.4.1. *For any instance of JIS with unit demands algorithm* STAB *returns a feasible solution with the value of at most* $e/(e-1)$ *times the optimum.*

Before giving the proof of the theorem let us introduce a preliminary lemma, the proof of which may be found in the appendix to this chapter.

Lemma 3.4.2. *Suppose we are given numbers* $1 > \Delta_1 \geq \Delta_2 \geq ... \geq \Delta_m \geq 0$, $\forall i = 1, .., m$, *and* $\Delta_{m+1} = 0$. *Further given are positive numbers* $a_1, a_2, ..., a_m$ *and* Y. *Then we have:*

$$\min_{j=1,...,m+1} \left(\sum_{r=1}^{j-1} a_r + \frac{1}{1-\Delta_j} Y\right) \leq \frac{e}{e-1} \left(\sum_{r=1}^{m} a_r \Delta_r + Y\right). \tag{3.4.4}$$

Proof.(of the theorem.) Consider an instance $\mathcal{I}$ of JIS with unit demands and let (z^{lp}, y^{lp}, s^{lp}) and (z^*, y^*, s^*) be respectively an optimal LP solution

and the solution returned by the algorithm for $\mathcal{I}$. We show that for any instance $\mathcal{I}$ of JIC with unit demands holds:

$$uz^* + vy^* + ps^* \leq \frac{e}{e-1}(uz^{lp} + vy^{lp} + ps^{lp}). \tag{3.4.5}$$

Since $(uz^{lp}+vy^{lp}+ps^{lp})$ is at most the optimal value of JIS with unit demands, the theorem follows.

Assume that the rows of the grid are sorted so that: $1 \geq z_1^{lp} \geq z_2^{lp} \geq ... \geq z_m^{lp} \geq 0$. Further, suppose there are q z^{lp}-values equal 1, i.e., $z_1^{lp} = ...z_q^{lp} = 1$, $1 > z_{q+1}^{lp} \geq z_{q+2}^{lp} \geq ... \geq z_m^{lp} \geq 0$.

From the design of the algorithm we know that:

$$uz^* + vy^* + ps^* = \min_{j\in\{0,...,m\}}(uz^j + vy^j + ps^j) \leq \min_{j\in\{q,...,m\}}(uz^j + vy^j + ps^j), \tag{3.4.6}$$

where $(\bar{z}^j, \bar{y}^j, \bar{s}^j)$, $\forall j \in \{0, ..., m\}$, is the candidate solution number j constructed by the algorithm applied to $\mathcal{I}$.

Claim 1. For any $j \in \{q, ..., m\}$:

$$u\bar{z}^j + v\bar{y}^j + p\bar{s}^j \leq \sum_{r=1}^{j} u_r + \frac{1}{1 - z_{j+1}^{lp}}(vy^{lp} + ps^{lp}).$$

Let us prove it. Take some $j \in \{q, ..., m\}$ and consider $(\bar{z}^j, \bar{y}^j, \bar{s}^j)$. By construction:

- $\bar{z}_r^j = 1$, $\forall r \leq j$,
- $\bar{z}_r^j = 0$, $\forall r \geq j+1$,
- $(\bar{y}^j, \bar{s}^j)$ is an optimal solution to the reduced problem JIS($\bar{z}^j$).

Claim 1.1. $u\bar{z}^j \equiv \sum_{r=1}^{m} u_r\bar{z}_r^j = \sum_{r=1}^{j} u_r$. Obvious.

Claim 1.2.

$$v\bar{y}^j + p\bar{s}^j \leq \frac{1}{1 - z_{j+1}^{lp}}(vy^{lp} + ps^{lp}).$$

To prove that, we establish that the fractional solution

$$\left(\frac{1}{1 - z_{j+1}^{lp}}y^{lp}, \quad \frac{1}{1 - z_{j+1}^{lp}}s^{lp}\right), \tag{3.4.7}$$

where we set $z_{m+1}^{lp} = 0$, is feasible to the LP relaxation of JIS($\bar{z}^j$). Then Proposition 3.2.1 implies that the value of $(\bar{y}^j, \bar{s}^j)$ is less than or equal to the value of solution (3.4.7), i.e, the statement of Claim 1.2.

Let us establish the feasibility of (3.4.7) to JIS($\bar{z}^j$). Consider any interval i, $i \in \{1, ..., n\}$, and show that the following constraint is satisfied:

$$\frac{1}{1 - z^{lp}_{j+1}} \sum_{c \in [l_i, r_i]} y^{lp}_c + \frac{1}{1 - z^{lp}_{j+1}} s^{lp}_i \geq 1 - \bar{z}^j_{\rho_i}. \tag{3.4.8}$$

In case $\bar{z}^j_{\rho_i} = 1$ the inequality trivially holds. Otherwise, if $\bar{z}^j_{\rho_i} = 0$, from the construction of $\bar{z}^j$ follows that $\rho_i \geq j + 1$. The ordering of the z^{lp}-values implies that $z^{lp}_{\rho_i} \leq z^{lp}_{j+1}$. Then, using this and feasibility of LP solution (z^{lp}, y^{lp}, s^{lp}) the left-hand side of (3.4.8) can be rewritten and bounded as follows:

$$\frac{1}{1 - z^{lp}_{j+1}} \Big(\sum_{c \in [l_i, r_i]} y^{lp}_c + s^{lp}_i \Big) \geq \frac{1}{1 - z^{lp}_{j+1}} (1 - z^{lp}_{\rho_i}) \geq \frac{1}{1 - z^{lp}_{j+1}} (1 - z^{lp}_{j+1}) \geq 1,$$

which implies (3.4.8). Therefore constraint (3.4.8) is satisfied for any i, $i \in \{1, ..., n\}$, and solution (3.4.7) is feasible to the LP relaxation of JIS($\bar{z}^j$). This proves Claim 1.2 and Claim 1.

From (3.4.6) and Claim 1:

$$uz^* + vy^* + ps^* \leq \min_{j=q,...,m} \left(\sum_{r=1}^{j} u_r + \frac{1}{1 - z^{lp}_{j+1}} (vy^{lp} + ps^{lp}) \right).$$

Using index $p = j - q$ we can rewrite the right-hand side as:

$$\min_{p=0,..,m-q} \left(\sum_{r=1}^{q+p} u_r + \frac{(vy^{lp} + ps^{lp})}{1 - z^{lp}_{p+q+1}} \right) = \min_{p=0,..,m-q} \left(\sum_{r=1}^{q} u_r + \sum_{r=q+1}^{q+p} u_r + \frac{(vy^{lp} + ps^{lp})}{1 - z^{lp}_{p+q+1}} \right) =$$

$$= \min_{p=0,..,m-q} \left(\sum_{r=1}^{p} u_{r+q} + \frac{1}{1 - z^{lp}_{p+q+1}} (vy^{lp} + ps^{lp}) \right) + \sum_{r=1}^{q} u_r.$$

Apply now Lemma 3.4.2, assuming $(vy^{lp} + ps^{lp}) = Y$, $z_{p+q} = \Delta_p$, $\forall p = 0, ..., m - q$ and $u_{r+q} = a_r$, $\forall r = 1, ..., m - q$. Then the last expression can be bounded from above by

$$\frac{e}{e-1} \left(\sum_{r=1}^{m-q} u_{r+q} z^{lp}_{r+q} + (vy^{lp} + ps^{lp}) \right) + \sum_{r=1}^{q} u_r \leq$$

$$\leq \frac{e}{e-1}\left(\sum_{r=1}^{m-q} u_{r+q} z_{r+q}^{lp} + \sum_{r=1}^{q} u_r + (vy^{lp} + ps^{lp})\right).$$

Since $z_1 = ... = z_q = 1$, the last expression can be rewritten as

$$\frac{e}{e-1}\left(\sum_{r=q+1}^{m} u_r z_r^{lp} + \sum_{r=1}^{q} u_r z_r^{lp} + (vy^{lp} + ps^{lp})\right) = \frac{e}{e-1}(uz^{lp} + vy^{lp} + ps^{lp}),$$

which establishes inequality (3.4.5).

□

Theorem 3.4.3. *The running time of algorithm* STAB *is* $O(ULP(n, t+n+m) + mMCF(n, t+n))$, *where* $ULP(n, t+n+m)$ *is time needed to solve a linear programming problem with* $n \times (t+n+m)$ *matrix with 0,1 entries*[2] , *and* $MCF(t, t+n)$ *is the time needed to solve a minimum cost flow problem on a network with* t *nodes and* $t+n$ *arcs.*

Proof. The reduced problem has to be solved m times, and the solution of it costs $O(MCF(n, t+n))$ time (see Lemma 3.2.2). □

Corollary 3.4.4. *Algorithm* STAB *is a* $\frac{e}{e-1} \approx 1.582$- *approximation algorithm for JIS with unit demands.*

Integrality gap factor. We show that the integrality gap factor of the straightforward ILP formulation of JIS with unit demands (3.4.1)-(3.4.3) can be arbitrary close to $e/(e-1)$. This makes any other LP-rounding algorithm using the same ILP formulation unlikely to have a performance guarantee better than $e/(e-1)$.

Theorem 3.4.5. *There exists a family of instances* $\{\mathcal{I}_m\}_{m \in \mathbf{N}}$ *of JIS with unit demands, such that the optimal value of JIS on* $\mathcal{I}_m$ *tends to* $e/(e-1)$ *times the optimal value of the LP relaxation of (3.4.1)-(3.4.3) as* m *tends to* ∞.

Proof. For each $m \in \mathbf{N}$ we describe an instance $\mathcal{I}_m$ as follows. The grid has m rows and $t = m!$ columns. There are r intervals on row r numbered

[2] In view of the result of Tardos [48] this time only depends on the dimensions of the constraint matrix and does not depend on the input size of the right-hand side and objective vectors.

$r_1, ..., r_r$. For an interval r_j we have: $\rho_{r_j} = r$, $l_{r_j} = \frac{m!}{r}(j-1)+1$ and $r_{r_j} = \frac{m!}{r}j$. So, the intervals placed on row r do not overlap with each other and each of them intersects exactly $\frac{m!}{r}$ columns of the grid (see Figure 3.4). All column, row and interval weights are unit. The total number of intervals in the instance is $n = 1 + 2 + ... + m$.

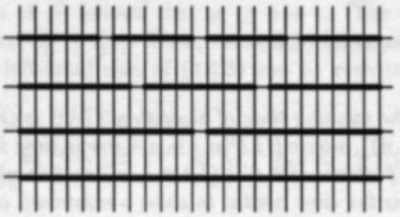

Figure 3.4: Instance $\mathcal{I}_4$.

We claim, that the following solution is optimal to the LP relaxation of (3.4.1)- (3.4.3) for $\mathcal{I}_m$:

$$\begin{aligned} z_r &= \begin{cases} 0, & \forall r = 1, ..., P \\ 1 - P/r, & \forall r = P+1, ..., m \end{cases} \\ y_c &= \frac{P}{m!}, \ \forall c = 1, ..., m!, \\ s_i &= 0, \ \forall i = 1, ..., n. \end{aligned} \tag{3.4.9}$$

Here $P = P(m)$ is the number, satisfying:

$$\frac{1}{m} + \frac{1}{m-1} + ... + \frac{1}{P+1} \leq 1 \text{ and } \frac{1}{m} + \frac{1}{m-1} + ... + \frac{1}{P+1} + \frac{1}{P} \geq 1.$$

It is easy to verify, that the value of this solution is:

$$value(z, y, s) = \sum_{c=1}^{t} y_c + \sum_{r=1}^{m} z_r + \sum_{i=1}^{n} s_i = m - P(\frac{1}{P+1} + \frac{1}{P+2} + ... + \frac{1}{m}).$$

First, we show that it is a feasible LP solution. Take any interval r_j and show that the constraint $z_{\rho_{r_j}} + \sum_{c \in [l_{r_j}, r_{r_j}]} y_c + s_{r_j} \geq 1$ is satisfied. Substituting the

values of the variables and noticing that the z-values in our solution can be expressed as: $z_r = max(1 - \frac{P}{r}, 0), \forall r = 1, ..., m$, we establish the validity of the constraint:

$$max(1-\frac{P}{r_j},0)+\sum_{c\in[l_{r_j},r_{r_j}]}\frac{P}{m!} = max(1-\frac{P}{r_j},0)+\frac{m!}{r_j}\frac{P}{m!} = max(1-\frac{P}{r_j},0)+\frac{P}{r_j} \geq 1.$$

Now let us prove optimality of the LP solution (3.4.9) by presenting a feasible dual solution to the LP relaxation of (3.4.1)- (3.4.3) which has the same value. Below is given the dual problem:

$$\begin{array}{lll} \text{Maximize} & \sum_{i=1}^{n} x_i & \\ \text{subject to} & \sum_{i:p_i=r} x_i \leq 1 & \forall r = 1, .., m \\ & \sum_{i:c\in[l_i,r_i]} x_i \leq 1 & \forall c = 1, .., t \\ & 0 \leq x_i \leq 1 & \forall i = 1, .., n \end{array} \tag{3.4.10}$$

Consider the following dual solution:

$$x_{r_j} = \begin{cases} 1/r, & \text{if } r = P+1, ..., m, \\ 1 - (\frac{1}{m} + \frac{1}{m-1} + ... + \frac{1}{P+1}), & \text{if } r = P, \\ 0 & \text{otherwise.} \end{cases} \tag{3.4.11}$$

In words, we assign $1/r$ to the dual variables corresponding to the intervals on row $r = P+1, ..., m$, $1-(\frac{1}{m}+\frac{1}{m-1}+...+\frac{1}{P+1})$ to the variables corresponding to the intervals from row $r = P$, and 0 to all the other variables.

It is easy to verify that this solution is feasible to the problem (3.4.10) and that its value

$$value(x) = \sum_{i=1}^{n} x_i = m - P(\frac{1}{m} + \frac{1}{m-1} + ... + \frac{1}{P+1}),$$

is equal to the value of (3.4.9), which proves the optimality of the latter to the LP relaxation of (3.4.1)- (3.4.3).

Denote the optimal LP value for $\mathcal{I}_m$ by $LP(\mathcal{I}_m)$. We know that

$$LP(\mathcal{I}_m) = m - P(\frac{1}{m} + \frac{1}{m-1} + ... + \frac{1}{P+1}).$$

Denote by $OPT(\mathcal{I})$ the optimum value of JIS with unit demands for $\mathcal{I}$. Let us show that $OPT(\mathcal{I}_m) = m$.

We use the mathematical induction. It is clear that $OPT(\mathcal{I}_1) = 1$, since exactly one element (column, row or interval) is needed to stab the only interval present in the instance $\mathcal{I}_1$. Assume now that it is proven, that $OPT(\mathcal{I}_{m-1}) = m-1$. Consider an instance $\mathcal{I}_m$. It is easy to see that $OPT(\mathcal{I}_m) \leq m$ (assign, for example, multiplicity 1 to each row). Let us now show that it also holds that $OPT(\mathcal{I}_m) \geq m$. Take an optimal solution to JIS with unit demands on $\mathcal{I}_m$, (z, y, s), and consider the following two cases. First, assume $z_m = 0$. This implies that we have to stab all the m non-overlapping intervals lying on row m with columns or intervals, which would require exactly m elements and therefore, $OPT(\mathcal{I}_m) \geq m$.
Second, assume $z_m = 1$. In this case all the intervals lying on row m are stabbed by the row. Observe that the intervals, that are not yet stabbed, constitute essentially the instance $\mathcal{I}_{m-1}$. From the induction hypothesis we know, that one has to select at least $m-1$ elements to stab all the intervals in $\mathcal{I}_{m-1}$, and therefore $OPT(\mathcal{I}_m) \geq 1 + m - 1 = m$.

So, we have shown that $OPT(\mathcal{I}_m) \leq m$ has to be true together with $OPT(\mathcal{I}_m) \geq m$, which implies $OPT(\mathcal{I}_m) = m$.

