DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SEQUENTIAL RECTANGULAR PACKING PROBLEM IN WIRELESS TELECOMMUNICATIONS WITH FUZZY EXTENSIONS

by

Uğur ELİİYİ

October, 2013

İZMİR

SEQUENTIAL RECTANGULAR PACKING PROBLEM IN WIRELESS TELECOMMUNICATIONS WITH FUZZY EXTENSIONS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Statistics Program

by
Uğur ELİİYİ

October, 2013
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"SEQUENTIAL RECTANGULAR PACKING PROBLEM IN WIRELESS TELECOMMUNICATIONS WITH FUZZY EXTENSIONS"** completed by **UĞUR ELİİYİ** under supervision of **PROF. DR. EFENDİ NASİBOĞLU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Efendi NASİBOĞLU

Supervisor

Prof. Dr. Urfat NURİYEV

Thesis Committee Member

Asst. Prof. Dr. Emel KURUOĞLU

Thesis Committee Member

Prof. Dr. Can Cengiz ÇELİKOĞLU

Examining Committee Member

Asst. Prof. Dr. Ayşegül ALAYBEYOĞLU

Examining Committee Member

Prof. Dr. Ayşe OKUR
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor Prof. Dr. Efendi Nasiboğlu, for his guidance, insight and encouragement throughout this Ph.D. study. His inspiration and counsel throughout the period of my study at Dokuz Eylül University were invaluable. It was extremely helpful for my academic career to have a chance to work with him.

I would like to express my appreciation to the other members of my thesis committee. I would like to thank Prof. Dr. Urfat Nuriyev, Asst. Prof. Dr. Emel Kuruoğlu and Prof. Dr. Can Cengiz Çelikoğlu for their continuous support during my doctoral education at Dokuz Eylül University, for their valuable suggestions, guidance and continuous support.

I would also like to thank to my other professors/friends for their guidance during my studies at Dokuz Eylül University. Great thanks to Assoc. Prof Dr. Selma Gürler, Assoc. Prof. Dr. Özlem Oruç and Prof. Dr. Halil Oruç for providing continuous support and smile whenever I needed the most.

I would like to emphasize my thankfulness with ultimate respect and gratitude to my family. I would like to thank my mother Nevin and my late father Nizamettin Eliiyi, my sister Seda, my second mother and father Hamide and Ali Türsel from the bottom of my heart. Last but not least, the continuous support, care, and love of my wife Deniz Türsel Eliiyi is the source and encouragement of this work. Her love has always given me unlimited strength. She is my best advisor forever.

Uğur ELİİYİ

SEQUENTIAL RECTANGULAR PACKING PROBLEM IN WIRELESS TELECOMMUNICATIONS WITH FUZZY EXTENSIONS

ABSTRACT

In this thesis, a rectangular packing problem in telecommunications context is considered. Namely, we introduce a resource allocation modeling framework for a sequential two-dimensional packing problem, which may have direct applications in wireless telecommunications area pertaining to the IEEE 802.16 standard. The time dimension implied by the sequential consideration of frames adds a third dimension to the packing problem to some extent. We extend the common features of the frame packing problem to include realistic and state-of-the-art features of the current wireless data transfer processes. Three novel and representative mathematical programming models are developed for the problem, which are intended for contribution both to academic literature and professional practice. The developed models aim optimal usage of the physical layer defined by the standard, which involves data packages sent from a base station to a fixed or mobile user station. The data transmitted for each user are modeled as rectangular blocks, dimensions of which correspond to time duration and frequencies used in data transfer. Placement of these rectangular blocks in a sequence of identical rectangle frames is optimized by the developed models, aiming to maximize profit, minimize waste or minimize the rectangle count. Quality of service constraints such as maximum delay in transfer and minimum data transmission rates restrict the placement of variable-sized rectangles. We present the framework for all models, which handle demand partitioning and rectangle packing simultaneously. The foundations for fuzzy measures and parametrization are also proposed in this thesis, in order to mimic more realistic evaluation of actual network resources for practical problems. Thorough extensive experimentation, the performance of the developed models in terms of both solution times and quality are investigated. We also discuss alternative approaches to improve solution performances for the new models.

Keywords : Two-dimensional packing, three-dimensional packing, fuzzy optimization, telecommunications, scheduling

KABLOSUZ İLETİŞİMDE ARDIŞIK DİKDÖRTGENSEL PAKETLEME PROBLEMİ VE BULANIK UZANIMLARI

ÖZ

Bu tezde telekomünikasyon alanındaki bir dikdörtgensel paketleme problemi ele alınmıştır. Özel olarak, IEEE 802.16 standardına ait kablosuz iletişim alanında doğrudan uygulamaları olabilecek sıralı iki boyutlu bir paketleme problemi bazında bir kaynak tahsisi modelleme çerçevesi sunulmaktadır. Zaman boyutunun sıralı dilimlerin paketlenmesi şeklinde ele alınması probleme bir ölçüde üçüncü bir boyut kazandırmaktadır. Ayrıca, ele alınan dilim paketleme probleminin özellikleri günümüz kablosuz veri transferinde kullanılan güncel teknolojiyi kapsayacak şekilde zenginleştirilmiştir. Problem için hem akademik literature hem de sektöre katkı sağlaması hedeflenen özgün ve temsil gücü yüksek üç matematiksel programlama modeli geliştirilmiştir. Geliştirilen modeller, standart çerçevesinde tanımlanan ve bir baz istasyonundan sabit veya mobil kullanıcı istasyonlarına veri paketi gönderimini sağlayan fiziksel katmanın optimal kullanımını hedeflemektedir. İletilen veri paketleri her kullanıcı için boyutları transferde kullanılan süre ve frekans aralıklarına karşılık gelen dikdörtgensel bloklar olarak modellenmektedir. Bu blokların sıralı özdeş dikdörtgensel dilimler üzerine yerleştirilmesi, geliştirilen modeller tarafından karı maksimize edecek, atığı minimize edecek veya dikdörtgen sayısını minimize edecek şekilde eniyilenmektedir. Maksimum gecikme ve minimum veri iletim hızı gibi hizmet kalitesi kısıtları değişken boyutlu dikdörtgen blokların yerleşimlerini etkilemektedir. Çalışmada talep bölüştürme ve dikdörtgensel paketlemeyi aynı anda sağlayan modeller için teorik bir çerceve sunulmuştur. Ayrıca, pratik problemlerde gerçek ağ kaynaklarının değerlendirilmesinde faydalı olabilecek bulanık ölçüt ve parametreler için temel bir yapı oluşturulmuştur. Gerçekleştirilen kapsamlı sayısal deneylerle, geliştirilen modellerin çözüm süresi ve kalitesi bazında performansları ölçümlenmiştir. Bunun yanında olası yeni modeller için çözüm performansını geliştirebilecek alternatif çözüm yaklaşımları da tartışılmıştır.

Anahtar kelimeler : İki boyutlu paketleme, üç boyutlu paketleme, bulanık optimizasyon, telekomünikasyon, çizelgeleme

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

Three-dimensional bin packing problem (3D-BPP) is generally defined as the packing of a given set of rectangular-shaped items into the minimum number of three-dimensional rectangular bins without any overlapping. It is assumed that the items are packed orthogonally, namely, with each edge parallel to the corresponding bin edge. There are an unlimited number of identical three-dimensional containers (bins) having width W, height H, and depth D, which can be used for allocating a set or list of n rectangular-shaped items, each characterized by width $w_j \leq W$, height $h_j, \leq H$ and depth $d_j \leq D$ ($j \in J = \{1,..., n\}$.