Let us estimate the ratio

$$\frac{OPT(\mathcal{I}_m)}{LP(\mathcal{I}_m)} = \frac{m}{m - P(\frac{1}{m} + \frac{1}{m-1} + ... + \frac{1}{P+1})}.$$

Recall the definition of P. Lemma 3.5.2, given in the appendix to this chapter, states that

$$\lim_{m \to \infty} \frac{m}{m - P(m)(\frac{1}{m} + \frac{1}{m-1} + ... + \frac{1}{P(m)+1})} = \frac{e}{e-1}$$

which establishes the result of the theorem. □

3.5 Appendix

Lemma 3.4.2 *Suppose we are given numbers* $1 > \Delta_1 \geq \Delta_2 \geq ... \geq \Delta_m \geq 0$ *and* $\Delta_{m+1} = 0$. *Further, given are positive numbers* $a_1, a_2, ..., a_m$ and Y. *Then we have:*

$$\min_{j=0,...,m} (\sum_{r=1}^{j} a_r + \frac{Y}{1 - \Delta_{j+1}}) \leq \frac{e}{e-1} (\sum_{r=1}^{m} a_r \Delta_r + Y). \tag{3.5.1}$$

Proof. Let us first denote: $C = \sum_{r=1}^{m} a_r \Delta_r + Y$, and introduce a set H :

$$H = \{\delta \in \mathbb{R}^{m+1} \mid 0 \le \delta_r < 1,\ \forall r = 1, ..., m+1,\ \sum_{r=1}^{m+1} a_r \delta_r \le C - Y\}.$$

The proof is structured as follows. First, we argue that the left-hand side of (3.5.1) is bounded from above as follows:

$$\min_{j=0,..,m} (\sum_{r=1}^{j} a_r + \frac{Y}{1-\Delta_{j+1}}) \le \sup_{\delta \in H} \min_{j=0,..,m} (\sum_{r=1}^{j} a_r + \frac{Y}{1-\delta_{j+1}}). \tag{3.5.3}$$

Then we establish the lemma by showing that

$$\sup_{\delta \in H} \min_{j=0,...,m} (\sum_{r=1}^{j} a_r + \frac{Y}{1-\delta_{j+1}}) \le \frac{e}{e-1} C. \tag{3.5.4}$$

It is easy to see that (3.5.3) follows from the observation that the vector $\Delta = (\Delta_1, ..., \Delta_m, \Delta_{m+1})$ belongs to H. Indeed, the supremum value of some function of δ over all $\delta \in H$ is at least as large as the value of this function for any specific $\delta \in H$, in particular for $\delta = \Delta$.

Let us establish (3.5.4). Denote:

$$M(\delta) = \min_{j=0,...,m} (\sum_{r=1}^{j} a_r + \frac{Y}{1-\delta_{j+1}}).$$

We have to show that

$$\sup_{\delta \in H} M(\delta) \le \frac{e}{e-1} C. \tag{3.5.5}$$

Let us for each $q \in \mathbb{R}$ introduce an auxiliary vector $\hat{\delta}(q) \in \mathbb{R}^{m+1}$:

$$\hat{\delta}_j(q) = \begin{cases} 1 - \frac{Y}{q - \sum_{r=1}^{j-1} a_r}, & \text{if } j \text{ is such that } \sum_{r=1}^{j-1} a_r \le q - Y, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5.6}$$

We prove now a sequence of claims.

Claim 1. For each $q \geq Y$, $M(\hat{\delta}(q)) = q$, i.e.,

$$\min_{j=1,\ldots,m+1}\left(\sum_{r=1}^{j-1} a_r + \frac{Y}{1-\hat{\delta}_j(q)}\right) = q.$$

Indeed, for all j such that $q - \sum_{r=1}^{j-1} a_r > Y$, the construction of $\hat{\delta}(q)$ implies that $\sum_{r=1}^{j-1} a_r + \frac{Y}{1-\hat{\delta}_j(q)} = q$, for all the other $j \in \{1,\ldots,m+1\}$, we have $\sum_{r=1}^{j-1} a_r \geq q - Y$ and $\hat{\delta}_j(q) = 0$, which implies that $\sum_{r=1}^{j-1} a_r + \frac{Y}{1-\hat{\delta}_j(q)} = \sum_{r=1}^{j-1} a_r + Y \geq q$. Moreover, since $q \geq Y$, there exists at list one j, namely $j = 1$, such that the expression under the minimum is equal to q. This implies Claim 1.

Claim 2. $\delta \in H$ implies $M(\delta) \geq Y$.
Holds by the definition of $M(\delta)$, provided that $\delta \in H$ implies $0 \leq \delta_j < 1$, $\forall j \in \{1,\ldots,m+1\}$.

Claim 3. $\delta \in H$ implies $M(\delta) = M(\hat{\delta}(M(\delta)))$.
Follows from Claim 1 and 2.

Claim 4. $\delta \in H$ implies $\hat{\delta}(M(\delta)) \in H$.
The condition $0 \leq \hat{\delta}_j(M(\delta)) < 1$ $\forall j \in \{1,\ldots,m+1\}$ is satisfied by the construction of $\hat{\delta}(M(\delta))$ for all $M(\delta)$. The claim will be proven if we show that $\sum_{r=1}^{m} a_r \hat{\delta}_r(M(\delta)) \leq C - Y$. Observe that $\hat{\delta}_j(M(\delta)) \leq \delta_j$, $\forall j \in \{1,\ldots,m+1\}$ implies $\sum_{r=1}^{m} a_r \hat{\delta}_r(M(\delta)) \leq \sum_{r=1}^{m} a_r \delta_r \leq C - Y$, where the last inequality holds since $\delta \in H$. So, let us show that $\hat{\delta}_j(M(\delta)) \leq \delta_j$, $\forall j \in \{1,\ldots,m+1\}$. Take any $j \in \{1,\ldots,m+1\}$, such that $M(\delta) - \sum_{r=1}^{j-1} a_r > Y$. Then the inequality

$$\hat{\delta}_j(M(\delta)) = 1 - \frac{Y}{M(\delta) - \sum_{r=1}^{j-1} a_r} \leq \delta_j$$

is implied by an equivalent inequality

$$M(\delta) \leq \sum_{r=1}^{j-1} a_r + \frac{1}{1-\delta_j} Y,$$

which holds by the definition of $M(\delta)$. For all the remaining $j \in \{1,\ldots,m+1\}$, $\hat{\delta}_j(M(\delta)) \leq \delta_j$ holds automatically, since $\hat{\delta}_j(M(\delta)) = 0$, while $\delta_j \geq 0$.

Claim 5. $\sup_{\delta \in H} M(\delta) \leq \sup_{q\,:\,\hat{\delta}(q) \in H} q$.

From Claim 3,

$$\sup_{\delta \in H} M(\delta) = \sup_{\delta \in H} M(\hat{\delta}(M(\delta))).$$

Claims 2 and 4 imply that:

$$\sup_{\delta \in H} M(\hat{\delta}(M(\delta))) \leq \sup_{\delta\,:\,\hat{\delta}(M(\delta)) \in H, M(\delta) \geq Y} M(\hat{\delta}(M(\delta))),$$

since the domain set of the second supremum is larger. Denote $M(\delta)$ by q, then the latter supremum can be represented and bounded from above as

$$\sup_{\substack{q\,:\,q = M(\delta) \text{ for some } \delta, \\ \hat{\delta}(q)) \in H, q \geq Y}} M(\hat{\delta}(q)) \leq \sup_{q\,:\,\hat{\delta}(q)) \in H, q \geq Y} M(\hat{\delta}(q)).$$

Now Claim 1 implies

$$\sup_{q\,:\,\hat{\delta}(q)) \in H, q \geq Y} M(\hat{\delta}(q)) = \sup_{q\,:\,\hat{\delta}(q)) \in H, q \geq Y} q \leq \sup_{q\,:\,\hat{\delta}(q)) \in H} q.$$

Claim 6. $\sup_{\delta \in H} M(\delta) \leq \frac{e}{e-1} C$.

From Claim 5:

$$\sup_{\delta \in H} M(\delta) \leq \sup_{q\,:\,\hat{\delta}(q)) \in H} q.$$

The claim will be established if we show that the latter supremum is bounded from above by $\frac{e}{e-1} C$, or, equivalently, that $\hat{\delta}(q) \in H$ implies $q \leq \frac{e}{e-1} C$. Recall the definition of $\hat{\delta}(q)$ (3.5.6). Denote:

$$j^* = \max\{j \mid \sum_{r=1}^{j-1} a_r < q - Y\}$$

Then the definition of $\hat{\delta}(q)$ can be rewritten as follows: $\forall j \in \{1, ..., m+1\}$

$$\hat{\delta}(q)_j = \begin{cases} 1 - \frac{Y}{q - \sum_{r=1}^{j-1} a_r}, & \text{if } j \leq j^* \\ 0, & \text{if } j > j^*. \end{cases} \tag{3.5.7}$$

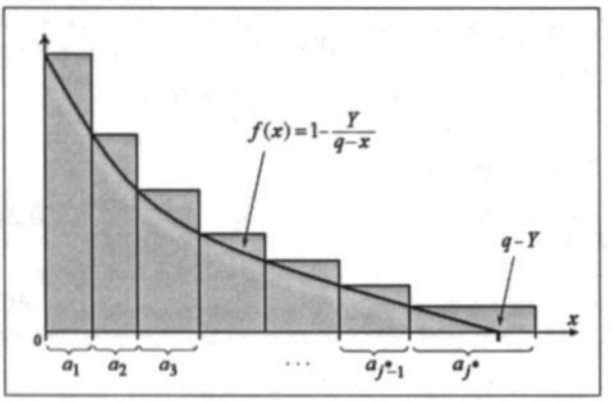

Figure 3.5: The shaded area equals $\sum_{r=1}^{j^*} a_r(1 - \frac{Y}{q-\sum_{p=1}^{r-1} a_p})$.

We know that $\hat{\delta}(q) \in H$ implies:

$$\sum_{r=1}^{m} a_r \hat{\delta}(q)_r \leq C - Y.$$

Using the new definition of $\hat{\delta}(q)$, rewrite the above sum as:

$$\sum_{r=1}^{m} a_r \hat{\delta}(q)_r = \sum_{r=1}^{j^*} a_r \hat{\delta}(q)_r = \sum_{r=1}^{j^*} a_r(1 - \frac{Y}{q - \sum_{p=1}^{r-1} a_p}) \leq C - Y. \quad (3.5.8)$$

Consider

$$\sum_{r=1}^{j^*} a_r(1 - \frac{Y}{q - \sum_{p=1}^{r-1} a_p}).$$

Observe that this sum is equal to the shaded area depicted in Figure 3.5, where the curve is the graph of function $f(x) = 1 - Y/(q - x)$. Clearly this area is larger than the area under the curve, which can be found as an integral of $f(x)$ on the interval $[0, q - Y]$, thus

$$\sum_{r=1}^{j^*} a_r(1 - \frac{Y}{q - \sum_{p=1}^{r-1} a_p}) \geq \int_0^{q-Y} (1 - \frac{Y}{q-x})dx =$$

$$= (x + Y \ln (q - x))|_0^{q-Y} = q - Y + Y \ln Y - Y \ln q.$$

Using this we obtain from (3.5.8):

$$q - Y + Y \ln Y - Y \ln q \leq C - Y,$$

or, equivalently,

$$q + Y \ln Y - Y \ln q \leq C.$$

Dividing both the sides by Y and using $\ln q - \ln Y = \ln q/Y$, we obtain:

$$\frac{q}{Y} - \ln \frac{q}{Y} \leq \frac{C}{Y}.$$

Using Lemma 3.5.1, we have:

$$\frac{q}{Y} \leq \frac{e}{e-1} \frac{C}{Y},$$

or, equivalently,

$$q \leq \frac{e}{e-1} C$$

So, $\frac{e}{e-1}C$ is an upper bound on q given that $\hat{\delta}(q) \in H$. This proves Claim 6 and establishes the lemma.

□

Lemma 3.5.1. *For any $\alpha > 0$ and β, $\alpha - \ln \alpha \leq \beta$ implies $\alpha \leq \frac{e}{e-1}\beta$.*

Proof. Our task is to show that, whenever β satisfies $\beta \geq \alpha - \ln \alpha$, it also satisfies $\beta \geq \frac{e-1}{e}\alpha$. For that it is enough to show that $\alpha - \ln \alpha \geq \frac{e-1}{e}\alpha$ for all $\alpha > 0$, or, equivalently, $\phi(\alpha) = \frac{1}{e}\alpha - \ln \alpha \geq 0, \forall \alpha > 0$.

Consider function $\phi(\alpha)$. It is continuous on $(0, +\infty)$. Using the elementary calculus we know that it achieves its minimum at the point $\alpha = e$. Since the $\phi(e) = 0$, we conclude that it is at least equal to 0 in any point in the interval $(0, +\infty)$, which proves the lemma.

□

Lemma 3.5.2. *Let $P(m) \in \mathbf{N}$ be defined as follows:*

$$\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1} \leq 1 \quad \textit{and} \tag{3.5.9}$$

$$\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1} + \frac{1}{P(m)} \geq 1. \tag{3.5.10}$$

Then,

$$\lim_{m\to\infty} \frac{m}{m - P(m)(\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1})} = \frac{e}{e-1}.$$

Proof. Let us first find $\lim_{m\to\infty} P(m)/m$. Observe that the following inequalities hold:

$$\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1} \geq \int_{P(m)+1}^{m+1} \frac{1}{x}\, dx = \ln \frac{m+1}{P(m)+1},$$

$$\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)} \leq \int_{P(m)-1}^{m} \frac{1}{x}\, dx = \ln \frac{m}{P(m)-1},$$

(the equalities follow from $\int_a^b 1/x\, dx = \ln b/a$.) Then (3.5.9) and (3.5.10) imply

$$1 \geq \ln \frac{m+1}{P(m)+1}, \qquad 1 \leq \ln \frac{m}{P(m)-1}.$$

From this we have:

$$\frac{m+1}{e} - 1 \leq P(m) \leq \frac{m}{e} + 1.$$

Dividing by m:

$$\frac{1 + 1/m}{e} - 1/m \leq \frac{P(m)}{m} \leq \frac{1}{e} + 1/m.$$

Now we see, that $\lim_{m\to\infty} P(m)/m = 1/e$.

Let us now find $\lim_{m\to\infty} (\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1})$. From (3.5.9) and (3.5.10) we have:

$$1 - \frac{1}{P(m)} \leq \frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1} \leq 1$$

Since we already know that $\lim_{m\to\infty} P(m) = \infty$, we have:

$$\lim_{m\to\infty} (\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1}) = 1.$$

Now consider:

$$\frac{m}{m - P(m)(\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1})} = \frac{1}{1 - \frac{P(m)}{m}(\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1})}.$$

Using $\lim_{m\to\infty} \frac{P(m)}{m} = 1/e$ and $\lim_{m\to\infty} (\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1}) = 1$ we have:

$$\lim_{m\to\infty} \frac{1}{1 - \frac{P(m)}{m}(\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{P(m)+1})} = \frac{1}{1-1/e} = \frac{e}{e-1},$$

which establishes the lemma. □

Chapter 4

Approximation and complexity of JIS_k

4.1 Introduction

In this chapter we consider a special case of the job interval stabbing problem. In this special case the number of intervals sharing a row is restricted to be at most a given integer k. Also, all the weights and demands are unit. We refer to this problem as JIS_k.

This chapter extends the results presented in Kovaleva & Spieksma [38] and [39]. In Section 4.2 we show that algorithm *STAB* described in Section 3.4 provides an approximation ratio of $\frac{1}{1-(1-1/k)^k}$ for JIS_k. In Section 4.3 we prove that JIS_2 is MAX SNP-hard, which implies that no polynomial approximation scheme (PTAS) can exist for JIS_2 unless $\mathcal{P} = \mathcal{NP}$. This result strengthens the result of Hassin and Megiddo [31], who establish NP-hardness of JIS_2. It extends to JIS, JIS with unit demands, JIS with unit capacities, considered in the previous chapters, and JIS_k, $k > 2$, since all these problems contain JIS_2 as a special case.

Preliminaries. Let us give a more formal definition of JIS_k: *given is a grid consisting of t columns and m rows. The columns are numbered consecutively from left to right. Further, a set of intervals I is given: $I = \{1, 2, ..., n\}$. The intervals are placed arbitrarily on the rows of the grid with the restriction that at most k intervals can be placed on one row. An interval i is specified by (l_i, r_i, ρ_i), where l_i, r_i are the numbers of the left- and the right-most columns stabbing interval i and ρ_i is the number of its row. We assume that*

the intervals are ordered according to nondecreasing r_i. *The task is to find a minimum cardinality subset of columns and rows so that each interval is stabbed by at least one selected column or row.*

4.2 Algorithm *STAB* for JIS_k

As usual, we write down a natural ILP formulation of JIS_k (here the solution is described by the decision variables $z \in \mathbf{Z}^m$ and $y \in \mathbf{Z}^t$):

$$\begin{array}{lll} \text{Minimize} & \sum_{c=1}^{t} y_c + \sum_{r=1}^{m} z_r & (4.2.1) \\ \text{subject to} & z_{\rho_i} + \sum_{c \in [l_i, r_i]} y_c \geq 1 \quad \forall i = 1, .., n & (4.2.2) \\ & z_r, y_c \in \mathbf{Z}_+ \qquad \forall r, c. & (4.2.3) \end{array}$$

Replacing the integrality constraints (4.2.3) by nonnegativity constraints $z_r \geq 0, y_c \geq 0,\ \forall r, c$, yields the LP relaxation.

We write: $\mathbf{1}z = \sum_{r=1}^{m} z_r$, $\mathbf{1}y = \sum_{c=1}^{t} y_c$.

Figure 4.1 shows algorithm *STAB* in the version adapted to JIS_k. Recall, that $\bar{z}, z^* \in \mathbf{Z}^m$, $\bar{y}, y^* \in \mathbf{Z}^t$ and $V \in \mathbf{R} \cup +\infty$.

1. solve the LP relaxation of (4.2.1)-(4.2.3), obtain an optimal LP solution y^{lp}, z^{lp};
2. number the rows of the grid so that $z_1^{lp} \geq z_2^{lp} \geq ... \geq z_m^{lp}$;
3. $V \leftarrow \infty$;
4. for $j = 0$ to m
   for $i = 1$ to j $\bar{z}_i \leftarrow 1$,
   for $i = j+1$ to m $\bar{z}_i \leftarrow 0$,
   solve $JIS(\bar{z})$, obtain $\bar{y}$;
   if $value(\bar{z}, \bar{y}) < V$, then $V \leftarrow value(\bar{z}, \bar{y})$, $z^* \leftarrow \bar{z}$, $y^* \leftarrow \bar{y}$.
5. return y^*, z^*

Figure 4.1: Algorithm *STAB*.

Theorem 4.2.1. *For any instance of JIS_k the solution returned by algorithm* STAB *describes a feasible solution with a value of at most $\frac{1}{1-(1-1/k)^k}$ times the optimum.*

Remark 4.2.1. Notice that this result improves the approximation ratio of $e/(e-1) \approx 1.582$ given in Theorem 3.4.1. For instance, for $k = 2$, $\frac{1}{1-(1-1/k)^k} \approx 1.333$. However, $\frac{1}{1-(1-1/k)^k}$ tends to $e/(e-1)$ as k increases.

The proof of the theorem is based on the following lemma, proven in the appendix of this chapter.

Lemma 4.2.2. *Given are real numbers $1 > \Delta_1 \geq \Delta_2 \geq ... \geq \Delta_m \geq 0$, a positive real number Y, and an integer $k \geq 2$. Then it holds:*

$$\min_{j=0,...,m} (j + OPT(Q_j(\Delta, Y, k))) \leq \frac{1}{1-(1-1/k)^k}(Y + \sum_{r=1}^{m} \Delta_i), \qquad (4.2.4)$$

where $OPT(Q_j(\Delta, Y, k))$ is the optimal value of the following optimization problem $Q_j(\Delta, Y, k)$:

$$\begin{array}{lll} \textit{Maximize} & (\frac{1}{1-\Delta_{j+1}}\chi_{j+1} + \frac{1}{1-\Delta_{j+2}}\chi_{j+2} + ... + \frac{1}{1-\Delta_m}\chi_m + \sum_{r=m+1}^{\infty}\chi_r) & \\ \textit{subject to} & \chi_1 + ... + \chi_m + \sum_{r=m+1}^{\infty}\chi_r \leq Y & \\ & 0 \leq \chi_r \leq k(1-\Delta_r), & \forall r = 1, ..., m \\ & 0 \leq \chi_r \leq k, & \forall r = m+1, ..., \infty \end{array} \qquad (4.2.5)$$

Proof.(of the Theorem) Consider any instance $\mathcal{I}$ of JIS_k, and let (z^{lp}, y^{lp}) and (z^*, y^*) be respectively an optimal LP solution and the solution returned by the algorithm for $\mathcal{I}$. Obviously, (z^*, y^*) is feasible to the ILP formulation of JIS_k. To establish the approximation ratio we show that the following inequality holds:

$$\mathbf{1}z^* + \mathbf{1}y^* \leq \frac{1}{1-(1-1/k)^k}(\mathbf{1}z^{lp} + \mathbf{1}y^{lp}). \qquad (4.2.6)$$

Since JIS_k is a minimization problem, $(\mathbf{1}z^{lp} + \mathbf{1}y^{lp})$ is at most the optimal value of JIS_k for $\mathcal{I}$ and hence the theorem follows.

Suppose $1 \geq z_1^{lp} \geq z_2^{lp} \geq ... \geq z_m^{lp} \geq 0$. Further, suppose that the first l z^{lp}-values equal 1 $(l \geq 0)$, i.e., $z_1^{lp} = ... = z_l^{lp} = 1$, $1 > z_{l+1}^{lp} \geq z_2^{lp} \geq ... \geq z_m^{lp} \geq 0$.

Let $(\bar{z}^j, \bar{y}^j)$ be the candidate solution number j constructed by the algorithm for $\mathcal{I}$ $(j \in \{0, ..., m\})$. From the design of the algorithm we know that the value of the returned solution is the minimum among the values of the candidate solutions:

$$\mathbf{1}z^* + \mathbf{1}y^* = \min_{j\in\{0,...,m\}} (\mathbf{1}\bar{z}^j + \mathbf{1}\bar{y}^j) \leq \min_{j\in\{l,...,m\}} (\mathbf{1}\bar{z}^j + \mathbf{1}\bar{y}^j), \tag{4.2.7}$$

In the main part of the proof we establish that

$$\mathbf{1}\bar{z}^j + \mathbf{1}\bar{y}^j \leq j + OPT(Q_j(z^{lp}, \mathbf{1}y^{lp}, k)), \ \forall j \in \{l, ..., m\}, \tag{4.2.8}$$

where $OPT(Q_j(\Delta, Y, k))$ is the optimum value of the optimization problem (4.2.5). Then (4.2.7) implies

$$\mathbf{1}\bar{z}^* + \mathbf{1}\bar{y}^* \leq \min_{j=l,...,m} (j + OPT(Q_j(z^{lp}, \mathbf{1}y^{lp}, k))).$$

Using a new index $p = j - l$ we can rewrite the right-hand side as:

$$\min_{p=0,...,m-l} (l + p + OPT(Q_{l+p}(z^{lp}, \mathbf{1}y^{lp}, k))) =$$

$$= \min_{p=0,...,m-l} (p + OPT(Q_{l+p}(z^{lp}, \mathbf{1}y^{lp}, k))) + l. \tag{4.2.10}$$

Now apply Lemma 4.2.2, assuming $\mathbf{1}y^{lp} = Y$, $z^{lp}_{l+r} = \Delta_r$, $\forall r = 1, .., m-l$, and noticing that $OPT(Q_{l+p}(z^{lp}, \mathbf{1}y^{lp}, k)) = OPT(Q_p(\Delta, Y, k))$ $\forall p = 1, .., m - l$. Then we can bound (4.2.10) from above by

$$\frac{1}{1-(1-1/k)^k}\left(\sum_{r=l+1}^{m} z^{lp}_r + \mathbf{1}y^{lp}\right) + l \leq \frac{1}{1-(1-1/k)^k}\left(\sum_{r=l+1}^{m} z^{lp}_r + \mathbf{1}y^{lp} + l\right).$$

Since $z^{lp}_1 = ... = z^{lp}_l = 1$, the last expression equals to

$$\frac{1}{1-(1-1/k)^k}\left(\sum_{r=l+1}^{m} z^{lp}_r + \mathbf{1}y^{lp} + \sum_{r=1}^{l} z^{lp}_r\right) = \frac{1}{1-(1-1/k)^k}(\mathbf{1}z^{lp} + \mathbf{1}y^{lp}),$$

which establishes inequality (4.2.6) and hence the whole theorem.

To establish (4.2.8) we prove 3 claims. Claim 1 validates an assumption, used later in the arguments. Claim 2 shows that $\mathbf{1}\bar{z}^j = j$, Claim 3 proves that $\mathbf{1}\bar{y}^j \leq OPT(Q_j(z^{lp}, \mathbf{1}y^{lp}, k))$. This implies (4.2.8).

Claim 1. We can assume that the optimal LP solution (z^{lp}, y^{lp}) satisfies the constraints (4.2.2) at equality, i.e. $z^{lp}_{\rho_i} + \sum_{c \in [l_i, r_i]} y^{lp}_c = 1$, $\forall i = 1, ..., n$.
If it is not true, we show that it is possible to modify $\mathcal{I}$ into a new instance $\mathcal{I}'$, with an optimal LP solution (z', y'), satisfying the constraints (4.2.2) at equality, so that if we prove (4.2.6) for $\mathcal{I}'$, this implies that (4.2.6) holds for $\mathcal{I}$.

We construct $\mathcal{I}'$ as follows: let N be a common denominator of the values $z^{lp}_1, .., z^{lp}_m, y^{lp}_1, .., y^{lp}_t$. Then $z^{lp}_r = a_r \frac{1}{N}, y^{lp}_c = b_c \frac{1}{N}$, $a_r, b_c \in \mathbf{Z}, \forall r, c$. Replace each column c in $\mathcal{I}$ by b_c copies of it with numbers $c_1, ..., c_{b_c}$. These are the columns of the new $\mathcal{I}'$. The set of rows is the same as in $\mathcal{I}$. Take now any interval i, such that $z^{lp}_{\rho_i} + \sum_{c \in [l_i, r_i]} y^{lp}_c > 1$, and shorten it so that it becomes stabbed by exactly $(1 - z^{lp}_{\rho_i})N$ new columns. This completes the modification of $\mathcal{I}$ into $\mathcal{I}'$ (see Figure 4.2).

Figure 4.2: Example of an instance $\mathcal{I}$ (left) and a new instance $\mathcal{I}'$ (right).

Observation 1. Any feasible LP solution (z', y') for $\mathcal{I}'$ can be transformed into a feasible LP solution (z, y) for $\mathcal{I}$ of the same value, as: $z = z'$, $y_c = \sum_{j=1}^{b_c} y'_{c_j}$.
Observation 2. The value of an optimal LP solution for $\mathcal{I}'$ is greater or equal to the value of an optimal LP solution for $\mathcal{I}$. (Follows from Observation 1.)
Observation 3. For any $\bar{z} \in \mathbf{Z}^m_+$, any feasible solution $\bar{y}'$ to the reduced problem JIS($\bar{z}$) for $\mathcal{I}'$ can be transformed into a feasible solution $\bar{y}$ of JIS($\bar{z}$) for $\mathcal{I}$, which has the same value, using the same transformation as in Observation 1. Notice that the both solutions are integral.
Observation 4. For any $\bar{z} \in \mathbf{Z}^m_+$, the value of an optimal solution to the reduced problem JIS($\bar{z}$) for $\mathcal{I}'$ is greater or equal to the value of an optimal solution of JIS($\bar{z}$) for $\mathcal{I}$. (Follows from Observation 3.)

Consider now a solution for $\mathcal{I}'$ (z', y'), where $z' = z^{lp}$, $y'_{c_j} = \frac{1}{N}$, $\forall c_1, .., c_{b_c}$, $\forall c = 1, .., t$.
Observation 5. The value of (z', y') is equal to the value of (z^{lp}, y^{lp}).
Observation 6. (z', y') is an optimal LP solution for $\mathcal{I}'$. Indeed, from the construction of $\mathcal{I}'$ follows that this is a feasible LP solution for $\mathcal{I}'$. Since

(z^{lp}, y^{lp}) is an optimal LP solution for $\mathcal{I}$, from observations 2 and 5 obtain that it is optimal.

Observation 7. Algorithm *STAB* applied to $\mathcal{I}'$, given an optimal LP solution (z', y'), returns a solution $(z^{*'}, y^{*'})$ with a value larger or equal to the value of solution (z^*, y^*) returned for $\mathcal{I}$, provided the optimal LP solution (z^{lp}, y^{lp}). This follows from the construction of the algorithms, the construction of solution (z', y') and Observation 4. Notice the value of each candidate solution produced by the algorithm for $\mathcal{I}'$ is greater or equal to the value of each candidate solution produced by the algorithm for $\mathcal{I}$.

Thus, we have that the value of (z', y') is equal to the value of (z^{lp}, y^{lp}) and the value of $(z^{*'}, y^{*'})$ is greater or equal to the value of (z^*, y^*). Therefore, inequality (4.2.6) for the original instance $\mathcal{I}$ is implied by inequality (4.2.6) for $\mathcal{I}'$ as follows:

$$\mathbf{1}z^* + \mathbf{1}y^* \leq \mathbf{1}z^{*'} + \mathbf{1}y^{*'} \leq$$

$$\leq \frac{1}{1-(1-1/k)^k}(\mathbf{1}z' + \mathbf{1}y') = \frac{1}{1-(1-1/k)^k}(\mathbf{1}z^{lp} + \mathbf{1}y^{lp}).$$

Therefore it is enough to establish (4.2.6) for instance $\mathcal{I}'$ and the optimal LP solution (z', y'), which satisfies the constraints (4.2.2) at equality. This proves Claim 1.

Consider the candidate solution $(\bar{z}^j, \bar{y}^j)$, for some $j \in \{l, ..., m\}$. By construction:

- $\bar{z}^j_r = 1, \ \forall r \leq j,$
- $\bar{z}^j_r = 0, \ \forall r \geq j+1,$
- $\bar{y}^j$ is an optimal solution to the reduced problem $\mathrm{JIS}(\bar{z}^j)$.

Claim 2. $\mathbf{1}\bar{z}^j = \sum_{r=1}^{m} \bar{z}_r = j$. (Obvious.)

Claim 3. $\mathbf{1}\bar{y}^j \leq OPT(Q_j(z^{lp}, \mathbf{1}y^{lp}, k))$.

To show this we introduce a solution y^j, and show that it is feasible to the LP relaxation of formulation (3.2.1)-(3.2.3) of the reduced problem $\mathrm{JIS}(\bar{z}^j)$ (Claim 3.1), and that $\mathbf{1}y^j \leq OPT(Q_j(z^{lp}, \mathbf{1}y^{lp}, k))$ (Claim 3.2). Then, by Proposition 3.2.1, $\mathbf{1}\bar{y}^j \leq \mathbf{1}y^j \leq OPT(Q_j(z^{lp}, \mathbf{1}y^{lp}, k))$.

Before we introduce the solution y^j, let us partition the columns of the grid into m subsets $S_1, S_2, ..., S_m$, where $S_r \subset \{1, ..., t\}$, $\forall r = 1, ..., m$, in the following way:

$$S_r = \bigcup_{i:\rho_i = r} [l_i, r_i], \tag{4.2.11}$$

i.e., S_r is the set of columns stabbing intervals in row r. Now let us describe the construction of y^j ($j \in \{1, ..., m\}$):

$$y_c^j = \begin{cases} \frac{1}{(1-z_{j+1}^{lp})} y_c^{lp}, & \text{if } c \in S_{j+1} \\ \frac{1}{(1-z_{j+2}^{lp})} y_c^{lp}, & \text{if } c \in S_{j+2} \backslash S_{j+1} \\ \ldots & \ldots \\ \frac{1}{(1-z_m^{lp})} y_c^{lp}, & \text{if } c \in S_m \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_{m-1}) \\ y_c^{lp} & \text{otherwise} \end{cases}$$

Claim 3.1. y^j is feasible to the LP relaxation of the formulation (3.2.1)-(3.2.3) of JIS($\bar{z}^j$).

To prove that, for any interval i we show that the following inequality holds:

$$\sum_{c \in [l_i, r_i]} y_c^j \geq 1 - \bar{z}_{\rho_i}^j. \tag{4.2.12}$$

Indeed, if $\bar{z}_{\rho_i}^j = 1$, the inequality holds automatically. Otherwise, if $\bar{z}_{\rho_i}^j = 0$, from the construction of $\bar{z}^j$ we know that $\rho_i \geq j+1$. Consider $\sum_{c \in [l_i, r_i]} y_c^j$. Take any $c \in [l_i, r_i]$. Recall that $[l_i, r_i] \subset S_{\rho_i}$, and therefore, $c \in S_{\rho_i}$. Since $\rho_i \geq j+1$, we know that

- either $c \in S_{\rho_i} \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_{\rho_i - 1})$,
- or $c \in (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_{\rho_i - 1})$.