The problem is NP-hard in the strong sense, since it is a generalization of the well-known (one-dimensional) bin packing problem (1D-BPP). In 1D-BPP, a set of n positive values w_j has to be partitioned into the minimum number of subsets so that the total value in each subset does not exceed a given bin capacity W. 1D-BPP is proven to be NP-hard in the strong sense (Garey & Johnson, 1979; Korte & Vygen, 2008). So there is little hope of finding even a pseudo-polynomial time optimization algorithm for 3D-BPP.

The 3D-BPP is closely related to other three-dimensional loading problems, such as Knapsack Loading and Container Loading. Naturally, it finds many industrial applications like packaging, mechanical and electronical design, vehicle and pallet loading, cutting of materials in production (cutting stock problem), loading area, multiprocessor scheduling, task assignment, and several scheduling problems (Eliiyi & Eliiyi, 2009).

In the knapsack loading of a container each item has an associated profit, and the problem is to choose a subset of the items that fits into a single container (bin) so that maximum profit is loaded. If the profit of an item is set to its volume, this corresponds to the minimization of wasted space. In the container loading version, all the items have to be packed into a single bin, having an infinite height. The problem

is thus to find a feasible solution that minimizes the height to which the bin is filled (Martello & Toth, 1990). There are several heuristics in the literature for these problems, some of which are to be mentioned in the following section (Aho et al., 1983; Pisinger, 2002).

Another important and frequently studied special case is the two-dimensional bin packing problem (2D-BPP), which again finds its obvious applications in areas related to the cutting context. Obviously 3D-BPP can also be generalized by adding dimensions like time (like for scheduling a bread oven usage) according to the nature of our problems.

The model of the problem can be adapted to a vast variation of problems not necessarily limited to packing and loading applications. Furthermore, the assumptions of the problem can be modified to represent the real-life situation more realistically. For instance, balancing conditions and weight distribution of the placed items, rotation constraints, geometry of the items or the bins should be considered in most cases. Even the amount of knowledge of the item placements in bins might be considered as in the cases of online bin packing problems.

The main goal of this thesis is to develop novel exact optimization models for solving the downlink (DL) subframe allocation problem in Mobile WiMAX (IEEE Std 802.16, 2009) telecommunications standard by satisfying real-network constraints such as maximum delay, minimum throughput and subscription priorities with power consumption and utilization objectives throughout a sequence of frames (Eliiyi & Nasibov, 2010). The downlink (transmission direction from the base station to the mobile stations) subframe allocation problem in Mobile WiMAX corresponds to a two-dimensional packing problem, and we approach this allocation problem through a sequence of DL subframes used in the data transfer to consider also other QoS constraints such as maximum delay.

As a result of a thorough review of the related literature and through a careful analysis of the current global technological developments, which will be fully

disclosed in the next chapter, it is recognized that new studies on modeling and solution to the problem of our concern is more than necessary. Including new and existing features of the Mobile WiMAX technology in the models for taking into account the most recent telecommunications industry requirements is essential. The success of the developed models and algorithms will also depend on the solution performances attained by sufficient experimentation with realistic problem instances.

These results and facts constitute the main motivation of this thesis, particularly considering the latest technology is of great value to the wireless communication domain, and an optimization approach in this area will contribute significantly to the existing literature. We extend the common features of the frame packing problem to include realistic and state-of-the-art features of the current wireless data transfer processes. The novel and representative mathematical programming models developed for the problem are intended for contribution both to academic literature and professional practice. The foundations for fuzzy measures and parametrization are also intended to be laid in this thesis, in order to mimic more realistic evaluation of actual network resources for practical problems. Thorough extensive experimentation, the performance of the developed models in terms of both solution times and quality are also investigated.

The problem considered in this thesis is an extended form of the multiple knapsack problem with identical capacities (Kellerer et al., 2005), as it will be explained in following chapters. However, the existence of the extra constraints and different objectives allows handling various decision making alternatives, and the variable item or demand sizes requires a nonlinear approach in modeling the problem, rendering it even harder.

By taking into account the time division structure of the resource allocation sequence and other features of the telecommunications standard, we extend the two-dimensional nature of the problem to three-dimensional by adding additional knapsack and partitioning constraints. In other words, the time element through the consideration of a sequence of frames adds a third dimension to this otherwise two-

dimensional packing problem. The continuous nature of the problem through time justifies the formulation through a sequence of frames rather than on a single frame. Since most transfers in a wireless telecommunication environment will not be completed in time captured by a single frame, we believe that the new problem definition in this thesis for partitioning and packing decisions involving multiple users over a sequence of frames, as well as novel mathematical formulations, offer significant contributions to related literature.

The outline of this thesis is as follows. In the next chapter, a review of the existing literature on the multi-dimensional bin packing problem is presented along with specific contributions in wireless telecommunications and fuzzy information. We define the problem and present the proposed mathematical models in Chapter 3, along with some existing preliminary results and approaches for the bin packing problem. Experimental computation is presented, analyzed and discussed in Chapter 4. A fuzzy perspective to the problem is presented in detail in Chapter 5. Finally, summary and conclusions are reported in Chapter 6, along with potential future research areas.

CHAPTER TWO
LITERATURE SURVEY

In this chapter, we present a thorough review of the existing literature on multi-dimensional bin packing problems.

2.1 Multi-dimensional Bin Packing Problem

The NP-completeness and NP-hardness of the multi-dimensional bin packing problem are presented by the prominent work of Garey & Johnson (1979). The authors proved the complexity of the problem by showing that the basic bin packing problem in one dimension (1D-BPP) contained 3-PARTITION as a special case, which is one of the basic NP-Complete problems in the strong sense. Due to this complexity, most of the literature on bin packing problems deals with approximation algorithms, heuristics, and their performances.

Given n items with weights w_j and n bins each with capacity c, the mathematical formulation of 1D-BPP was given as below by Martello & Toth (1990).

minimize $z = \sum_{i=1}^{n} y_i$

subject to $\sum_{j=1}^{n} w_j x_{ij} \le c y_i, \quad i \in N = \{1,...,n\},$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad j \in N,$$

$$y_i \in \{0,1\}, \qquad x_{ij} \in \{0,1\}, \qquad i \in N, j \in N,$$

where $y_i = \begin{cases} 1, \text{ if bin } i \text{ is used;} \\ 0, \text{ otherwise,} \end{cases}$ $x_{ij} = \begin{cases} 1, \text{ if item } j \text{ is assigned to bin } i; \\ 0, \text{ otherwise.} \end{cases}$

assuming $w_j \le c$, for $j \in N$.

The basic bin packing problem is considered as one of the milestones for analysis of the approximation algorithms. Effects of lower bounds and probabilistic aspects of the problem on the performance of the algorithms, as well as the worst-case and average-case behaviours of the algorithms for the 1D-BPP have been the primary research directions in the area of NP-hard problems (Johnson, 1973; Johnson et al., 1974; Baker & Coffman, 1981; Martello & Toth, 1990; Coffman et al., 1996). For a given list L of items and algorithm A, $A(L)$ being the number of bins used when A is applied to L, the optimum number of bins for a packing of L is denoted by $z(L)$, and the ratio $(A(L) / z(L))$ is denoted by $R_A(L)$. The absolute worst-case ratio R_A for algorithm A is then defined as:

$R_A = \inf\{r \geq 1: R_A(L) \leq r \text{ for all lists } L\}$,

and the asymptotic worst-case performance ratio as:

$R_A^{\infty} = \inf\{r \geq 1: \text{ for some } N > 0, R_A(L) \leq r \text{ for all } L \text{ with } z(L) > N\}$.

Additionally, if all items of the list have a maximum size of α, the bounded-size performance ratios can be defined as $R_A(\alpha)$ and $R_A^{\infty}(\alpha)$.

In one of the earliest studies on 3D-BPP, approached from a container loading perpective, an approximation algorithm was presented for minimizing total height with an asymptotic performance bound better than a conjectured one in the literature (Miyazawa & Wakabayashi, 1997). The authors defined an asymptotic performance bound α of an algorithm A as follows: If there exists a constant β such that for all item lists L, in which all boxes have height at most Z, then $A(L) < \alpha OPT(L) + \beta Z$ holds. Here, $A(L)$ denotes the height of the packing generated by the algorithm A when applied to the list, and $OPT(L)$ denotes the height of an optimal packing of L. Furthermore, if for any small ε and any large M, both positive, there is an instance L such that $A(L) > (\alpha - \varepsilon)OPT(L)$ and $OPT(L) > M$, then again α is called as the asymptotic performance bound of the algorithm A.

The same authors also studied a variant of the same problem, allowing a rotation in the z-axis (height), and developed four algorithms with respective to bottom shapes of the items. Special bottom shape chosen to be square, the algorithms they

presented had the same complexity O(*n log n*) as other known algorithms while having better asymptotic performance bounds (Miyazawa & Wakabayashi, 2000). A similar study introduced an efficient algorithm by formulating a geometrical model that reduced the general three-dimensional packing problem to a finite enumeration scheme (Faina, 2000). Proving the validity of the algorithm, the author proposed a numerical estimate of the corresponding asymtotic performance bound.

As in the case of 1D-BPP, it is often observed in the literature of 3D-BPP that new lower bounds are defined by comparing their performances and dominance relations with the previous ones and each other, and the worst-case analysis of these bounds are examined. In one such study, an exact algorithm was developed for selecting a subset of items that can be packed into a single bin, while maximizing the total volume packed (Martello et al., 2000). The authors used the bounds they found to obtain two approximation algorithms and an exact branch-and-bound algorithm. A later study by Fekete & Schepers (2001) aimed to obtain new fast lower bounds, based on dual feasible functions, and provided a general framework for establishing new bounds. Similarly, new lower bounds for the problem have been proposed by Boschetti (2004), where the items have fixed orientation. The bounds were extended by allowing 90º rotations, and experimental tests were evaluated for comparing the effectiveness of the new lower bounds.

One disadvantage of relying on worst-case analysis is that, in many real-world applications the worst case is almost never observed. Therefore, more meaningful results explaining the typical behaviours of the algorithms are necessary. Examining the average-case behaviours of the developed heuristics follows this requirement. In such an attempt, Coffman et al. (1988) investigated methods for obtaining formal probabilistic analyses of heuristics for makespan scheduling and one-dimensional bin packing, and presented many of the key results in these research areas.

In another study regarding the probabilistic aspect of the problem (Federgruen & van Ryzin, 1997), a unified probabilistic analysis was presented for a general class of bin packing problems, describing the objects by a given number of discrete or

continuous attribute values. Bins were defined as sets of objects, and the collection of feasible bins was merely required to satisfy some general consistency properties. The asymptotic optimal value was defined as the value of an easily specified linear program, whose size was independent of the number of objects to be packed. The analysis suggested that the developed heuristic run in linear time. The authors showed that the algorithm had both polynomially fast convergence and polynomial running time in several important cases, and they described how their results could be used to analyze a general vehicle routing model with capacity and time window constraints.

The average-case analysis of algorithms usually assumes independent, identical distributions for the inputs. Kenyon (1996) introduced the random-order ratio, a new average-case performance metric for bin packing heuristics, and proposed upper and lower bounds for this metric for the *Best-Fit* heuristics. An alternative definition of the random-order ratio was also introduced by Coffman et al. (2008). The authors shown that the two definitions yielded the same result for *Next-Fit*, and the random-order ratio of *Next-Fit* was 2, which was also equal to its asymptotic worst-case ratio.

For obtaining tighter bounds for problems of optimal packing within one or several containers, some new relaxations were developed that led to linear programming (LP) models. In such an effort, a column generation-based approach was discussed by Scheithauer (1999) along with computational results, and several relaxations were defined for container and multi-container loading problems. In another LP-based study, a mixed-integer linear programming (MILP) formulation of the problem that determined a filling of a big rectangular box with as many small rectangular boxes as possible was proposed (Padberg, 2000). The author presented a more general formulation that yielded a tighter LP bound of the convex hull than the previous MILP approaches for the problem.

The cube packing problem (CPP) is defined as a special case of 3D-BPP, where a given list of small cubes is placed into a minimum number of larger identical cubes. A parametric version of this problem was defined on online and offline algorithms,

and respective asymptotic performance bounds were presented by Miyazawa & Wakabayashi (2003). For a generalized *d*-dimensional cube packing (*d*-CPP) version of the problem, two approximation algorithms were developed by Kohayakawa et al. (2004). The first of the algorithms was shown to have an asymptotic performance bound that can be made arbitrarily close to $2 - (1/2)^d$, and the latter improved one could be made arbitrarily close to $2 - (2/3)^d$. The authors stated that these were the first results with non-exponential bounds.

In another heuristic approach, a multi-faced buildup technique was used in the packing procedure with no requirement for the packed boxes to form flat layers (Lim et al., 2003). The basic algorithm was augmented by a look-ahead strategy, yielding an average packing utilization that improved the existing benchmarks significantly. The same authors have formulated two heuristics dealing with homogeneous and heterogeneous categories in a later study (Lim et al., 2005), regarding box selection, space selection, box orientation and new space generation sub-problems. Lins et al. (2002) studied a specific problem of packing 3D boxes into an *n*-container, where the boxes can be packed in a given subset of their 6 possible positionings. The symmetries in the packings were analyzed through the use of an ordered set of three directed graphs with the same edges.

Techniques derived from bin packing algorithms have been used in several studies in other contexts. As an example, Coffman et al. (1978) described a fast bin packing-based algorithm for scheduling *n* independent tasks on *m* identical parallel processors in a nonpreemptive fashion while minimizing the makespan (total timespan required to process all the given tasks). Similarly, Garey et al. (1978) examined some special cases of the resource constraints in a scheduling problem, reducing the problem to BPP for determining the best performance guarantees for the developed approximation algorithms. In another study, the NP-hardness of the generalized fixed job scheduling problem was proven by transforming an instance of the problem to BPP (Fischetti et al., 1989). In addition, the authors used BPP solutions for obtaining tight lower bounds, which were then employed in a branch-and-bound algorithm for obtaining the optimal solution of the studied scheduling problem. Another interesting

study showed that the protein folding problem, which deals with the interactions within the amino acid chains that form a protein's well-defined three dimensional structure, is NP-complete, by a nontrivial transformation of a popular biophysical model (hydrophobic-hydrophilic) for protein folding to a modified bin packing model (Berger & Leighton, 1998).

In many studies, metaheuristic approaches for the BPP problem are employed. Such a study by Faroe et al. (2003) used a heuristic based on guided local search (GLS), starting with an upper bound on the number of bins obtained by a greedy heuristic. The proposed algorithm iteratively decreased the number of bins, each time searching for a feasible packing of the boxes. A general tabu search technique for the solution of 2D and 3D-BPP, as well as any of their variants requiring the minimization of the number of bins, was developed by Lodi et al. (2004), along with the implementation of the corresponding computer code. The user of the computer code was only requested to provide a procedure that gave an approximate solution to the actual variant to be solved. A two-level tabu search was presented in a recent study (Crainic et al., 2009), where the first-level aimed to reduce the number of bins, and the second optimized the packing of the bins. The latter procedure reduced the size of the search space, based on an interval graph representation of the packing, which was previously proposed by a study defining a combinatorial characterization of higher-dimensional orthogonal packing (Fekete & Schepers, 2004a).