From the construction of y^j it is clear, that in the first case $y_c^j = y_c^{lp}/(1 - z_{\rho_i}^{lp})$, and in the second case, y_c^j equals y_c^{lp} multiplied by the coefficient $1/(1 - z_r^{lp})$ for some $r = j+1, ..., \rho_i - 1$. Due to the ordering of the z^{lp}-values, this coefficient is at least as large as $1/(1 - z_{\rho_i}^{lp})$. Thus, for any $c \in [l_i, r_i]$, we have $y_c^j \geq y_c^{lp}/(1 - z_{\rho_i}^{lp})$. Using this, and remembering that (z^{lp}, y^{lp}) is feasible to the LP relaxation of (4.2.1)-(4.2.3), i.e., satisfies $z_{\rho_i}^{lp} + \sum_{c \in [l_i, r_i]} y_c^{lp} \geq 1$, we have for any $i = 1, ..., n$:

$$\sum_{c \in [l_i, r_i]} y_c^j \;\geq\; \frac{1}{(1 - z_{\rho_i}^{lp})} \sum_{c \in [l_i, r_i]} y_c^{lp} \;\geq\; \frac{1 - z_{\rho_i}^{lp}}{1 - z_{\rho_i}^{lp}} = 1.$$

This proves Claim 3.1.

Claim 3.2. $\mathbf{1} y^j \leq OPT(Q_j(z^{lp}, \mathbf{1} y^{lp}, k))$.

Firstly, for any subset of columns $S \subset \{1, ..., t\}$, denote:

$$Y(S) = \sum_{c \in S} y_c^{lp}.$$

From the construction of y^j:

$$\mathbf{1}y^j = \sum_{c=1}^{t} y_c^j = \frac{1}{1-z_{j+1}^{lp}} Y(S_{j+1}) + \frac{1}{1-z_{j+2}^{lp}} Y(S_{j+2} \backslash S_{j+1}) + \ldots + \frac{1}{1-z_m^{lp}} Y(S_m \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_{m-1})) + Y(\{1,\ldots,t\} \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_m)). \quad (4.2.13)$$

Consider the optimization problem $Q_j(z^{lp}, \mathbf{1}y^{lp}, k)$ (see (4.2.5)):

$$\text{Maximize } \left(\frac{1}{1-z_{j+1}^{lp}} \chi_{j+1} + \frac{1}{1-z_{j+2}^{lp}} \chi_{j+2} + \ldots + \frac{1}{1-z_m^{lp}} \chi_m + \sum_{r=m+1}^{\infty} \chi_r \right)$$

$$\text{subject to } \chi_1 + \ldots + \chi_m + \sum_{r=m+1}^{\infty} \chi_r \le \mathbf{1}y^{lp} \quad (4.2.14)$$

$$0 \le \chi_r \le k(1 - z_r^{lp}), \quad \forall r = 1,\ldots,m \quad (4.2.15)$$

$$0 \le \chi_r \le k, \quad \forall r = m+1,\ldots,\infty \quad (4.2.16)$$

To prove Claim 3.2, for any $j \in \{1,\ldots,m\}$ we expose a feasible solution to $Q_j(z^{lp}, \mathbf{1}y^{lp}, k)$, whose value is equal to $\mathbf{1}y^j$. Consider the following assignment:

$$\chi_{j+1} = Y(S_{j+1}),\ \chi_{j+2} = Y(S_{j+2} \backslash S_{j+1}),\ \ldots,\ \chi_m = Y(S_m \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_{m-1})),$$

$$\sum_{r=m+1}^{\infty} \chi_r = Y(\{1,\ldots,t\} \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_m))$$

(The value of $Y(\{1,\ldots,t\} \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_m)$ is distributed arbitrarily among the variables χ_r, $r \ge m+1$, so that the constraint (4.2.16) is satisfied.) From (4.2.13), the value of this assignment is equal to $\mathbf{1}y^j$. Let us show that this assignment is feasible to $Q_j(z, \mathbf{1}y, k)$. First, constraint (4.2.16) is satisfied by construction. Second, constraint (4.2.14) holds since

$$Y(S_{j+1}) + Y(S_{j+2} \backslash S_{j+1}) + \ldots + Y(S_m \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_{m-1})) + Y(\{1,\ldots,t\} \backslash (S_{j+1} \cup S_{j+2} \cup \ldots \cup S_m)) = Y(\{1,\ldots,t\}) = \sum_{c=1}^{t} y_c^{lp} = \mathbf{1}y^{lp}$$

Third, let us show, that constraint (4.2.15) holds too. Using the definition of sets S_r (4.2.11), our assumption in Claim 1 and the fact that there are at

most k intervals per row in $\mathcal{I}$, we obtain, that, for each $r = j+1, ..., m$:

$$Y(S_r \setminus (S_{j+1} \cup S_{j+2} \cup ... \cup S_{r-1})) \leq Y(S_r) = \sum_{c \in S_r} y_c^{lp} =$$

$$= \sum_{i:\rho_i = r} \sum_{c \in [l_i, r_i]} y_c^{lp} = \sum_{i:\rho_i = r} (1 - z_{\rho_i}^{lp}) \leq k(1 - z_r^{lp}).$$

We can conclude that the exposed assignment is feasible to $Q_j(z^{lp}, \mathbf{1}y^{lp}, k)$, which implies that its value $\mathbf{1}y^j$ is upper bounded by $OPT(Q_j(z^{lp}, \mathbf{1}y^{lp}, k))$. This proves Claim 3.2, subsequently, Claim 3, and hence the whole theorem. □

Recall, that algorithm *STAB* is a polynomial algorithm (see Theorem 3.4.3).

Corollary 4.2.3. *Algorithm* STAB *is a* $\frac{1}{1-(1-1/k)^k}$*- approximation algorithm for* JIS_k.

Remark 4.2.2. Observe that JIS_∞ = JIS with unit weights and $\lim_{k\to\infty} \frac{1}{1-(1-1/k)^k} = \frac{e}{e-1}$. Therefore Theorem 4.2.1 implies Theorem 3.4.1 of Chapter 3 for all instances of JIS with unit weights. We also note that the proof of Theorem 4.2.1 does not allow a straightforward generalization to the weighted version of JIS_k.

Integrality gap factor. Here we give a lower bound on the integrality gap factor of the ILP formulation of JIS_k (4.2.1)-(4.2.3) for each $k \in \mathbb{N}$.

Recall, that the integrality gap factor ρ_k of the ILP formulation of JIS_k is the supremum ratio between the optimum values of the ILP formulation $OPT(\mathcal{I})$ and of its LP relaxation $OPT(\mathcal{I})$ over all instances $\mathcal{I}$, i.e.,

$$\rho_k = \sup_{\mathcal{I} \in JIS_k} \frac{OPT(\mathcal{I})}{LP(\mathcal{I})}.$$

Lemma 4.2.4. *For each* $k \in \mathbb{N}$, *the integrality gap factor* ρ_k *of the ILP formulation (4.2.1)-(4.2.3) of* JIS_k *satisfies:*

$$\frac{k}{k - P(k)(\frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{P(k)+1})} \leq \quad \rho_k \quad \leq \frac{1}{1-(1-1/k)^k}, \qquad (4.2.17)$$

where $P(k)$ *is the number, such that:*

$$\frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{P+1} \leq 1 \quad \textit{and} \quad \frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{P+1} + \frac{1}{P} \geq 1. \qquad (4.2.18)$$

Proof. We claim that for any $k \in \mathbf{N}$

$$\rho_k \geq \frac{k}{k - P(k)(\frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{P(k)+1})},$$

where $P(k)$ is the number satisfying (4.2.18). Indeed, consider the family of instances $\{\mathcal{I}_m\}_{m \in \mathbf{N}}$ described in the proof of Theorem 3.4.5 and observe that for each $k \in \mathbf{N}$, $\mathcal{I}_k$ is in fact an instance of the problem JIS_k, since it has at most k intervals per row and all the numerical parameters of it (weights and capacities) are unit. In that proof is shown that the ratio between the optimum value of JIS_k for $\mathcal{I}_k$ ($OPT(\mathcal{I}_k)$) and the value its optimum LP solution ($LP(\mathcal{I}_k)$) is

$$\frac{OPT(\mathcal{I}_k)}{LP(\mathcal{I}_k)} = \frac{k}{k - P(\frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{P+1})},$$

which proves our claim.

On the other hand, inequality 4.2.6, established in the proof of Theorem 4.2.1, implies that for each k, $\rho_k \leq \frac{1}{1-(1-1/k)^k}$.

□

Observe, that the values of both bounds in (4.2.17) are 4/3 for $k = 2$. Also if $k \to \infty$, both values tend to $e/(e-1)$ (see Lemma 3.5.2). Figure 4.3 shows the lower and upper bounds on ρ_k for different $k \in \{2, ..., 50\}$.

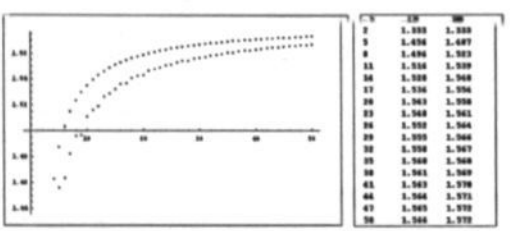

Figure 4.3: An illustration of the lower and upper bounds on the integrality gap factor ρ_k for some $k \in \{2, ..., 50\}$.

We conclude, that the approximation ratio provided by algorithm *STAB* for JIS_k is unlikely to be improved by any other LP-rounding algorithms based on formulation (4.2.1)-(4.2.3) for $k = 2$, and can admit only a slight improvement for $k > 2$ with a value that tends to 0 when $k \to \infty$.

4.3 A Non-Approximability Result

In this section we prove that JIS_2 is hard to approximate arbitrarily closely in polynomial time (unless $\mathcal{P} = \mathcal{NP}$). We assume familiarity with some of the issues in approximation and complexity, see for instance Ausiello et al. [6] or Papadimitriou [41].

Preliminaries. Let us first shortly describe some of the concepts we need.

- An L-reduction. Given two combinatorial optimization problems A and B, where A is a maximization and B is a minimization problem. An L-reduction from A to B is a pair of functions R and S such that:
  (a)R and S are computable in polynomial time,
  (b) for any instance $\mathcal{I}$ of A with optimum value $OPT(\mathcal{I})$, $R(\mathcal{I})$ is an instance of B with optimal value $OPT(R(\mathcal{I}))$, such that

  $$OPT(R(\mathcal{I})) \leq \alpha \cdot OPT(\mathcal{I})$$

  for some positive constant α;
  (c) for any feasible solution s of $R(\mathcal{I})$, $S(s)$ is a feasible solution of $\mathcal{I}$ such that

  $$OPT(\mathcal{I}) - c(S(s)) \leq \beta |OPT(R(\mathcal{I})) - c(s)|$$

  for some positive constant β, where $c(S(s))$ and $c(s)$ denote the values of solution $S(s)$ and s, respectively.
  An L-reduction is an approximation preserving reduction, that is, if problem B can be approximated within $1 - \epsilon$ then problem A can be approximated within $1 - \alpha\beta\epsilon$ (assuming that there is an L-reduction from A to B).

- The class MAX SNP is a class that contains optimization problems that are approximabled in polynomial time within a constant factor.

- The problem *Maximum Bounded 3-Satisfiability*(MAX 3-SAT-3):
  Input: A set of Boolean variables $X = \{x_1, x_2, ..., x_n\}$ and a set $C = \{C_1, C_2, ..., C_r\}$ of clauses over X. Each clause $C_j (j = 1, ..., r)$ consists of at most three literals and each variable $x_i (i = 1, ..., n)$ occurs at most three times in C (either as literal x_i or as literal $\bar{x}_i$).
  Goal: Find a truth assignment for the variables such that the number of satisfied clauses in C is maximum.
  Measure: The number of satisfied clauses in C.

The following result is proven in [41] (see also [6]):

Lemma 4.3.1. *MAX 3-SAT-3 is MAX SNP-hard.*

Arora et al. [5] proved the following result:

Lemma 4.3.2. *If there is a PTAS for some MAX SNP-hard problem, then* $\mathcal{P} = \mathcal{NP}$.

We now have sketched the tools that enable us to prove that JIS_2 has no PTAS (unless $\mathcal{P} = \mathcal{NP}$): this can be done by exhibiting an L-reduction from MAX 3-SAT-3 and using the above Lemmas.

Theorem 4.3.3. *JIS_2 does not have a PTAS unless* $\mathcal{P} = \mathcal{NP}$.

Proof. We prove the theorem by presenting an L-reduction ([41]) from MAX 3-SAT-3 to JIS_2 (see Introduction to this chapter). The result in [5] then establishes the theorem.

Take any instance of MAX 3-SAT-3, let $C = \{C_1, C_2, \ldots, C_r\}$ be a set consisting of r disjunctive clauses, each containing at most 3 literals. Let $x_1, x_2, \ldots, x_n$ denote the variables in the r clauses and, for each $i = 1, \ldots, n$, let $m(i)$ denote the number of occurrences of variable x_i (either as literal x_i or as literal $\bar{x}_i$). Arbitrarily index the occurrences of variable x_i as occurrence $1, 2, \ldots, m(i)$. Notice that without loss of generality we can assume that each variable occurs at least twice in C, thus we have $2 \leq m(i) \leq 3$ for all i and that $\sum_i m(i) \leq 3r$. Moreover, we will also assume (again wlog) that each variable occurs at least once unnegated and at least once negated in C.

We now construct an instance of JIS_2. Recall that JIS has a graph-theoretical interpretation. Then JIS_2, as a special case of JIS, can be formulated in a graph-theoretic context as follows. Construct a graph in which there is a node for each interval and two nodes are connected if they share a column (blue edge), or if they share a row (red edge). Thus, the graph constructed is the edge union of an interval graph and a matching. Notice that a monochromatic maximal clique in such a graph corresponds to a row or a column in JIS_2. In fact, finding a monochromatic clique cover of minimum size is exactly JIS_2.

So, let us construct an instance of JIS_2 as a graph $G = (V, E)$ which is the edge union of an interval graph and a matching. Let I denote an instance of MAX 3-SAT-3 and $R(I)$ the corresponding instance of JIS_2 with corresponding optimal values $OPT(I)$ and $OPT(R(I))$.

For each variable x_i in I, $i = 1, \ldots, n$, we have a subgraph $H1_i = (V1_i, E1_i)$ in $R(I)$, where $V1_i = \{v_{ij} | j = 0, \ldots, 5\}$ and $E1_i = \{\{v_{ij}, v_{i,j+1}\} | j = 0, \ldots, 5\}$ (indices modulo 6). So for each variable x_i in I we have a cycle consisting of 6 nodes in $R(I)$. We refer to the edges $\{v_{i0}, v_{i1}\}, \{v_{i2}, v_{i3}\}$ and $\{v_{i4}, v_{i5}\}$ as T edges, and to the edges $\{v_{i1}, v_{i2}\}, \{v_{i3}, v_{i4}\}$ and $\{v_{i5}, v_{i6}\}$ as F edges. Thus the cycle consists of alternating T and F edges (see Figure 4.4.)

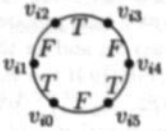

Figure 4.4: The subgraph $H1_i$.

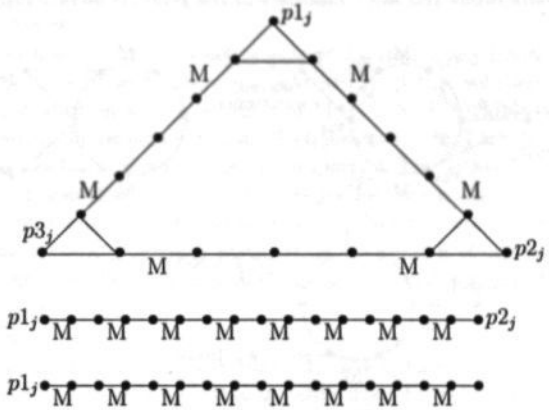

Figure 4.5: The subgraph $H2_j$ when $|C_j| = 3$ (upper figure), when $|C_j| = 2$ (middle figure: P_{17}) and when $|C_j| = 1$ (lower figure: P_{17}).

For each clause C_j in I, $j = 1, \ldots, r$, we have a subgraph $H2_j = (V2_j, E2_j)$ in $R(I)$ depending on the cardinality of C_j, as depicted in Figure 4.5.

When no ambiguity is likely to occur, we refer to the nodes $p1_j$, $p2_j$ and $p3_j$ as p-nodes. Notice that each of the subgraphs in Figure 4.5 has the

property that a clique cover of its vertices has cardinality at least 9, whereas if 1, 2 or even 3 p-nodes need not be covered, one needs at least 8 cliques.