Fekete & Schepers (2004a) presented a new approach for modeling packings, using a graph-theoretical characterization of the feasible packings. Their characterization allowed to deal with classes of packings that share a certain combinatorial structure, instead of having to consider one packing at a time. Using elegant algorithmic properties of certain classes of graphs, the characterization were used as the basis for a nice branch-and-bound framework. Based on this study, the same authors proposed a new approach for obtaining classes of lower bounds for higher-dimensional packing problems (Fekete & Schepers, 2004b), improving and simplifying several well-known bounds from previous literature. Following these two studies, a two-level tree search algorithm was developed in a later study for

solving higher-dimensional packing problems to optimality (Fekete et al., 2007). Computational results were reported, including optimal solutions for all 2D test problems from recent literature.

Miyazawa & Wakabayashi (2007) introduced approximation algorithms for the 2D- and 3D-BPP, and the 3D strip packing problem for a special case where each of the dimensions of the items to be packed was at most $1/m$ of the corresponding dimension of the recipient, m being a positive integer parameter. They analyzed the asymptotic performance of these algorithms. In a more recent study, Miyazawa & Wakabayashi (2009) presented approximation algorithms for the 3D-BPP and 3D strip packing problem, allowing 90º rotations. They presented the asymptotic performance bounds of both algorithms. The algorithms were designed for the more general case where the bounded dimensions of the bin given in the input were not necessarily equal. Moreover, they showed that the general versions of these problems were as hard to approximate as the corresponding oriented version.

Bansal et al. (2006) showed that, unlike the 1D case, the 2D-BPP could not have an asymptotic polynomial time approximation scheme (APTAS), unless $P = NP$. The authors presented an APTAS for the special case of packing d-dimensional cubes into the minimum number of unit cubes. They also proposed a polynomial time algorithm for packing arbitrary 2D rectangles into at most OPT square bins with sides of length $1+\varepsilon$, where OPT denotes the minimum number of unit bins required to pack these rectangles. As a corollary, they obtained the first approximation scheme for the problem of placing a collection of rectangles in a minimum-area encasing rectangle. Hifi (2002) studied the two-staged unconstrained 2D-BPP, and adapted some heuristics that used hill-climbing strategies, which produced a good trade-off between the computational time and the solution quality.

Lodi et al. (2002) reviewed solution approaches for the general 2D-BPP. They listed the heuristics and exact algorithms in the literature both for bin packing where the objective is to pack all the items into the minimum number of units, and for strip packing. In another noteworthy study, Martello & Vigo (1998) proposed new lower

bounds which are used within an exact branch-and-bound algorithm by investigating a well known lower bound and determining its worst-case performance. Kenyon & Rémila (2000) presented an approximation scheme for 2D strip packing problem based on a new linear programming relaxation. For any given ε, their algorithm found a feasible solution within a factor of $(1+\varepsilon)$ of the optimum with a polynomial running time complexity both in number of items and in $1/\varepsilon$. Caprara & Monaci (2004) on the other hand, dealt with the 2D knapsack problem (2KP), aimed at packing a maximum-profit subset of rectangles. They considered the natural relaxation of 2KP given by the 1KP, with item weights equal to the rectangle areas. They presented four exact algorithms based on that relaxation, proving the worst-case performance of the associated upper bound, and computationally compare them.

The vector scheduling problem and its dual problem, namely, the vector bin packing problem are also related problems to the BPP. Such problems naturally arise when scheduling tasks that have multiple resource requirements is of concern. The vector scheduling problem aims to schedule n d-dimensional tasks on m machines such that the maximum load over all dimensions and all machines will be minimized. The vector bin packing problem, on the other hand, seeks to minimize the number of bins needed to schedule all n tasks such that the maximum load on any dimension across all bins is bounded by a fixed quantity, e.g. 1. Chekuri & Khanna (2004) obtained a variety of approximability and inapproximability results, improving earlier known results for these problems.

In the container loading context, A Peak Filling Slice Push algorithm for the 3D-BPP was developed by Maarouf et al. (2008). The algorithm recursively divided the container into smaller slices and then filled each slice with boxes before pushing them to minimize the wasted space. The distributor's or multi-pallet loading problem was considered by Terno et al. (2000). The objective of finding the best space utilization was restricted by a list of practical aspects, such as technological constraints, weight distribution over the pallet, and stability aspects. A branch and bound based heuristic was developed for the 3D case, using a layer-wise loading strategy with optimal 2D loading patterns.

A novel heuristic based on wall-building approach was proposed by Pisinger (2002) for maximizing the packed volume. The heuristic decomposed the problem into a number of layers, which again were split into a number of strips. The packing of a strip was formulated as a Knapsack Problem with capacity equal to the width or height of the container. The depth of a layer as well as the thickness of each strip was determined through a branch-and-bound approach where at each node only a subset of the branches was explored. Several ranking rules regarding layer depths and strip widths were presented and compared for homogeneous and heterogeneous instances. Large-sized instances with a total box volume up to 90% were solved to optimality, and average fillings of container volume exceeding 95% were obtained for these instances.

For dealing with a major drawback considering many practical issues in container loading problems, some studies took into account the stability of the packed items or the weight distribution of the cargo (Castro Silva et al., 2003; Davies & Bischoff, 1999). In the latter study, the authors considered postprocessing approaches, putting forward a new container loading heuristic. The heuristic was evaluated against several existing approaches, and it was shown to be capable of producing loading arrangements which combined high space utilization with an even weight distribution of the cargo. In a more recent study, a new heuristic approach was proposed for tackling problems where the cargo had varying degrees of load bearing strength (Bischoff, 2006). In such cases, the placement rules must ensure that the weight resting on an item remains below the maximum it can withstand without suffering crushing damage. Limiting the time required to produce a good solution and the amount of technical expertise needed by the user are some key considerations. The experimental test results of the study by Bischoff (2006) demonstrated that the heuristic outperformed other approaches that had been suggested for this type of problem, and that it also performed well on some problems where load bearing strength was not an issue.

The issue of balancing conditions and items consisting of clusters of parallelepipeds (mutually orthogonal, i.e. tetris-like items) is quite frequent in space

engineering, and was studied in a real-world application that dealt with an Automated Transfer Vehicle project funded by the European Space Agency (ESA). An MILP-based heuristic was proposed that solved the reduced MILP model (Fasano, 2004; Fasano, 2008). Dealing with non-standard 3D-packing issues, a recursive procedure based on a non-blind local search philosophy was developed. The concept of abstract configuration, concerning the relative positions between items, was also introduced in this study. The heuristic generated a sequence of good abstract configurations and iteratively solved a reduced MILP model, by fixing the relative positions of the items corresponding to the current abstract configuration.

Several reported algorithms assume the online version of the BPP. In the online BPP, only the layout of the previous items/boxes on the partially filled container and the size of the box to be placed next are known at each stage, but no information is available about the forthcoming items. This corresponds to situations in which items are physical objects, and there is no intermediate space to store them before placing them in the bins. Such a problem is encountered when robots are used instead of the traditional manual operation in pallet loading. A bin packing algorithm that can construct its packings under such circumstances is called an online algorithm and the following initial solution approaches in the literature propose algorithms in that main category.