To connect the subgraphs introduced so far in $R(I)$, consider some clause C_j, and consider the first variable occurring in this clause C_j, say x_i. Let this be the q-th occurrence of this variable x_i in C, $q \in \{1, \ldots, m(i)\}$. If the variable x_i occurs as literal x_i add the edges $\{p1_j, v_{i,2q-2}\}$ and $\{p1_j, v_{i,2q-1}\}$ to E. (The node $p1_j$ will then be referred to as a true p-node). If the variable x_i occurs as literal $\bar{x}_i$ add the edges $\{p1_j, v_{i,2q-1}\}$ and $\{p1_j, v_{i,2q}\}$ to E. (The node $p1_j$ will then be referred to as a false p-node). Consider now the second (third) variable occurring in C_j, say x_l, and let this be the q-th occurrence of this variable x_l in C, $q \in \{1, \ldots, m(l)\}$. If the variable x_l occurs as literal x_l add the edges $\{p2_j, v_{l,2q-2}\}$ and $\{p2_j, v_{l,2q-1}\}$ ($\{p3_j, v_{l,2q-2}\}$ and $\{p3_j, v_{l,2q-1}\}$) to E. If the variable x_l occurs as literal $\bar{x}_l$ add the edges $\{p2_j, v_{l,2q-1}\}$ and $\{p2_j, v_{l,2q}\}$ ($\{p3_j, v_{l,2q-1}\}$ and $\{p3_j, v_{l,2q}\}$) to E. This is done for all clauses C_j, $j = 1, \ldots, r$. See Figure 4.6 for a graphic representation of the way in which the subgraphs $H1_i$ are connected to the p-nodes of subgraphs $H2_j$.

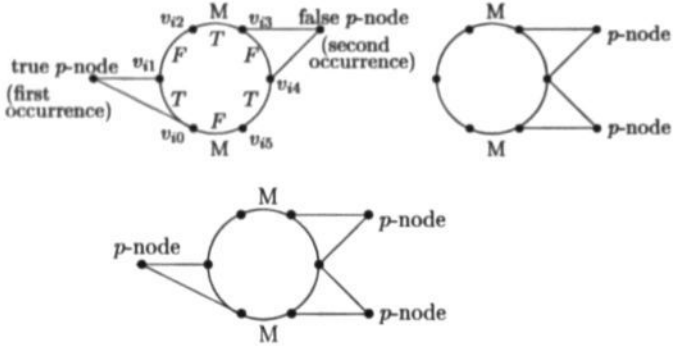

Figure 4.6: The subgraph $H1_i$ and its connections when $m(i) = 2$ (upper figure, 2 possibilities) and when $m(i) = 3$ (lower figure).

Now the graph $G = (V, E)$ is completely specified.

We now exhibit a matching M in G. M consists of two parts: edges in $\cup_i H1_i$ and edges in $\cup_j H2_j$. For the first part we take the edges that are marked with an 'M' in Figure 4.6, for the second part we take the edges that are marked with an 'M' in Figure 4.5.

Obviously, M is indeed a matching. Also, one can verify that the remaining edges in G form an interval graph (notice that each $H2_j$ disconnects).

In the remaining part of the proof, with the word "clique cover" a monochromatic clique cover is meant, that is a clique cover with the edges of each clique having the same color.

In order to show that this reduction is an L-reduction, consider the following. Observe that $v \equiv OPT(I) \geq \frac{1}{2}r$. (Indeed, by considering the assignment: all variables true, and: all variables false, it follows that each clause is true in at least one of both assignments). We have:

$$OPT(R(I)) \leq 3n + 9r = \frac{9}{2}r + 9r \leq 27v = 27 \cdot OPT(I).$$

The first inequality follows from the fact that 3 cliques can be selected from each $H1_i$, $i = 1, \ldots, n$ to cover its nodes, and 9 cliques can be selected from each subgraph $H2_j$, $j = 1, \ldots, r$ to cover its nodes. Since $n \leq \frac{3}{2}r$, the inequality follows.

Consider now an arbitrary solution to $R(I)$, that is any (monochromatic) clique cover s in G with size $c(s)$. We will map this solution s using an intermediate solution s' to a solution of MAX 3-SAT-3, called $S(s)$. To do this we need the following definition. A clique cover s in G is called *consistent* iff for each $i = 1, \ldots, n$, the following property holds: either the nodes from $V1_i$ are in cliques that contain only T edges and are maximal, or the nodes from $V1_i$ are in cliques that contain only F edges and are maximal.

Now we state a procedure which takes as input a clique cover s. The output of the procedure is a consistent clique cover called s' with the property that $c(s') \leq c(s)$.

Procedure

Consider s. For $i = 1, \ldots, n$, consider $V1_i$. If $V1_i$ fails the property because a clique in s is not maximal, this is easily fixed by enlarging one or more cliques by adding the appropriate p-node (and perhaps adjusting the clique cover in an obvious manner). If it fails the property because it has a T edge as well as an F edge from $H1_i$ in a clique we do the following. Notice that in this case at least 4 cliques are used for the nodes in $H1_i$. Consider the p-nodes that are contained in cliques used to cover nodes of $H1_i$. Now, if among those p-nodes there are at least 2 false p-nodes take the 3 maximal F-cliques, else take the 3 maximal T-cliques. And, if necessary, take as a single clique the p-node not covered by these maximal T or F cliques (there

can be at most 1, so $c(s') \leq c(s)$).

End of Procedure

After applying this procedure to any clique cover s in G, a consistent solution s' is delivered with $c(s') \leq c(s)$. Since s' is consistent, it is now straightforward to identify the corresponding solution $S(s)$ in MAX 3-SAT-3: simply set variable x_i, $i = 1, \ldots, n$ true if all T-edges in subgraph $H1_i$ are in the clique cover s', else set x_i false. How many clauses in I are satisfied by this truth assignment? Observe that the construction of G implies that if for some consistent clique cover s a p-node from some $H2_j$ is contained in a clique that covers also nodes from some $H1_i$ we need 8 cliques to cover $H2_j$. Let there be l subgraph $H2_j$ for which at least one p-node is gone in this way. As easy to verify, this happens if and only if l clauses in I are satisfied by this truth assignment.

Again, let $v = OPT(I)$, let s be some feasible clique cover, and let $c(S(s)) = l$. The following (in)equalities are true:

- $c(s) \geq c(s')$ (by construction),
- $c(s') = 3n + 8l + 9(r - l) = 3n + 9r - l$ (by construction), and
- $OPT(R(I)) \leq 3n + 8v + 9(r - v) = 3n + 9r - v$ (consider the truth assignment that is optimum for I, that is, v clauses are satisfied by this assignment. We can exhibit in $R(I)$) a corresponding clique cover by selecting 'T' or 'F' edges in subgraphs $H1_i$ in accordance to the truth assignment, and extending them to the maximal cliques by including p-nodes if possible. This requires $3n$ cliques. Observe that after that exactly v subgraphs $H2_j$ have at least one of the p-nodes covered. We need 8 cliques to cover these subgraphs $H2_j$ and 9 cliques to cover the others. This gives a clique cover of size $3n+8v+9(r-v) = 3n+9r-v$).

Thus, for any clique cover s with $c(S(s)) = l$:

$$c(s) - OPT(R(I)) \geq 3n + 9r - l - (3n + 9r - v) = v - l = OPT(I) - c(S(s)),$$

which finishes the proof. □

Remark 4.3.1. Notice that the theorem remains true when the number of intervals that share a column is bounded by 3.

4.4 Appendix

Lemma 4.2.2 *Given are real numbers $1 > \Delta_1 \geq \Delta_2 \geq \ldots \geq \Delta_m \geq 0$, a positive real number Y and an integer $k \geq 2$. Then it holds:*

$$\min_{j=0,\ldots,m} (j + OPT(Q_j(\Delta, Y, k))) \leq \frac{1}{1-(1-1/k)^k}(Y + \sum_{r=1}^{m} \Delta_i), \qquad (4.4.1)$$

where $OPT(Q_j(\Delta, Y, k))$ is the optimal value of the following optimization problem $Q_j(\Delta, Y, k)$:

$$\begin{aligned}
&\text{Maximize } (\tfrac{1}{1-\Delta_{j+1}}\chi_{j+1} + \tfrac{1}{1-\Delta_{j+2}}\chi_{j+2} + \ldots + \tfrac{1}{1-\Delta_m}\chi_m + \textstyle\sum_{r=m+1}^{\infty}\chi_r) \\
&\text{subject to} \quad \chi_1 + \ldots + \chi_m + \textstyle\sum_{r=m+1}^{\infty}\chi_r \leq Y \qquad (4.4.2)\\
&\qquad\qquad 0 \leq \chi_r \leq k(1-\Delta_r), \qquad \forall r = 1,\ldots,m \qquad (4.4.3)\\
&\qquad\qquad 0 \leq \chi_r \leq k, \qquad\quad \forall r = m+1,\ldots,\infty \qquad (4.4.4)
\end{aligned}$$

Proof. The proof consists of two claims. In Claim 1 we show that the left-hand side of (4.4.1) is upper bounded by the following supremum:

$$\sup_{f(\cdot) \in H} \; G(f(\cdot)) \qquad (4.4.5)$$

where

$$G(f(\cdot)) = \min_{x \in \mathbb{R}_+} \; (f(x) + k(f(x+Y) - f(x))), \qquad (4.4.6)$$

and the class of functions H is defined as follows:

$$H = \left\{ f(\cdot) : \mathbb{R}_+ \to \mathbb{R}_+ \;\middle|\; \begin{array}{l} f(\cdot) \text{ is continuous, increasing, concave,} \\ f(0) = 0, f(x) \leq x/k + \sum_{r=1}^{m} \Delta_r \end{array} \right\}. \qquad (4.4.7)$$

In Claim 2 we show, that this supremum is upper bounded by the right-hand side of (4.4.1), which proves the lemma.

Claim 1.

$$\min_{j=0,\ldots,m} (j + OPT(Q_j(\Delta, Y, k))) \leq \sup_{f(\cdot) \in H} \; G(f(\cdot)),$$

where H and $G(f(\cdot))$ are defined in (4.4.6) and (4.4.7).
To establish this, it is sufficient to exhibit a particular function $\hat{f}(\cdot) \in H$, such that:

$$G(\hat{f}(\cdot)) = \min_{j=0,\ldots,m} (j + OPT(Q_j(\Delta, Y, k))). \tag{4.4.8}$$

Then supremum of $G(f(\cdot))$ over all the possible $f(\cdot) \in H$ is clearly larger or equal to $G(\hat{f}(\cdot))$.

Before we describe the function $\hat{f}(\cdot)$, let us define an auxiliary function $F(\cdot)$: $\mathbb{R}_+ \to \mathbb{R}_+$ as follows:

$$F(q) \equiv \sum_{r=1}^{\lfloor q \rfloor} k(1-\Delta_r) + (q - \lfloor q \rfloor)\ k\ (1 - \Delta_{\lfloor q \rfloor + 1}), \tag{4.4.9}$$

where we set $\Delta_r = 0$, $\forall r \geq m+1$ (see Figure 4.7).

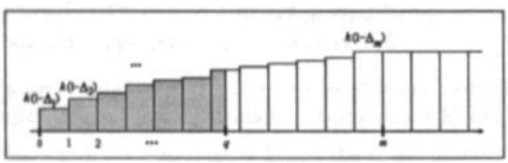

Figure 4.7: Imagine a sequence of rectangles, each of unit length, put on the abscissa axis as shown above. The height of rectangle i is $k(1-\Delta_i)$. Then the value of $F(q)$ is equal to the shaded area.

Observe that $F(\cdot)$ is
- continuous;
- increasing, since $\Delta_r < 1$, and therefore $(1-\Delta_r) > 0$, $\forall r = 1, \ldots, \infty$,
- convex, since the coefficients Δ_r are non-increasing with increasing r, and therefore the coefficients $(1-\Delta_r)$ non-decrease with increasing r.
- $F(0) = 0$;
- $F(q) \geq k(q - \sum_{r=1}^{m} \Delta_r)$, $\forall q \in \mathbb{R}_+$, since $F(q)$ can be also represented as:

$$F(q) = k(q - (\sum_{r=1}^{\lfloor q \rfloor} \Delta_r + (q - \lfloor q \rfloor)\Delta_{\lfloor q \rfloor + 1})),$$

and then it is easy to see that the inequality follows from the fact, that $(\sum_{r=1}^{\lfloor q \rfloor} \Delta_r + (q - \lfloor q \rfloor)\Delta_{\lfloor x \rfloor + 1})) \leq \sum_{r=1}^{m} \Delta_r$, $\forall q \in \mathbb{R}_+$.
- $F(q)$ is linear on each of the intervals $[j, j+1]$, $j = 0, \ldots, m-1$, and on $[m, +\infty)$ (see Figure 4.8).

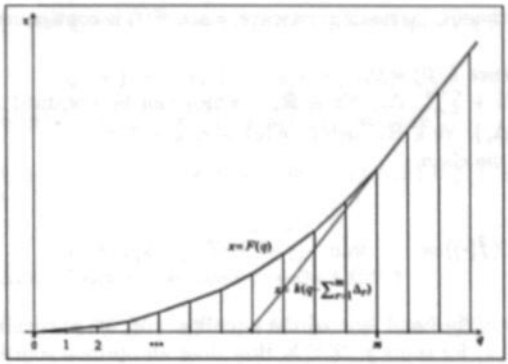

Figure 4.8: Function $F(q)$.

We are now ready to present $\hat{f}(\cdot) : \mathbf{R}_+ \to \mathbf{R}_+$. We define:

$$\hat{f}(\cdot) \equiv F^{-1}(\cdot)$$

(since $F(\cdot)$ is increasing, $F^{-1}(\cdot)$ exists.)

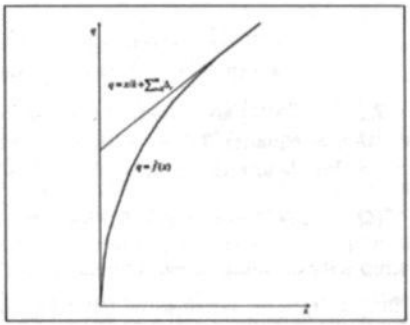

Figure 4.9: Function $\hat{f}(x)$.

Claim 1.1. $\hat{f}(\cdot) \in H$.
Indeed, $\hat{f}(\cdot)$ has the following properties:
- $\hat{f}(\cdot) : \mathbf{R}_+ \to \mathbf{R}_+$ since $F(\cdot) : \mathbf{R}_+ \to \mathbf{R}_+$;

- $\hat{f}(\cdot)$ is continuous, increasing, concave, since $F(\cdot)$ is continuous, increasing, convex;
- $\hat{f}(0) = 0$, since $F(0) = 0$;
- $\hat{f}(x) \leq x/k + \sum_{r=1}^{m} \Delta_r$, $\forall x \in \mathbb{R}_+$, which can be obtained from $F(q) \geq k(q - \sum_{r=1}^{m} \Delta_r)$, $\forall q \in \mathbb{R}_+$, using: $F(q) = x$, $q = \hat{f}(x)$.

This proves the claim.

Claim 1.2.

$$G(\hat{f}(\cdot)) = \min_{j=0,\ldots,m} (j + OPT(Q_j(\Delta, Y, k))). \tag{4.4.10}$$

Consider the right-hand side of the equality, and, in particular, the problem $Q_j(\Delta, Y, k)$ for some j, Δ, Y, k. How does an optimal solution look like? Notice, that this problem is essentially the LP relaxation of the well known Knapsack problem and the coefficients $\frac{1}{1-\Delta_{j+1}}, \frac{1}{1-\Delta_{j+2}}, \ldots, \frac{1}{1-\Delta_r}, \ldots$ in the objective function are non-increasing. Then, obviously, when distributing the value of Y among $\chi_{j+1}, \chi_{j+2}, \ldots$, one should assign the largest possible values to the earliest in this order variables. Therefore the following solution is optimal to $Q_j(\Delta, Y, k)$:

$$\begin{aligned} \chi_{j+1} &= k(1 - \Delta_{j+1}) \\ &\ldots \\ \chi_{j+p} &= k(1 - \Delta_{j+p}) \\ \chi_{j+p+1} &= \delta k(1 - \Delta_{j+1}), \end{aligned}$$

where p and δ ($p \in \mathbb{Z}_+$, $0 < \delta < 1$) are chosen so that the constraint (4.4.2) is satisfied by this solution as equality. Observe, that the value of this solution is kq, where $q = p + \delta$. We denote this q by $q_j(\Delta, Y, k)$. Thus

$$OPT(Q_j(\Delta, Y, k)) = kq_j(\Delta, Y, k), \ \forall j = 0, \ldots, m.$$

In the sequel for simplicity of notation we refer to $q_j(\Delta, Y, k)$ as q_j.

Observe, that q_j, for $j = 0, \ldots, m$, satisfies the following equality:

$$q_j : \ F(j + q_j) - F(j) = Y \tag{4.4.11}$$

Indeed, this follows from the definition of $F(\cdot)$ (4.4.9) and the condition that (4.4.2) is satisfied as equality.
From here, $q_j = F^{-1}(Y + F(j)) - j$. Denote: $x_j = F(j)$, $\forall j = 0, \ldots, m$. Then,

$j = F^{-1}(x_j)$ and $q_j = F^{-1}(Y + x_j)) - F^{-1}(x_j)$. Replacing $F^{-1}(\cdot)$ by $\hat{f}(\cdot)$, obtain:

$$q_j = \hat{f}(Y + x_j)) - \hat{f}(x_j), \ \forall j = 0, ..., m.$$

Using this together with $j = F^{-1}(x_j) = \hat{f}(x_j)$, we have:

$$\min_{j=0,...,m} (j + OPT(Q_j(\Delta, Y, k))) = \min_{\substack{j=0,...,m \\ x_j = \hat{f}^{-1}(j)}} (\hat{f}(x_j) + k(\hat{f}(Y + x_j) - \hat{f}(x_j)))$$

Now to establish Claim 1.2 we need to show that

$$\min_{\substack{j=0,..,m \\ x_j = \hat{f}^{-1}(j)}} (\hat{f}(x_j) + k(\hat{f}(Y + x_j) - \hat{f}(x_j))) = \min_{x \in \mathbb{R}_+} \left(\hat{f}(x) + k(\hat{f}(x + Y) - \hat{f}(x))\right) \tag{4.4.13}$$

We do this by showing, that the function $\hat{f}(x) + k(\hat{f}(x + Y) - \hat{f}(x))$ is continuous and concave on each of the intervals $[x_j, x_{j+1}]$, $\forall j = 0, ..., m - 1$, and is increasing on $[x_m, +\infty)$. Therefore the minimum can be achieved only at the endpoints $x_0, x_1, ..., x_m$.

Indeed, consider function $\hat{f}(x) + k(\hat{f}(x + Y) - \hat{f}(x))$ on the interval $[x_j, x_{j+1}]$ for some $j = 0, ..., m - 1$. This function can be also presented as: $k\hat{f}(x + Y) - (k - 1)\hat{f}(x)$. Since $k \geq 2$, the first term here is positive and the second is negative. Since $\hat{f}(\cdot)$ is concave everywhere on $\mathbb{R}_+$, we know that $\hat{f}(x + Y)$ is concave on $[x_j, x_{j+1}]$. Now notice, that $\hat{f}(x)$ is linear on each of $[x_j, x_{j+1}], j = 0, ..., m - 1$. This follows from the fact that $F(q)$ is linear on $[j, j + 1]$, $j = 0, ..., m - 1$. Obviously, a concave function minus a linear function is again concave.

Now we show that $k\hat{f}(x + Y) - (k - 1)\hat{f}(x)$ is increasing in $[x_m, +\infty)$. Recall, that $F(q)$ is increasing and linear in $[m, +\infty]$. Therefore $\hat{f}(x) = F^{-1}(\cdot)$ is increasing and linear in $[x_m, +\infty)$. This implies that the derivatives of $\hat{f}(x + Y)$ and $\hat{f}(x)$ are equal to the same positive number for all $x \geq x_m$. Then the derivative of $k\hat{f}(x + Y) - (k - 1)\hat{f}(x)$ is positive in $[x_m, +\infty)$.

We have proved (4.4.13) and this completes the proof of Claim 1.2 and consequently, the proof of Claim 1.

Claim 2.

$$\sup_{f(\cdot) \in H} G(f(\cdot)) \leq \frac{1}{1 - (1 - 1/k)^k} C,$$

where

$$C = Y + \sum_{r=1}^{m} \Delta_r,$$

$$G(f(\cdot)) = \min_{x \in \mathbb{R}_+} (f(x) + k(f(x+Y) - f(x)))$$

and the set of functions H using notation C can be described as

$$H = \left\{ f(\cdot) : \mathbb{R}_+ \to \mathbb{R}_+ \;\middle|\; \begin{array}{l} f(\cdot) \text{ is continuous, increasing, concave,} \\ f(0) = 0, f(x) \leq x/k + C - Y \end{array} \right\}.$$

Claim 2.1.

$$\sup_{f(\cdot) \in H} G(f(\cdot)) = \sup_{g \,:\, f^g(\cdot) \in H} g,$$

where for each $g \in \mathbb{R}_+$ function $f^g(\cdot)$ is defined as follows:
- $f^g(pY) = g(1 - (1 - 1/k)^p)$, $\forall p \in 0 \cup \mathbb{N}$,
- $f^g(x)$ is continuous in $[0, +\infty)$ and linear in each $[(p-1)Y, pY]$, $p \in \mathbb{N}$.

Let us prove this claim. First of all, notice that $f^g(\cdot)$ is completely defined by the above characterization. An example of this function is shown in Figure 4.10.

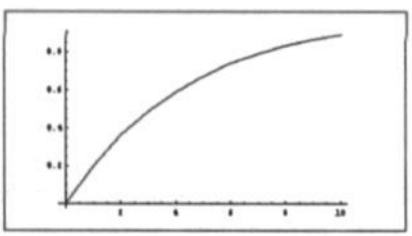

Figure 4.10: Illustration of function $f^g(x)$ for $k = 5$, $g = 1$.

To prove Claim 2.1 it is enough to show that, for any $f(\cdot) \in H$, there exists a $f^{\hat{g}}(\cdot) \in H$, with $\hat{g} \geq 0$, such that

$$G(f(\cdot)) = G(f^{\hat{g}}(\cdot)) = \hat{g}.$$

To show that, we prove 2 subsidiary claims.

Claim 2.1.1. For any $g \geq 0$:

$$G(f^g(\cdot)) \equiv \min_{x \in \mathbb{R}_+} (f^g(x) + k(f^g(x+Y) - f^g(x))) = g.$$

Let us prove this. By definition, $f^g(x)$ is linear in each of the intervals $[(p-1)Y, pY]$, $p \in \mathbf{N}$. This implies that function $(f^g(x) + k(f^g(x+Y) - f^g(x)))$ is linear in each of these intervals as well. Therefore the minimum over all $x \geq 0$ is achieved in one of the endpoints $0, Y, 2Y, ..., pY, ...$ Consider $(f^g(x) + k(f^g(x+Y) - f^g(x)))$ in the point $x = pY$, for some $p \in \{0, 1, 2, ...\}$:

$$(f^g(pY) + k(f^g((p+1)Y) - f^g(pY))) =$$

$$= \big(g(1-(1-1/k)^p) + k(g(1-(1-1/k)^{p+1}) - g(1-(1-1/k)^p))\big) =$$

$$= g\,(1-(1-1/k)^p - k(1-1/k)(1-1/k)^p + k(1-1/k)^p) =$$

$$= g\,(1-(1-1/k)^p - (k-1)(1-1/k)^p + k(1-1/k)^p) =$$

$$= g\,(1-(1-1/k)^p - k(1-1/k)^p + (1-1/k)^p + k(1-1/k)^p) = g.$$

This proves Claim 2.1.1.

Claim 2.1.2. For any function $f(\cdot) \in H$ and $\hat{g}$, such that $\hat{g} = G(f(\cdot))$, the following is true: $f^{\hat{g}}(\cdot) \in H$.

Observe, that this is implied by $f^{\hat{g}}(x) \leq x/k + C - Y, \forall x \in \mathbf{R}_+$, which in turn is implied by $f^{\hat{g}}(x) \leq f(x)$, $\forall x \in \mathbf{R}_+$, since from $f(\cdot) \in H$ follows that $f(x) \leq x/k + C - Y, \forall x \in \mathbf{R}_+$.

So, let us establish that $f^{\hat{g}}(x) \leq f(x)$, $\forall x \in \mathbf{R}_+$. Recall, that $f^{\hat{g}}(x)$ is linear in each of the intervals $[(p-1)Y, pY]$, $p \in \mathbf{N}$, and notice that $f(x)$ is concave in $\mathbf{R}_+$. Then it is sufficient to show that:

$$f^{\hat{g}}(x) \leq f(x), \ \forall x = pY, \ p \in 0 \cup \mathbf{N}.$$

We use mathematical induction on p. For $p = 0$, $f^{\hat{g}}(0) = f(0) = 0$ and the inequality trivially holds.

Suppose, for $p-1$ we have proven: $f^{\hat{g}}((p-1)Y) \leq f((p-1)Y)$, and let us show, that $f^{\hat{g}}(pY) \leq f(pY)$.

Observe, that $f^{\hat{g}(\cdot)}$ can be represented in a recursive way as follows:

$$f^{\hat{g}}(pY) = \hat{g}/k + f^{\hat{g}}((p-1)Y)\ (1-1/k). \tag{4.4.15}$$

Then from the definition of $\hat{g}$ we have that

$$\hat{g} \leq f((p-1)Y) + k(f(pY) - f((p-1)Y)).$$

This implies:

$$f(pY) \geq \hat{g}/k + f((p-1)Y)\ (1-1/k).$$

By the induction hypothesis and (4.4.15) we can bound this from below as:

$$f(pY) \geq \hat{g}/k + f((p-1)Y)\,(1-1/k) \;\geq\; \hat{g}/k + f^{\hat{g}}((p-1)Y)\,(1-1/k) = f^{\hat{g}}(pY).$$

Thus we have proven that $f^{\hat{g}}(x) \leq f(x)$, $\forall x = pY$, $p \in 0 \cup \mathbf{N}$, which implies $f^{\hat{g}}(x) \leq f(x) \leq x/k + C - Y$, $\forall x \in \mathbf{R}_+$, and therefore $f^{\hat{g}}(\cdot) \in H$, which proves Claim 2.1.2.

These 2 claims imply that for any $f(\cdot) \in H$, there exists $f^{\hat{g}}(\cdot) \in H$, with $\hat{g} \geq 0$, such that

$$G(f(\cdot)) = G(f^{\hat{g}}(\cdot)) = \hat{g}.$$

This implies Claim 2.1.

Claim 2.2.

$$\sup_{g\,:\,f^g(\cdot)\in H} g \leq \frac{1}{1-(1-1/k)^k}\,C.$$

Indeed, $f^g(\cdot) \in H$ implies $f^g(x) \leq x/k + C - Y$, for all $x \in \mathbf{R}_+$, and in particular, for $x = kY$. From this, using the definition of $f^g(\cdot)$, obtain:

$$f^g(kY) \equiv g(1-(1-1/k)^k) \leq \frac{kY}{k} + C - Y,$$

therefore

$$g \leq \frac{1}{(1-(1-1/k)^k)}\,C,$$

which proves Claim 2.3, subsequently, Claim 2 , and establishes the lemma. □

Remark 4.4.1. This Lemma implies Lemma 3.4.2 from Chapter 3 in case when all the values $a_1, a_2, ..., a_m$ from the conditions of Lemma 3.4.2 are equal 1. To see that assume $k \to \infty$.

Chapter 5

An improved PTAS for the rectangle packing problem with a bounded height ratio: a new dynamic programming procedure.

5.1 Introduction

In this chapter we consider a special case of the following problem: given is a set of weighted, axis-parallel, closed rectangles in the plane, find a maximum weight subset of nonoverlapping rectangles.

We refer to this problem as the *rectangle packing problem*. From a graph-theoretical point of view, this problem is the maximum weight independent set problem in the intersection graph of axis-parallel closed rectangles.

The special case we are interested in is the case, when the ratio between the largest height of a rectangle and the smallest is bounded by B. We refer to it as the *rectangle packing problem with a bounded height ratio* RP^B.

Complexity and previous research. The rectangle packing problem is a notoriously difficult problem. Even the unweighted case restricted to squares of unit size is known to be NP-hard (see Fowler et al. [23]). To our knowledge it is still an open problem whether the rectangle packing problem (even its unweighted version) can be approximated within a constant factor in

polynomial time. So far the best known approximation algorithm by Berman et al. [15] achieves a factor of $O(\log_b n)$ for any constant b if running in time $n^{O(b)}$. However, in many cases a particular application suggests restrictions simplifying the model. For instance, Hochbaum & Maass [35], in connection to applications in VLSI design, considered the unweighted packing problem restricted to unit squares or disks (hypercubes or balls in higher dimensions). They developed a so-called shifting technique that enabled them to construct a polynomial time approximation scheme (PTAS), which for each $k \geq 2$ provides an approximation factor $(1 - 1/k)$ and runs in time $n^{O(k^2)}$ (n is the number of objects in the input.)

The case when the objects are fat (i.e., the ratio between the sizes along x and y-coordinates is bounded from above and below) but can have possibly varying sizes (for example, squares or discs on the plane, not necessary of unit size) was considered in Erlebach et al. [20] and Chan [16]. These authors independently developed two PTAS, using strategies that are similar in spirit. The running time $n^{O(k^4)}$ achieved for R^2 in [20] was improved to $n^{O(k)}$ by [16].

Agarwal et al. [2] were interested in packing rectangles of unit height (an unweighted version), a problem suggested by an application in the cartography. Combining the shifting technique with dynamic programming they generalized the PTAS by Hochbaum & Maass to this problem and reduced its running time to $n^{O(k)}$.

Our contribution. This chapter is based on results presented in [37].
The already mentioned paper by Agarwal et al. [2] describes a PTAS for the unweighted case of the rectangle packing problem restricted to rectangles of unit height. Observe that this is an unweighted case of RP^1, a special case of RP^B. This PTAS uses the shifting technique by Hochbaum & Maass [35] in combination with a dynamic programming subroutine and requires $O(n^{2k-1})$ time and memory space achieving an approximation factor of $(1-1/k)$. This result can be easily generalized to RP^B. To a large extent the running time and memory consumption of this PTAS are determined by the dynamic programming procedure it uses. Using a different, more sophisticated dynamic programming routine we reduce the time and memory requirements of the PTAS of [2]. For $k > B$ the improved PTAS provides approximation factor of $(1 - B/k)$, while running in time $O(kn^k)$ and using $O(n^k)$ of memory.

Outline of the chapter. In section 5.2 we give the formal problem statement and describe the framework of the PTAS of [2] as applied to RP^B. In section 5.3 we describe our dynamic programming procedure.

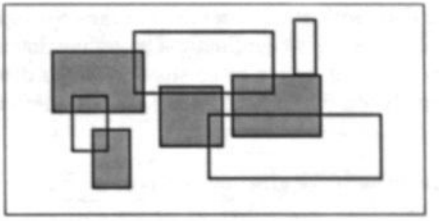

Figure 5.1: A problem instance and a feasible solution (shaded).

5.2 The Polynomial-Time Approximation Scheme

We consider the following problem $\mathcal{P}$: *given is a set $\mathcal{R}$ of n axis-parallel, closed rectangles $\mathcal{R}= \{1, 2, ..., n\}$, rectangle i having a height h_i, such that $(\max_{i \in \mathcal{R}} h_i)/(\min_{i \in \mathcal{R}} h_i) \leq B$. Each rectangle i is specified in terms of its Cartesian coordinates $[l_i, r_i] \times [b_i, b_i + h_i]$ (so the point (l_i, b_i) is the lower left vertex of rectangle i.) The rectangles are ordered according to non-decreasing r_i. For each rectangle i a weight $w(i)$ is specified, $w(i) \in \mathbb{R}^+, \forall i \in 1, ..., n$. The goal is to find a maximum weight subset of pairwise non-overlapping rectangles.*

An example of a problem instance and a feasible solution to it is shown in Figure 5.1. Notice that the rectangles are closed and therefore two rectangles sharing an edge are considered to be overlapping.

Without loss of generality assume that B is integer, otherwise we can always take $\lceil B \rceil$ instead of B.

Denote $h_{max} = \max_{i \in \mathcal{R}} h_i$, $h_{min} = \min_{i \in \mathcal{R}} h_i$. Thus $\frac{h_{max}}{h_{min}} \leq B$.

Following [2], for each integer $k > B$, we describe a polynomial time algorithm $\mathcal{A}^k$ such that $\mathcal{A}^k(\mathcal{I}) \geq (1 - B/k)OPT(\mathcal{I})$ for any instance $\mathcal{I}$ of $\mathcal{P}$, where $\mathcal{A}^k(\mathcal{I})$ and $OPT(\mathcal{I})$ are respectively the value delivered by $\mathcal{A}^k$ and the value of the optimum solution for $\mathcal{I}$.

Algorithm $\mathcal{A}^k$: (Input: instance $\mathcal{I}$ and integer $k > B$.)