The simplest approximate online approach to the bin packing problem was the *Next-Fit* (NF) algorithm (Johnson, 1973). In this algorithm, the first item is assigned to bin 1, which is the current bin at the start, and each arriving item with increasing indices 2,…,n are considered in order whether it fits the current bin or not. If it does, the item is assigned to the current bin; otherwise it is placed in a new bin, which then becomes the current one. The time complexity of this approximation algorithm is $O(n)$, and it is proved that the absolute worst-case performance ratio is 2. Thus, for any instance I of BPP, the solution value $NF(I)$ given by the algorithm and the optimal solution value $z(I)$ fulfill the bound $NF(I) \leq 2z(I)$. An example for NF is given in Figure 2.1.

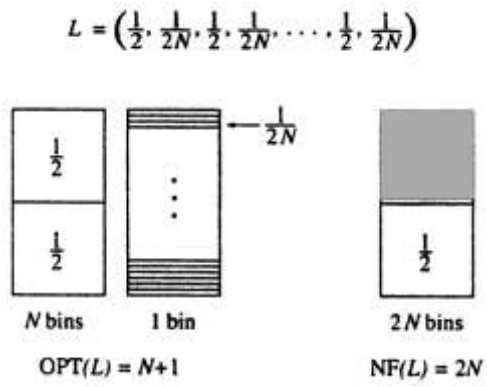


Figure 2.1 Worst-case example for the *Next-Fit* algorithm (Source: Coffman et al. 1996, p. 49)

Since the list given in the example above has no item with size greater than 1/2, it can be concluded that $R_{NF}^{\infty}(\alpha) = 2$ for all $\alpha \geq 1/2$. Moreover, as α decreases under that bound, so does $R_{NF}^{\infty}(\alpha)$, specifically we have $R_{NF}^{\infty}(\alpha) = 1/(1 - \alpha)$ for $\alpha \leq 1/2$ (Johnson, 1973).

The algorithm *First-Fit* (FF) by Ullman (1971), similarly handles the items according to their increasing indices, and assigns each item to the smallest-indexed initialized bin into which it fits. When the current item considered cannot fit into any initialized bins at hand, a new bin is utilized. Using the notation above for NF, Johnson et al. (1974) proved that $FF(I) \leq (17/10)\, z(I) + 2$, and for some instances where $z(I)$ is arbitrarily large, $FF(I) > (17/10)\, z(I) - 8$. Hence the asymptotic worst-case performance ratio R_{FF}^{∞} for algorithm FF is 17/10.

Best-Fit (BF) algorithm was developed (Ullman, 1971; Eilon & Christofides, 1971; Garey et al., 1972) by modifying FF through assigning the current item to the feasible bin having the smallest residual capacity. If there is no such bin, a new bin is initiated. If there is more than one having the same capacity, the item is assigned to the one with the lowest index. BF has the same worst-case performance ratios as FF (Johnson et al., 1974). FF and BF have the same time complexity as $O(n\log n)$, achieved by employing a 2-3 tree approach (Aho et al., 1983; Martello & Toth, 1990).

All the three online algorithms described above can be extended with an offline approach assuming that the sizes (w_j) of the items are known beforehand and items are sorted and indexed accordingly as $w_1 \geq w_2 \geq \ldots \geq w_n$. Then applying NF, FF or BF to these items, one obtains *Next-Fit Decreasing* (NFD), *First-Fit Decreasing* (FFD) and *Best-Fit Decreasing* (BFD) algorithms having the same time complexity as $O(n\log n)$ and respectively with asymptotic worst-case performance ratios, 1.691, 1.222 and 1.222 (Johnson et al., 1974; Baker & Coffman, 1981; Coffman et al., 1996). In general, the asymptotic worst-case ratio difference between packing rules gets smaller as the lists are ordered in decreasing item size.

Another study on this problem focused on the packing of boxes distributed to different customers from a central packaging depot (Hemminki et al., 1998). The boxes had various sizes and they arrived on a conveyor belt one at a time, where it was not allowed to move the boxes already placed. The objective was to produce efficient and stable loads by an online packing algorithm. The algorithms the authors developed were based on the above BF and FF principles.

Another special case of the multi-dimensional problem is the 2D variable-sized BPP. In this problem, the packing a set of rectangular items into a set of rectangular bins is of concern. The bins have different sizes and different costs, and the objective is to minimize the overall cost of the bins used for packing the rectangles. Pisinger & Segurdi (2002) presented an integer-linear programming formulation of this problem and employed Dantzig–Wolfe decomposition, introducing very good quality lower bounds for the problem justified by a computational study. They also developed a branch-and-price-based exact algorithm for the problem. In another variation of the BPP considering different shapes, given a fixed set of identical or different-sized circular items, the problem deals with finding the smallest object within which the items can be packed. Circular, triangular, squared, rectangular and also strip objects were considered by Birgin & Sobral (2008). They treated 2D and 3D problems, presenting twice-differentiable models for all these problems and employing a strategy to reduce the complexity of evaluating the models.

Instead of fixed sized bins, the bin-stretching problem allows to pack the items while stretching the size of the bins as least as possible. Some studies presenting various online algorithms for the bin-stretching problem determined best lower bound performances for specific stretching factors, and pointed out that the bin-stretching problem is also equivalent to the classical scheduling (load balancing) problem, in which the value of the makespan (maximum load) is known in advance (Azar & Regev, 2001; Epstein, 2003). Their results may be used as comparative measures for a possible bin size fuzzification scheme. In a similar variation of the BPP called the extensible BPP, the number of bins is specified as part of the input, and bins may be extended to hold more than the usual unit capacity. The cost of a bin is taken as 1 if it is not extended, whereas it is taken as the size of the bin if it is extended. With a total cost minimization objective for this problem, the study by Coffman & Lueker (2006) presented a fully polynomial time asymptotic approximation scheme (FPTAAS) with comments on complexity of achieving stronger results.

A related but complicated study on proof-assisted property testing considered approximate probabilistically checkable proof (PCP) techniques for the multidimensional bin-packing problems (Batu et al., 2005). Particularly, the authors showed how a verifier could be quickly convinced that a set of multidimensional blocks can be packed into a given number of bins, extending a heaviness property investigated in the one-dimensional case to the multidimensional case.

The most common metric used to evaluate the effectiveness of a packing technique is generally the percentage of the space used. An inherent limitation of this metric is its inability to differentiate between two different packing arrangements of the same set of objects. Dickinson & Knopf (2000) proposed the point moment metric for both the 2D and 3D cases. The metric is based on evaluating the compactness of the remaining free space in a packing arrangement. This measure is the ratio of a defined moment calculated for the current free space and the initial free packing space. The developed metric can also be extended to *n*-dimensional packing problem.

A typology of problems in an area provides the basis for a structural analysis of the underlying problem types, the identification and definition of standard problems, the development of models and algorithms, problem generators, etc. Fortunately, such an extensive review study can be found in literature for the area of cutting and packing (C&P) problems (Wäscher et al., 2007). Defining common identical structures of cutting and packing problems, the authors categorised all problem types in the literature. For instance, both packing and cutting problems deal generally with two sets of elements, namely a set of large objects (input, supply) and a set of small items (output, demand), which are defined exhaustively in one, two, three or an even larger number (n) of geometric dimensions. Hence, apart from the studies mentioned in this section, the interested reader can refer to this study for a thorough review of the C&P literature.

The next section reviews the existing studies on the fuzzy versions of the BPP.