- Consider an instance $\mathcal{I}$. Draw a grid consisting of horizontal lines placed at a distance of at least h_{min} from each other, so that each rectangle is intersected by at least one line. It is easy to verify that

this can be done with at most n lines in time $O(n \log n)$. Indeed, let b_j be the smallest bottom coordinate of a yet not intersected rectangle. Then draw a line at $b_j + \varepsilon$ some small ε, if the distance between b_j and the last drawn line is bigger than h_{min}. Otherwise draw a line at a distance of h_{min} above the last line.

- Number the lines in the grid consecutively.
- For each $j \in \{0, ..., k-1\}$ compose a new instance $\mathcal{I}_j$ containing all the rectangles of $\mathcal{I}$ except those intersecting the lines from the set $\{i : i \bmod k = j\}$.
- Solve the problem for $\mathcal{I}_j$ for each $j \in \{0, ..., k-1\}$ to optimality using the dynamic programming procedure $\mathcal{DP}$ described in section 5.3.
- Return a solution with the best value.

The following theorem is an extension of the result first established in [31]:

Theorem 5.2.1. *For any instance $\mathcal{I}$ algorithm $\mathcal{A}^k$ delivers a solution with value at least $(1 - B/k)$ times the optimum.*

Proof. Let us denote the total weight of rectangles in a subset X by $w(X)$. Consider some optimum solution $S^{OPT}(\mathcal{I})$ and its intersections with $\mathcal{I}_j$, for some $j \in \{0, ..., k-1\}$: $S^{OPT}(\mathcal{I}) \cap \mathcal{I}_j$. The key observation is the following:

$$\sum_{j=0}^{k-1} w(S^{OPT}(I) \cap \mathcal{I}_j) \geq (k - B) w(S^{OPT}(\mathcal{I})).$$

To verify this, observe that each rectangle i can be intersected by at most B lines of the grid, since the distance between the lines is at least h_{min} and the biggest height of a rectangle in the instance is at most Bh_{min}. Therefore i is included in all instances $\mathcal{I}_j$ except at most B, thus in at least $k - B$ $\mathcal{I}_j$-s. Therefore the weight of each rectangle $i \in S^{OPT}$ enters at least $k - B$ terms of the sum and thus is counted at least $k - B$ times in the left-hand side of the equality.

Let $OPT(\mathcal{I}_j)$ and $OPT(\mathcal{I})$ be the optimum value of $\mathcal{P}$ for $\mathcal{I}_j$ and $\mathcal{I}$ respectively.

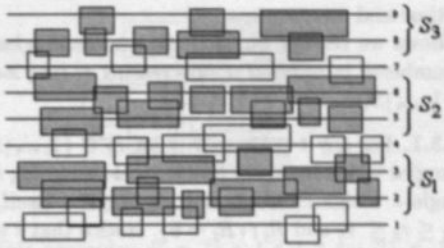

Figure 5.2: A subinstance $\mathcal{I}_1$ for $k = 3$.

Since $S^{OPT}(\mathcal{I}) \cap \mathcal{I}_j$ is a feasible solution to $\mathcal{P}$ for instance $\mathcal{I}_j$ we have: $w(S^{OPT}(\mathcal{I}) \cap \mathcal{I}_j) \leq OPT(\mathcal{I}_j)$. Using this and $w(S^{OPT}(\mathcal{I})) = OPT(\mathcal{I})$ we have:

$$\sum_{j=0}^{k-1} OPT(\mathcal{I}_j) \geq \sum_{j=0}^{k-1} w(S^{OPT}(\mathcal{I}) \cap \mathcal{I}_j) \geq (k - B)OPT(\mathcal{I})$$

Thus one of the $OPT(\mathcal{I}_j)$-s has to be greater or equal to $\frac{k-B}{k} OPT(\mathcal{I})$. □

5.3 Dynamic Programming Procedure $\mathcal{DP}$

Let us now describe how to solve a subinstance $\mathcal{I}_j$ ($j \in \{0, ..., k-1\}$) to optimality in time polynomial in n.

Observe that by construction $\mathcal{I}_j$ consists of at most $\lceil n/k \rceil$ subsets $(S_0, S_1, ..., S_{\lfloor n/k \rfloor})$ of rectangles, where any two rectangles from different subsets do not intersect each other (see Figure 5.2). Therefore an optimal solution on I_j can be found as the union of the optimal solutions on the subsets S_i, $i = 0, 1, .., \lfloor n/k \rfloor$.

Observe that a subset S_i contains all the rectangles of $\mathcal{I}$ lying strictly between the lines with numbers $k \cdot (i-1) + j$ and $k \cdot i + j$. Since each rectangle is intersected by some line, all the rectangles in S_i are intersected by $k-1$ lines situated between the lines $k \cdot (i-1) + j$ and $k \cdot i + j$. This implies that there can be at most $k-1$ nonoverlapping rectangles in S_i that intersect a common vertical line.

After giving some definitions we describe a dynamic programming procedure $\mathcal{DP}$ that finds an optimal solution to $\mathcal{P}$ for any $S \in \{S_0, S_1, ..., S_{\lfloor n/k \rfloor}\}$

using $O(|S|^k)$ time and memory space.

In what follows we restrict our attention to rectangles from the set S. Let them be numbered in order of nondecreasing r_i (the x-coordinate of the right edge) as $1, ..., |S|$.

Definition 5.3.1. For any $t \in \{1, ..., |S|\}$ let $P_t \subset \{1, ..., |S|\}$ be the subset of all the rectangles which intersect the vertical line going through the right edge of rectangle t without intersecting rectangle t itself, i.e., $P_t = \{i \in \{1, ..., |S|\} \mid l_i \leq r_t \leq r_i \text{ and } R_i \cap R_t = \emptyset\}$. Notice that $t \notin P_t$.

Definition 5.3.2. For any $t \in \{1, ..., |S|\}$ and for any set of nonoverlapping rectangles $P \subset \{1, ..., |S|\}$, let $IS(t, P)$ and $OPT(t, P)$ be respectively a maximum weight independent set in the set of rectangles $\{i \mid i \leq t \text{ and } R_i \cap R_j = \emptyset, \ \forall j \in P\}$ and its weight.

Theorem 5.3.1. *For any $t \in \{1, ..., |S|\}$ and for any set of nonoverlapping rectangles P such that $P \subset P_t$*

$$OPT(t, P) = \max\{OPT(t_1, (P \cup t) \cap P_{t_1}) + w(t), \ OPT(t_2, P \cap P_{t_2})\}, \quad (5.3.1)$$

and

$$IS(t, P) = \begin{cases} IS(t_1, (P \cup t) \cap P_{t_1}) \cup \{t\}, \\ \qquad \text{if } OPT(t_1, (P \cup t) \cap P_{t_1}) + w(t) \geq OPT(t_2, P \cap P_{t_2}) \\ IS(t_2, P \cap P_{t_2}), \ \text{otherwise} \end{cases}$$

where
$t_1 = \max\{j \mid j < t, \ R_j \cap R_i = \emptyset \ \forall i \in P \cup t\}$,
$t_2 = \max\{j \mid j < t, \ R_j \cap R_i = \emptyset \ \forall i \in P\}$,
$R_0 = \emptyset, \ P_0 = \emptyset, \ IS(0, P) = \emptyset$ *and* $OPT(0, P) = 0 \ \forall P$.

Proof. Recall that $OPT(t, P)$ and $IS(t, P)$ are the maximum weight independent set and its weight respectively in the set of all rectangles $i \leq t$, not intersecting rectangles from P. Since $P \subset P_t$ and t does not overlap with any rectangle of P_t, t does not overlap with any rectangle of P and is eligible in $IS(t, P)$. We can decide whether t belongs to $IS(t, P)$ based on the weight of $IS(t, P)$ in both the cases, which can be calculate as described below.

First, consider the hypothesis that $t \notin IS(t, P)$. Then $IS(t, P) = IS(t - 1, P)$. By definition of t_2 all rectangles j, such that $t_2 < j \leq t - 1$ (if such

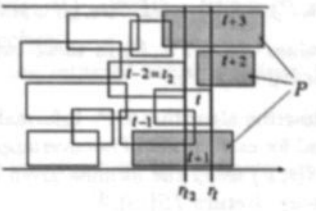

Figure 5.3: In this example: $P = \{t+1, t+2, t+3\} \in P_t$, $t_2 = t-2$, $P_{t_2} = \{t-1, t, t+1, t+3\}$, $(P \backslash P_{t_2}) = \{t+2\}$ and $(P \cap P_{t_2}) = \{t+1, t+3\}$.

j exist), overlap with some rectangles in P. Thus these rectangles can not enter a maximum weight independent set in the set of rectangles $\{i \mid i \leq t$ and $R_i \cap R_j = \emptyset$, $\forall j \in P\}$, hence $IS(t-1, P) = IS(t_2, P)$. Let us show that $IS(t_2, P) = IS(t_2, P \cap P_{t_2})$. Clearly, $P = (P \cap P_{t_2}) \cup (P \backslash P_{t_2})$. Consider the rectangles from the set $(P \backslash P_{t_2})$. By definition, rectangle t_2 does not overlap with any of the rectangles in P. Therefore all rectangles from P which intersect the line $x = r_{t_2}$ belong to P_{t_2}. This implies that no rectangle from $(P \backslash P_{t_2})$ intersects the line $x = r_{t_2}$. Further, since $P \subset P_t$, we know that all rectangles in $(P \backslash P_{t_2})$ intersect the vertical line $x = r_t$. Thus we have that the rectangles in $(P \backslash P_{t_2})$ intersect $x = r_t$ and do not intersect $x = r_{t_2}$. By ordering of the rectangles, $t_2 < t$ implies $r_{t_2} \leq r_t$. This means that rectangles in $(P \backslash P_{t_2})$ lie entirely in the right half-plane from the line $x = r_{t_2}$ and therefore can not overlap with any rectangle $i \leq t_2$, since these rectangles lie in the left half-plane from the line $x = r_{t_2}$ (see Figure 5.3). Therefore the set $\{i \mid i \leq t_2$ and $R_i \cap R_j = \emptyset$, $\forall j \in P\}$ is equal to the set $\{i \mid i \leq t_2$ and $R_i \cap R_j = \emptyset$, $\forall j \in P \cap P_{t_2}\}$, hence $IS(t_2, P) = IS(t_2, P \cap P_{t_2})$.

Thus we have, that if it is known that $t \notin IS(t, P)$,

$$IS(t, P) = IS(t-1, P) = IS(t_2, P) = IS(t_2, P \cap P_{t_2}), \text{ and hence}$$

$$OPT(t, P) = OPT(t_2, P \cap P_{t_2}).$$

The second hypothesis is $t \in IS(t, P)$. Then $IS(t, P) = t \cup IS(t-1, P \cup t)$. Applying similar arguments, obtain $IS(t-1, P \cup t) = IS(t_1, P \cup t) = IS(t_1, (P \cup t) \cap P_{t_1})$. Thus, if $t \in IS(t, P)$ we have

$$IS(t, P) = t \cup IS(t-1, P \cup t) = t \cup IS(t_1, P \cup t) = t \cup IS(t_1, (P \cup t) \cap P_{t_1}),$$

$$OPT(t,P) = w(t) + OPT(t_1, (P \cup t) \cap P_{t_1}).$$

Comparing now the values of $OPT(t,P)$ in these two cases, we determine whether $t \in IS(t,P)$ or $t \notin IS(t,P)$. □

Now we are ready to describe algorithm $\mathcal{DP}$. Informally, it works as follows: for each $t = 1, ..., |S|$ and for each subset of nonoverlapping rectangles $P \subset P_t$, find $OPT(t,P)$ and $IS(t,P)$ using the formula given in Theorem 5.3.1, and store them in the memory. Return $IS(|S|, \emptyset)$.

The formal description of the algorithm is shown in Figure 5.4.

1. For each t from 1 to $|S|$
   for each subset $P \subset P_t$ of nonoverlapping rectangles
   $t_1 \leftarrow \max\{j \mid j < t,\ R_j \cap R_i = \emptyset\ \forall i \in (P \cup t)\}$,
   $t_2 \leftarrow \max\{j \mid j < t,\ R_j \cap R_i = \emptyset\ \forall i \in P\}$,
   if $t_1 = 0$ then set $OPT(t_1, (P \cup t) \cap P_{t_1}) \leftarrow 0$,
   else retrieve $OPT(t_1, (P \cup t) \cap P_{t_1})$ from memory
   if $t_2 = 0$ then set $OPT(t_2, (P \cup t) \cap P_{t_2}) \leftarrow 0$,
   else retrieve $OPT(t_2, (P \cup t) \cap P_{t_2})$ from memory
   if $OPT(t_1, (P \cup t) \cap P_{t_1}) + w(t) \geq OPT(t_2, P \cap P_{t_2})$
   then $OPT(t,P) \leftarrow OPT(t_1, (P \cup t) \cap P_{t_1}) + w(t)$,
   retrieve $IS(t_1, (P \cup t) \cap P_{t_1})$,
   $IS(t,P) \leftarrow IS(t_1, (P \cup t) \cap P_{t_1}) \cup \{t\}$
   else $OPT(t,P) \leftarrow OPT(t_2, P \cap P_{t_2})$,
   retrieve $IS(t_2, P \cap P_{t_2})$,
   $IS(t,P) \leftarrow IS(t_2, P \cap P_{t_2})$
   store $OPT(t,P), IS(t,P)$ in memory.

2. Retrieve and return $IS(|S|, \emptyset)$.

Figure 5.4: Algorithm $\mathcal{DP}$.

Theorem 5.3.2. *Algorithm $\mathcal{DP}$ returns a maximum weight independent set of the set of rectangles $\{1, 2, ..., |S|\}$.*

Proof. First, notice that the algorithm is defined correctly, i.e., at the moment when it retrieves from memory $OPT(t_1, (P \cup t) \cap P_{t_1})$ or $OPT(t_2, P \cap P_{t_2})$, these values are available. Indeed, since $t_1 < t$ and $(P \cup t) \cap P_{t_1} \subset P_{t_1}$

is a subset of nonoverlapping rectangles, $OPT(t_1, (P \cup t) \cap P_{t_1})$ has been evaluated and stored before.

The optimality of the returned solution follows from the definition of $IS(|S|, \emptyset)$. □

Theorem 5.3.3. *The running time and memory consumption of algorithm $\mathcal{DP}$ are both $O(|S|^k)$.*

Proof. First let us answer the question how many different combinations (t, P) exists, such that $t \in \{1, ..., |S|\}$ and $P \subset P_t$ is a subset of nonoverlapping rectangles. Recall that there can be at most $k-1$ nonoverlapping rectangles intersecting a common vertical line in S. Since all the rectangles in P intersect the vertical line $x = r_t$ and do not overlap with rectangle t, which also intersect $x = r_t$, there can be at most $k-2$ rectangles in P. Therefore there are at most $\binom{|S|}{k-2} < |S|^{k-2}/(k-2)$ such subsets $P \subset P_t$, and thus at most $|S|^{k-1}/(k-2)$ allowed combinations (t, P). We can use a $(k-1)$-dimensional array to store all the combinations. Each (t, P) then corresponds to a unique address $(t, j_1, j_2, ..., j_q)$, where $\{j_1, j_2, ..., j_q\} = P$, $q \leq k-2$ and $j_1 < j_2 < ... < j_q$.

Observe that it requires $O(|S|)$ memory space to store $IS(t, P)$. (We assume that a number in the binary encoding can be stored in one unit of memory, and therefore storage of $OPT(t, P)$ does not require additional memory space.) Therefore the total memory requirement of the algorithm is $O(|S|^k)$.

Let us find a bound for the running time. Since there are at most $|S|^{k-1}/(k-2)$ allowed combinations (t, P), there are at most that many loops in the cycle of step 1. Note, that to find all the suitable subsets P does not cost additional work. The execution of the body of the cycle requires $O(|S|k)$ time. This is the time needed to find t_1 and t_2. Note, that retrieval and putting values by a particular address in the memory does not cost additional time since we are using a direct access memory structure. So, the total running time of the procedure $\mathcal{DP}$ is at most $O(|S|^k)$. □

Remark 5.3.1. The memory requirement of the procedure $\mathcal{DP}$ can be reduced to $O(|S|^{k-1})$ if we do not store $IS(t, P)$ for each allowed pair (t, P) in the memory, but instead store an indicator showing whether or not t belongs to $IS(t, P)$. This information allows us to restore $IS(|S|, \emptyset)$ at the end.