2.2 Fuzzy Approaches

There are very few studies in literature on multi-dimensional BPP and the related variants that take into account the fuzziness of the constraints, objectives, and relations between the items and/or bins. This lack of interest may be attributed mainly to the tight correspondence of the problem modeling to real-life physical problems, allowing or seeking only for optimal placements.

However, in a lot of situations, the defined problems require to be satisfied by alternative admissible solutions, taking into account the human or expert choices. Similar to the multi-objective decision approaches, sources of fuzziness are to be determined, and should be employed for obtaining quick and nice solutions. Some examples from the literature on fuzzy BPP are presented below.

As mentioned in the previous section, many metaheuristic approaches are used for obtaining near-optimal solutions for 3D-BPP. The same approaches also attracted attention for the fuzzy versions of the problem. For instance, a genetic algorithm

(GA) was presented by Runarsson et al. (1996) for solving an online dual BPP using fuzzy objectives. In the dual BPP, the items are packed into a maximum number of bins, assuring a minimum weight for each bin. The dynamic class the authors defined assumed, as other online algorithms, that the items must be packed sequentially. However, they assumed that more than one future item at a time can be seen. The number of bins being packed at any time was fixed, and a bin was replaced by an empty one as soon as it was filled. Their results showed that the fuzzy packing scheme was essential to solving the problem, and due to the nature of the problem the GA behaved closely to that of a micro-GA.

An interesting study by Dexter et al. (1999) considered a medical decision-making process concerning the operating room (OR) utilization in a hospital. The goal was to use computer simulation to evaluate ten scheduling algorithms from the management science literature to determine their relative performance at scheduling as many hours of add-on elective cases as possible into the open OR time. The authors collected data from a surgical services information system for hours of open OR time available for add-on cases in each OR each day, and for the duration of each add-on case. These empirical data were used in computer simulations of case scheduling, to compare algorithms appropriate for the variable-sized BPP with bounded space. Here, the variable size referred to different amounts of open time for each OR. The algorithm that maximized OR utilization was *Best Fit Descending* with fuzzy constraints, and this algorithm achieved OR utilizations that were 4% larger than the algorithm with the poorest performance.

Kim et al. (2001) considered another fuzzy BPP that dealt with packing non-rigid rectangles into an open rectangular bin, as in the strip packing problem. The authors employed fuzziness in the height dimension by using triangular fuzzy numbers. The goal of the fuzzy BPP was to minimize both the height of a packing and the extra cost due to the reduction of each piece. The total cost of the problem was represented as the sum of the height cost and the extra cost due to reductions of the pieces, which was called the reduction cost. Reducing the height of an item decreased the overall height cost but increased the reduction cost due to lower quality of the item. A closed

form solution was presented for the fuzzy BPP, in which fuzzy numbers were triangular and the reduction cost was formulated using a quadratic function.

In apparel/textile industry, manufacturers develop standard allowed minutes (SAMs) databases on various manufacturing operations in order to facilitate scheduling, while effective production schedules ensure smoothness of downstream operations. However, as the parameters in an apparel manufacturing environment are fuzzy and dynamic, these rigid production schedules based on SAMs become futile in the presence of any uncertainty. A recent study in this context (Mok et al., 2007) proposed a fuzzification scheme to fuzzify the static standard times so as to incorporate some uncertainties, in terms of both job-specific and human related factors, into the fabric-cutting scheduling problem. A GA-based optimization procedure was proposed to search for fault-tolerant schedules using genetic algorithms such that makespan and scheduling uncertainties were minimized. Experimental results using two sets of real production data indicated that the genetically optimized fault-tolerant schedules not only improved the operation performance but also minimized scheduling risks.

Regarding the ranking and defuzzification techniques in literature for possible use in the fuzzy versions of the BPP, the methods proposed by Dubois & Prade (1983), and Fortemps & Roubens (1996), may be employed as in scheduling problems.

One of the important contributions to fuzzy BPP considered a new statement of the BPP with the evaluation of the packing quality under fuzzy source constraints (Nasibov, 2004). A finite interactive algorithm for solving the problem was developed and its accuracy was justified. The author presented estimates for the a priori determination of the maximum degree of quality of packing that accelerate the process of the solution of the problem. Besides the items to be allocated, there were two sets of containers in this study. The first set included m main containers, each represented by S_j, where j=1,...,m; and the second set had only one reserve container S_{m+1}. The fuzzy relations between the items and the containers imposed certain constraints on the placement of the items. Four relations were defined reflecting the

degree of mutual attachment of items, the mutual compatibility of items, the mutual attachment of an item to a container, and the mutual compatibility of an item to a container, resulting in matrices taking values in the interval [0, 1]. Containers were to be filled with respect to certain conditions (e.g. sufficient degree of filling factor so that the consistency degree of the final packing was maximized), and a certain classical total indicator/measure, such as volume or weight, was minimized for the items placed in the reserve container.

A more recent study by Nasibov (2007) specified the task implementation competence of a group of executives in the form of fuzzy relations for high-performance allocation of tasks. The two optimization criteria considered in the study were the maximization of the aggregated degree of competence of the entire allocation, and the maximization of the degree of the overall level of employment of standard executives. Aggregation was performed by means of the Hurwicz operator and the Ordered Weighted Average (OWA) operator, which rendered the model more flexible and allowed the decision-maker to pursue different strategies. A two-stage heuristic algorithm was proposed for the solution of the problem, which was a generalized variant of task allocation, which is a special case of BPP. An analysis of different algorithms and assessment of the results of computational experiments were conducted. In the general version of task allocation, the skill of the executives was not taken into account, only the quantitative constraints of the workload were considered. However, through the use of the theory of fuzzy sets, it was possible to take into account high-performance relations between tasks and executives in the form of fuzzy relations.

Through careful analysis of the studies in literature, the following main application areas of multi-dimensional BPP problems and their fuzzy versions are identified as follows:

- Loading problems (container, vehicle, pallet, cargo),
- Job scheduling, budgeting problems,
- Packaging design, loading area,
- Cutting stock, trim loss problem, textile/apparel applications

- Mechanical and electronical design (nanotechnological),
- Multiprocessor scheduling, load balancing,
- Resource (memory, space, operating room) allocation,
- Task allocation,
- Genetic engineering, biophysics,
- Parallel query optimization,
- Wireless telecommunications (data transfer),
- Assessment of approximation algorithm complexities.

Among these application areas, the multi-dimensional packing problems in wireless telecommunications dealing with resource allocation in wireless data transfer constitute the main focus area of this thesis. For this reason, the following section presents a detailed review of the studies on multi-dimensional BPP, pertaining specifically to the wireless telecommunications domain.

2.3 Multi-dimensional Packing in Wireless Telecommunications

In order to better grasp the contributions by the studies in this area, we first present some basic terminology below.

The WiMAX (Worldwide Interoperability for Microwave Access) is one of the broadband wireless access technologies, which is based on the IEEE 802.16 standard (2009). This standard with its last ammendment (IEEE 802.16m, 2011) achieved the distinction of being recognized as a true 4G technology by the International Telecommunication Union (ITU). 802.16m, which is also called as WirelessMAN-Advanced or WiMAX-2, has been recently (March 31st, 2011) approved by IEEE as a new global standard for mobile WiMAX. The aim of this technology is providing wireless data transfer using various transmission modes, including point-to-multipoint connections and portable or fully mobile cellular type access. A WiMAX base station (BS) can provide broadband wireless access in range up to 50 kms for fixed stations and 5 to 15 kms for mobile stations (MS) with a maximum data download rate of up to 1 Gbps and upload rate of 100 Mbps (IEEE 802.16m, 2011).