We conclude with the following theorem:

Theorem 5.3.4. *Algorithm $\mathcal{A}^k$ runs in at most $O(kn^k)$ time and requires at most $O(|n|^k)$ memory space for any $k > B$.*

Proof. We have to solve $\mathcal{P}$ for all the subinstances $\mathcal{I}_j$, $j = 0, ..., k-1$. For each of them we solve all the subinstances S_i, $i = 0, 1, .., \lfloor n/k \rfloor$, each of which is solved by algorithm $\mathcal{DP}$ in at most $O(|S|^k)$ time. Since

$$|S_0|^k + |S_1|^k + ... + |S_{\lfloor n/k \rfloor}|^k < (|S_0| + ... + |S_{\lfloor n/k \rfloor}|)^k \leq n^k,$$

each of $\mathcal{I}_j$ can be solved in $O(n^k)$ time. Since there are k of $\mathcal{I}_j$-s, the whole algorithms takes at most $O(kn^k)$ time.

Since algorithm $\mathcal{DP}$ requires at most $O(|S|^k)$ memory space and $|S| \leq n$, algorithm $\mathcal{A}$ requires at most $O(|n|^k)$ memory space. □

Bibliography

[1] Akcoglu, K., J. Aspnes, B. DasGupta, and M-Y. Kao [2002], Opportunity cost algorithms for combinatorial auctions, in: E. J. Kontoghiorghes, B. Rustem and S. Siokos (eds), *Applied Optimization: Computational Methods in Decision-Making, Economics and Finance*, Kluwer Academic Publishers.

[2] Agarwal, P.K., M. van Kreveld, and S. Suri [1998], Label placement by maximum independent set in rectangles. *Computational geometry: Theory and Applications* **11**: 209-218.

[3] Ahuja, R.K., T.L. Magnanti, and J.B. Orlin [1993], *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall.

[4] Aerts, J., and E.J. Marinissen [1998], Scan chain design for test time reduction in core-based ICs, *Proceedings of IEEE International Test Conference ITC '98*, Washington DC.

[5] Arora, S., C. Lund, R. Motwani, M. Sudan, and M. Szegedy [1998], Proof verification and intracatbility of approximation problems, *Journal of the ACM* **45**: 501-555.

[6] Ausiello, G., P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi [1999], *Complexity and approximation. Combinatorial Optimization Problems and their Approximability Properties*, Springer-Verlag, Berlin.

[7] Balas, E., and M. Padberg [1976], Set partitionning - a survey, *SIAM Review* **18**: 710-760.

[8] Bar-Noy, A., S. Guha, J. Naor, and B. Schieber [2001], Approximating the throughput of multiple machines in real-time scheduling, *SIAM Journal on Computing* **31**: 331-352.

[9] Bar-Noy, A., R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber [2001], A unified approach to approximating resource allocation and scheduling, *Journal of the ACM* **48**: 1069-1090.

[10] Bar-Yehuda, R., M.M. Halldorsson, J. Naor, H. Shachnai and I. Shapira [2002], Scheduling Split Intervals, in: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*: 732-741.

[11] Bar-Yehuda, R., S. Even [1981], A linear-time approximation algorithm for the weighted vertex cover problem, *Journal of Algorithms* **2**: 198-203.

[12] Bar-Yehuda, R., S. Even [1985], A local-ratio theorem for approximating the weighted set cover problem, *Annals of Discrete Mathematics* **25**: 27-46.

[13] Bellman, R. [1957], *Dynamic Programming*, Princeton University Press, Princeton, NJ.

[14] Berman, P. and B. DasGupta [2000], Multi-phase algorithms for throughput maximization for real-time scheduling, *Journal of Combinatorial Optimization* **4**: 307-323.

[15] Berman, P., B. DasGupta, S. Mithukrishnan, and S. Ramaswami [2001], Improved approximation algorithms for rectangle tilling and packing, in: *Procedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*: 427-436.

[16] Chan, T.M. [2003], Polynomial-time approximation schemes for packing and piercing fat objects, *Journal of Algorithms* **46**: 178-189.

[17] Calinescu, G., A. Chakrabarti, H.J. Karloff, and Y. Rabani [2002], Improved approximation algorithms for resource allocation, in: *Proceedings of the 9th IPCO Conference, Lecture Notes in Computer Science* **2337**: 401-414.

[18] Chuzhoy, J., R. Ostrovsky, and Y. Rabani [2001], Approximation algorithms for the job interval selection problem and related scheduling problems, in: *Proceedings of 42st Annual IEEE Symposium on the Foundations of Computer Science (FOCS'01)*: 348-356.

[19] Cornuejols, G. [2000], Combinatorial Optimization: Packing and Covering, lecture notes, published by SIAM (2001) in the CBMS-NSF Regional Conference Series in Applied Mathematics CBMS 74, can also be found on *www.integer.gsia.cmu.edu*.

[20] Erlebach, T., K. Jansen, E. Seidel [2001], Polynomial time approximation schemes for geometric graphs, in: *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*: 671-679.

[21] Erlebach, T. and F.C.R. Spieksma [2003], Interval selection: applications, algorithms, and lower bounds. *Journal of Algorithms*, **46**: 27-53.

[22] Fukuda, T., Y. Morimoto, S. Morishita, and T. Tokuyama [1996], Data mining using two-dimentional optimized association rules: schemes, algorithms and visualization, in: *Proceeding of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD'96)*: 13-23

[23] Fowler, R.J., M.S. Paterson, and S.L. Tanimoto [1981], Optimal packing and covering in the plane are NP-complete, *Information Processing Letters* **12**: 133-137.

[24] Fox, G., G. Lyzenga, S. Otto, J. Salmon and D. Walker [1988], *Solving problems on concurrent processors: general techniques and rectangle problems*, Prentice Hall, Englewood Cliffs, NJ.

[25] Garg, N., V.V. Vazirani, and M. Yannakakis [1997], Primal-dual approximation algorithms for integral flow and multicut in trees, *Algorithmica* **18**: 3-20.

[26] Gaur, D., T. Ibaraki, and R. Krishnamurti [2002], Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem, *Journal of Algorithms* **43**: 138-152.

[27] Goemans, M.X., and D.P. Williamson [1997], The primal-dual method for approximation algorithms and its application to network design problems, in: D.S. Hochbaum (eds), *Approximation algorithms for NP-hard problems*, PWC publishing company, Boston.

[28] Golumbic, M.C. [1980], *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, San Diego, California.

[29] Garfinkel, R.S., and G.L. Nemhauser [1972], Optimal set covering: A survey, in: *A.M. Geoffrion (ed), Perspectives on Optimization: A collection of expository articles*: 164-183.

[30] Hastad, J. [1999], Clique is hard to approximate within $n^{1-\epsilon}$, *Acta Mathematica* **182**: 105-142.

[31] Hassin, R., and Megiddo, N. [1991], Approximation algorithm for hitting objects with straight lines, *Discrete Applied Mathematics* **30**: 29-42.

[32] Hochbaum, D.S. [1997], *Approximation algorithms for NP-hard problems*, PWC publishing company, Boston.

[33] Hochbaum, D.S. [1997], Approximating covering and packing problems: set cover, vertex cover, independent set and related problems, in: D.S.Hochbaum (eds), *Approximation algorithms for NP-hard problems*, PWC publishing company, Boston.

[34] Hochbaum, D.S. [1982], Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing* **11**: 555-556.

[35] Hochbaum, D.S. and W. Maass [1985], Approximation schemes for covering and packing problems in image processing and VLSI, *Journal of the ACM* **32**: 130-136.

[36] Jain, K., and V. Vazirani [2001], Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation, *Journal of the ACM* **48**: 274-296.

[37] Kovaleva, S. [2003], Improved PTAS for the unit-height rectangle packing problem: a new dynamic programming procedure, in: H.K. Bhargava and N.Ye (eds), *Computational Modeling and Problem Solving in the Networked World, Proceedings of the Eighth INFORMS Computing Society Conference*, Kluwer Academic Publishers: 177-189.

[38] Kovaleva, S., and F.C.R. Spieksma [2001], Approximation of a geometric set covering problem, in: *Proceedings of the 12th Annual International Symposium on Algorithms and Computation (ISAAC '01), LNCS* **2223**: 493-501.

[39] Kovaleva, S., and F.C.R. Spieksma [2002], Primal-dual approximation algorithms for a packing-covering pair of problems, *RAIRO-Operations Research* **36**: 53-72.

[40] Phillips, C.A., R.N. Uma, and J. Wein (2000), Off-line admission control for general scheduling problems, in: *Proceedings of the eleventh annual ACM-SIAM Symposium On Discrete Algorithms (SODA'00)*: 879-888.

[41] Papadimitriou, C.H. [1994], *Computational Complexity*, Addison-Wesley, Reading, Massachussets.

[42] Poon, S-H., C-S. Shin, T. Strijk, and A. Wolff [2001], Labeling points with weights, In: *Proceedings of the 12th Annual International Symposium on Algorithms and Computing (ISAAC'01), LNCS* **2223**: 610-622.

[43] Raghavan, P., and C. Thompson [1987], Randomized rounding, *Combinatorica* **7**: 365-374.

[44] Raz, R., and Safra, S. [1997], A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP, in: *Proceedings of the 29th Annual ACM Symposium on Theory of Computation, ACM*: 475-484.

[45] Schrijver, A. [1986], *Theory of integer and linear programming*, John Wiley and Sons.

[46] Seznec, A. [1997], A new case for skewed-associativity. Technical Report RR-3208, INRIA, Rennes (France).

[47] Spieksma, F.C.R. [1999], On the approximability of an interval scheduling problem, *Journal of Scheduling* **2**: 215-227.

[48] Tardos, E. [1986], A strongly polynomial algorithm to solve combinatorial linear programs, *Operations Research* **34**: 250-256.

[49] Trotter, L.E. [1985], Discrete packing and covering, in: *O'heigeartaigh et al, annotated bibliography*: 21-31.

[50] Torng, E. [1998], A unified analisys of paging and caching, *Algorithmica* **20**: 175-200.

[51] Vazirani, V.V. [2001], *Approximation Algorithms*, Springer-Verlag, Berlin.

[52] Veeramachaneni, V., P. Berman, and W. Miller [2003], Alighning two fragment sequences, *Discrete Applied Mathematics* 127: 119-143.

[53] Vemuganti, R.R. [1998], Applications of set covering, set packing and set partitioning models, a survey, in: *Du and Pardalos (eds.) Handbook of Combinatorial Optimization* (vol 1), Kluwer Academic Publishers: 573-746.

Summary

This thesis develops approximation algorithms for several combinatorial optimization problems. These problems arise as special cases of the well known in combinatorial optimization set packing and hitting set problems, where the subsets in the input have a certain geometric structure. For illustration, consider the following examples:
Problem 1: given is a set of horizontal line segments placed arbitrarily in the plane. Find a maximum subset of segments the projections of which to the horizontal and vertical axes do not overlap mutually. See Figure 7.1 for illustration.

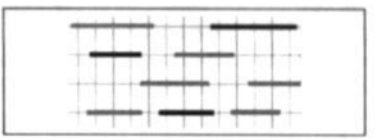

Figure 7.1: An instance of problem 1 and its optimal solution (shaded).

Problem 2: given is a set of horizontal line segments placed arbitrarily in the plane. Find a minimum number of horizontal and vertical lines, such that each segment is intersected by at least one line. Figure 7.2 illustrates this situation.

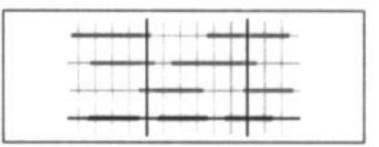

Figure 7.2: An instance of problem 2 and its optimal solution.

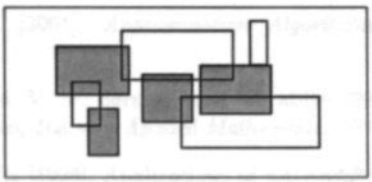

Figure 7.3: An instance of problem 3 and its optimum solution (shaded).

Problem 3: given is a set of axis-parallel closed rectangles of unit height and arbitrary length. Find the maximum number of non-overlapping ones. See Figure 7.3.

In this thesis we study the problems described above and more general versions of them including numerical parameters such as weights, demands and capacities. These problems find applications in molecular biology, printed circuit board manufacturing, cashing, load balancing, map labeling, database decision support and some other areas. Since all the studied problems are NP-hard, it is not realistic to find a polynomial (i.e, fast) algorithm that always returns an optimal solution. One of the ways is then to search for *approximation algorithms*, i.e. polynomial algorithms, which always find a solution of the value - for a minimization problem - at most δ times the value of an optimal solution ($\delta > 1$); for a maximization problem, the value of the solution obtain by the algorithm should be at least δ times the optimum value ($\delta < 1$). Parameter δ, called an approximation guarantee, determines how close are the algorithmic solutions to the optimum, and serves as a criterion of quality of an algorithm: the closer is δ to 1, the better. An algorithm with an approximation guarantee δ is called a δ-approximation algorithm.

Approximation algorithms developed in this thesis are based on different techniques. Since our problems allow for a natural integer programming formulation, such techniques as the primal-dual scheme and rounding of the linear programming relaxation (LP-rounding) can be used. Using the primal-dual scheme we develop approximation algorithms for two different generalizations of problem 1. These algorithms simultaneously construct a pair of feasible solution: one to a generalization of problem 1 and one to the related so-called dual problem, which is a corresponding generalization of problem 2. Each of those algorithms is shown to be a 1/2-approximation algorithm for problem 1 and a 2-approximation algorithm for problem 2 at the same

time.

Further we study algorithms for problem 2 based on the LP-rounding technique. These algorithms prove in our case to provide better approximation factors than factor 2, provided by primal-dual algorithms. However, in contrast to primal-dual algorithms, LP-rounding algorithms have to solve a linear programming problem, which can be a time consuming although polynomial operation. We present several LP-rounding algorithms for different generalizations of problem 2. Moreover, we investigate whether or not and to which extant an approximation guarantee of our algorithms can be improved by another LP-rounding algorithm.

A polynomial-time approximation scheme (PTAS) for a maximization (minimization) problem is a family of δ-approximation algorithms, for each $\delta < 1$ ($\delta > 1$). In this thesis we show that no PTAS can exist for a certain class of problems, including problem 2, unless $\mathcal{P} = \mathcal{NP}$.

For problem 3, studied in the last chapter of this thesis, a polynomial time approximation scheme has been known before. It uses a so-called "shifting technique" in combination with a dynamic programming procedure. By designing a new dynamic programing procedure we improve the running time of this PTAS and its memory requirement.

Samenvatting

Dit proefschrift beschrijft approximatie-algoritmen voor verscheidene combinatorische optimaliseringsproblemen. De bestudeerde problemen zijn speciale gevallen van het bekende 'set-packing' probleem en het 'hitting-set' probleem; de speciale gevallen kenmerken zich door een geometrische structuur die in de invoer aanwezig is. Beschouw, ter illustratie de volgende voorbeelden.

Probleem 1: gegeven is een verzameling van horizontale lijnstukken, willekeurig in het vlak geplaatst. Vind nu een zo groot mogelijke deelverzameling van lijnstukken zodanig dat de doorsnede van de projecties van elk der lijnstukken zowel op de horizontale as als op de verticale as leeg is (zie Figuur 8.1).

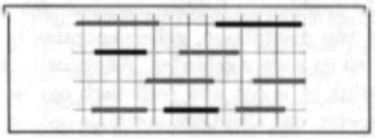

Figure 8.1: Een instantie van probleem 1 en de bijbehorende optimale oplossing (in grijs).

Probleem 2: gegeven is een verzameling van horizontale lijnstukken, willekeurig in het vlak geplaatst. Vind nu een minimaal aantal horizontale en verticale lijnen zodanig dat elk lijnstuk door tenminste één lijn doorsneden wordt (zie Figuur 8.2).

Probleem 3: gegeven is een verzameling van rechthoeken, parallel met de assen, die elk hoogte 1, en een willekeurige lengte hebben. Vind nu een maximaal aantal elkaar niet-overlappende rechthoeken (zie Figuur 8.3).

In dit proefschrift bestuderen we dergelijke problemen en hun generalisaties naar gewogen versies (waarbij een geselecteerd lijnstuk of rechthoek een gegeven gewicht oplevert) en/of naar versies met een gegeven vraag

About the Author

Sofia Kovaleva was born on Octorer 28, 1975 in Moscow, Russia. From September 1993 till June 1998 she studied mathematics and computer science at the Moscow State University, from which she received in June 1998 the degree Master of Science in Applied Mathematics.

From July 1999 till September 2003, Sofia Kovaleva was a Ph.D.-student at the Department of Mathematics of the Faculty of General Science at Maastricht University, The Netherlands. Her research in combinatorial optimization was supervised by Prof. dr. F.C.R. Spieksma and Prof. dr. ir. O.J. Vrieze.

Index