Apart from its technical qualities, three features of WiMAX mainly attract attention of the researchers (So-In et al., 2009a; Necker et al., 2008). These are:

(1) the use of Orthogonal Frequency Division Multiple Access (OFDMA),
(2) multiple Quality of Service (QoS) classes that define priorities between data, voice and video transmissions for satisfying service guarantees, and
(3) the so-called Media Access Control (MAC) scheduler of the BS, which utilizes the first two aspects.

The first feature, namely OFDMA, is based on Orthogonal Frequency Division Multiplexing (OFDM), which is a spread-spectrum technique for state-of-the-art broadband wireless systems. The frequency spectrum used for communications is divided into a large number of frequency subcarriers to serve different terminals in the same time intervals and through the same physical channels.

The second feature, namely the QoS classes, allows the BS to classify the terminals according to parameters like minimum throughput requirements regarding data transmission rates, and the delay constraints. The QoS support in wireless network connections is much more demanding than in wired networks due to its highly variable and unpredictable nature, depending both on time and locations of the terminals.

The last feature of WiMAX, i.e. the MAC scheduler, constitutes the main focus of this thesis, and is responsible for two tasks concerning resource allocation. The first is to determine the terminals that will be served in a specific time frame, thus forming a service queue while also determining the amount of data required for each terminal. This task depends mainly on the QoS parameters used for that particular network, and is studied under names like burst construction (Ohseki et al., 2007) or packet scheduling (Wongthavarawat & Ganz, 2003). The second task of the MAC scheduler is to assign time and frequency intervals to each terminal, referred to as frame packing, packet mapping or burst mapping in the literature (Ben-Shimol et al., 2006; Bacioccola et al., 2007; Ohseki et al., 2007; Necker et al., 2008; So-In et al., 2009b). Henceforth, we prefer to use the term “frame packing” in our study.

IEEE 802.16 standard does not impose any specific admission control mechanisms or resource allocation mechanisms for the scheduler. Therefore, the taks of the MAC scheduler, or scheduling in general, becomes a significant research area for all WiMAX equipment makers and network service providers. So-In et al. (2009a) provided an extensive review for the main issues considered in designing these mechanisms. The authors explained the physical layers and QoS classes defined in the IEEE 802.16 standard, and classified the schedulers based on channel state awareness. They listed various criteria and scheduler design factors with respect to different QoS considerations, which shape the algorithms and heuristics proposed in the literature. Throughput maximization and minimizing power consumption are among the major criteria while ensuring system scalability regarding the algorithm complexity.

In frame packing, the frame to be packed is a two-dimensional structure defined in the IEEE 802.16 standard using OFDMA, in which the frequency channel is divided into multiple subcarriers. These subcarriers are grouped into a number of subchannels. The time axis of the frame typically covers a 5 milisecond (ms) duration. Each user terminal is allocated a certain number of subchannels for a certain amount of time. Bidirectional data transfer can be achieved in two ways (So-In et al., 2009a):

(1) By frequency division duplexing (FDD) in which uplink (MS-to-BS direction) and downlink (BS-to-MS direction) transfers use different frequency bands,
(2) By time division duplexing (TDD) in which the uplink (UL) traffic follows the downlink (DL) traffic in time dimension.

Hence, each frame consists of DL and UL subframes. Figure 2.2 shows these in TDD mode. In FDD mode, the DL and UL subframes go parallel in time.

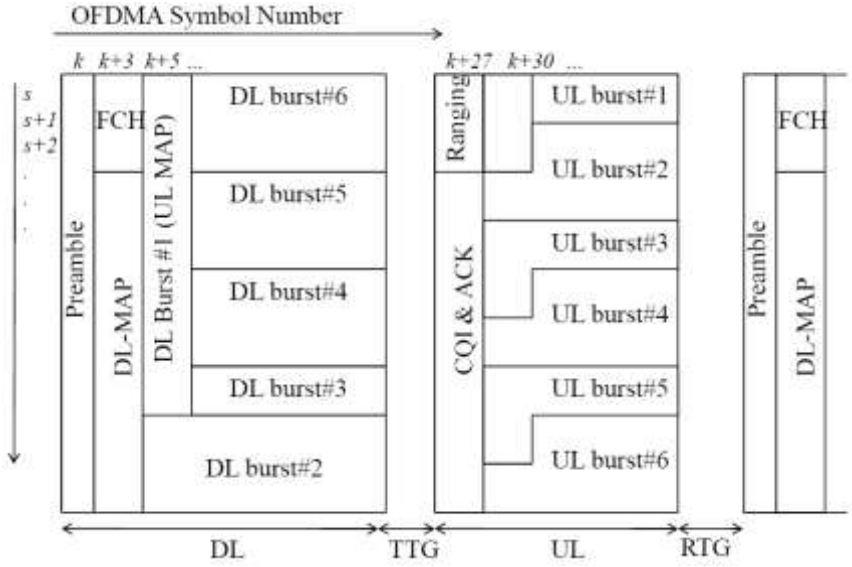


Figure 2.2 A sample OFDMA frame structure in TDD mode (Source: So-In et al., 2009b, pp. 1)

User terminals are mapped to rectangular areas (referred to as bursts) in the OFDMA frame. The unit of allocation in WiMAX is a "slot". The data transmission capacity of a slot depends upon the subchannelization, or modulation and coding methods. The standard allows more than one burst per mobile station, which increases the downlink map (DL-MAP) overhead, that is, the amount of control data. The DL-MAP and uplink map (UL-MAP) define the burst-start time and burst-end time, modulation types and forward error control mechanisms for each MS.

Other terms and parts of the frame shown in Figure 2.2 can be explained as follows: The preambles are used for time synchronization. Frame Control Header (FCH) defines MAP lengths and usable subcarriers. Transmit-receive Transition Gap (TTG) is the duration that the BS switches from transmit to receive mode, and Receive-transmit Transition Gap (RTG) occurs when the BS switches from receive to transmit mode. Channel Quality Indicator (CQI) is used to pass the channel state condition information. Each OFDM symbol is 102.8 microseconds (μs), so a 5ms frame is equal to a duration of 48.6 OFDM symbols. (So-In et al., 2009a; So-In et al., 2009b). The standard also allows more than one connections packed into one burst (burst compaction). However, in each case, the MAC scheduler is required to place

rectangles in the frame as efficiently as possible under technology and traffic-specific constraints.

The related multi-dimensional BPP literature is mainly on TDD systems, and in particular, on the packing of the DL subframe. Wongthavarawat & Ganz (2003) performed one of the very few studies on UL packing. They presented an UL packet scheduling algorithm and admission control policy to provide QoS support in terms of bandwidth and delay bounds for different traffic classes. They investigated the factors which affect the network performance through a simulation model.

Most studies on the DL subframe allocation problem aim to maximize the packing performance in a single frame, averaging and simplifying the QoS constraints of service users such as demand levels, subscription priorities, minimum transmission rate or maximum allowable delay (So-In et al., 2009a). As it was stated before, the BS performs periodically the tasks of selecting which data packets will be transmitted to the users (mobile stations) in the forthcoming frame, usually based on QoS requirements summarized above; and grouping the selected packets to rectangular data regions and allocating them to the frame disjointly in a way that they do not overlap with each other. Hence, there is a two-stage approach comprised of two periodical scheduling decisions, the first one about selecting the transmission target stations and their corresponding physical features on the frame structure, and second one on how to place the corresponding rectangles on the frame without any overlap. Cohen & Katzir (2008) named these stages as macro and micro scheduling problems respectively, and analyzed their computational complexity aspects developing two interdependent approximation algorithms. They transformed the macro scheduling problem into the Multiple Choice Knapsack Problem and showed that it can be solved in pseudo-polynomial time. Rather than achieving higher resource allocation rates, they aimed at simplifying the second phase by reducing the number of mobile stations to be packed in the DL subframe without considering any priorities or profits.

Ben-Shimol et al. (2006) were the first to introduce an OFDMA frame packing algorithm. They presented two heuristics, with and without QoS constraints, which were evaluated by extensive simulation experiments using the parameters of real systems. They placed the data bursts row by row with a non-increasing size order. Ohseki et al. (2007) proposed a burst construction and frame packing method in the DL subframe. Their objective was not only to decrease the control data ratio within the rectangles, but also the control data that must be transmitted at the head of every frame, to reach higher throughput. They defined deadlines for each connection using the QoS parameters and ordered them according to the remaining times to deadlines.

As the number of bursts to be packed in a frame increases, so does the running time of the packing solution. Similarly, when a single terminal connection is mapped into multiple rectangular areas as in a study by Bacioccola et al. (2007), additional DL-MAP overhead causes inefficient usage of network resources. This particular case may be effective when the connection really requires different reliable channels using different modulation and coding schemes for different connections, despite the additional cost incurred. In the burst compaction case where multiple terminals are packed into a single burst, the unique connection identifier (CID) helps differentiate the terminals. The DL-MAP overhead is reduced, but may lead to decoding delay at the terminal side.

Necker et al. (2008) proposed a genetic algorithm for the DL frame packing problem. Simulating the network traffic with three different QoS classes, they applied the Next-Fit-Decreasing-Height (NFDH) strip packing approach to maximize DL capacity utilization. In strip packing, the objective is to pack all items within the minimum height. They based their genome modeling on the variable width and height dimension sizes of the DL bursts, since there are many possible height and width combinations corresponding to a burst area.

The shapes of the rectangles may change according to the defined objective of the problem. For example, minimum power consumption objective forces the allocated rectangles to smaller widths, hence smaller processing times on mobile stations, as in

So-In et al. (2009b). The authors introduced an algorithm for the DL frame packing. They approached the BS resource allocation, or MAC resource scheduling, in two steps. First, the scheduler sorts each user terminal in a descending order based on their demands for satisfying the QoS throughput guarantee. Then, the bursts are packed from right to left and from bottom to top in the DL subframe. The authors chose the burst shape that is smallest in width to allow the receiving MS to shut down its electronic circuit for the remainder of the DL subframe, thus minimizing energy consumption. They enhanced their algorithm in a further study to reduce complexity by fixing width or height of the bursts, still ensuring the required QoS parameters (So-In et al., 2009c). In a similar way, the maximum bandwidth usage refers to a full height utilization of a frame, which corresponds to the utilization of the complete frequency range of the BS.

Israeli et al. (2008) approached the problem in a rather different way, although they name it as a sequential rectangle placement problem. They aimed to pack a queue of MS data packets, and investigated the complexity of subframe allocation as regards the ratio of data region areas to subframe area. They specified an order of data packets (jobs) using strict priorities. Hence for a user's rectangle to be placed in the DL subframe, they required all the higher-priority packets to be packed before it. They showed that their problem is NP-hard, but also that it could be approximated within a constant factor if every packet size was limited to some constant fraction of the frame area.

Wang et al. (2008) proposed a low complexity heuristic algorithm depending on weighted flexibility definitions for minimizing different type spaces in the downlink subframes, by adapting a quasi-human based heuristic from a previous study (Wu et al.,2002). They did not handle any type of priorities among the users, either.

The most relevant and important contribution regarding the problem considered in this thesis in the wireless telecommunications context is a recent study by Lodi et al. (2011). The authors introduced two highly efficient heuristics that were developed to handle the DL subframe allocation problem practically. The processing budget of the

base station's scheduling process was estimated not to exceed a 1 ms on a state-of-the-art PC (2.40 GHz, CORE 2 DUO E6600 Desktop, running under Linux). Different than our homogeneity assumption regarding the same data transmission capacity for each slot (as will be explained in the following chapter while defining the problem), they allowed different modulation and coding schemes (MCS) in the same DL subframe defined by two zone structures. In this manner, their approach in a way was able to handle channel quality feedbacks from the users. They also took the downlink and uplink map overhead trade-off into account arising from the allocation of a large number of users in one subframe. The first one of these DL subframe zones was called the distributed subcarrier permutation, and the other the adjacent subcarrier permutation zone. The former one corresponds to the homogeneous structure employed in our model, so each slot can carry the same amount of data to every MS. But for the second zone, each MS (user) can provide detailed information regarding the target transmission/receive rate per defined logical bands. Namely, in this zone the data packet size cannot be expressed as a number of slots. The locations of the slots in the zone matter, hence the rectangle sizes for a user in each logical band of the zone might be different.

The authors defined two different allocation models for the zones explained above and analyzed the problem complexities, considering their special cases by Hurkens et al. (2011). They proved strong NP-hardness for both models. The optimization algorithm for the Distributed Permutation Zone problem was based on a previous study by Cicconetti et al. (2010). Some of the algorithm procedures used the greedy approach for the subset-sum and dual subset-problems. Although these produced solutions with the same waste amounts, the associated packings were quite different from each other. As for the Adjacent Permutation Zone problem, the authors developed a two-step algorithm, differentiating the packing of high priority large sized data packets and the rest. They reformulated one of these steps as a generalized assignment problem, and adapted a heuristic from the knapsack literature (Martello & Toth, 1990).

Lodi et al. (2011) also conducted an extensive analysis of the algorithms with realistic parametrization and traffic load for IEEE 802.16, proving that they can still provide sufficiently good solutions even when configured for fast execution instead of optimality. They compared both algorithms with the most relevant ones in the literature (Wang et al. 2008; So-In et al. 2009b) and proved superior performance. They claimed that the proposed algorithms and the zone architecture are compatible with realistic QoS schemes, and could be applied in real-world BS MAC units. Finally it should be noted that, the average solution times for both algorithms were around 0.1 ms, much lower than their initial 1 ms target.

In view of the thoroughly-reviewed literature in this chapter, we can conclude that any kind of modeling and solution attempt to the problem considering the latest technology is of great value to the wireless communication domain, and will contribute significantly to the existing literature. Including new features of the Mobile WiMAX technology in the models for taking into account the most recent telecommunications industry requirements is essential. The success of the developed models and algorithms will also depend on the solution performances attained by sufficient experimentation with realistic problem instances.

CHAPTER THREE
PROBLEM DEFINITION AND MATHEMATICAL MODELS

The main focus of this study is the resource allocation problem, which is defined as a bin packing problem on a two-dimensional structure that is proposed by the IEEE 802.16 standard using OFDMA. The problem context and the basic terminology were previously presented in Section 2.3.

We extend the common features of the frame packing problem model to include more general characteristics of the wireless data transfer processes involved. Apart from building and presenting representative mathematical models, which are not presented explicitly in most of the related studies in the literature, we lay the foundations for some fuzzy measures and parametrization to mimic more realistic evaluation of actual network resources.

Our problem is an extended form of the multiple knapsack problem with identical capacities (Kellerer et al., 2005). Identical bins or knapsacks correspond to the frames in time division duplexing (TDD). In addition, we deal with extra constraints and seek different objectives other than profit maximization for being able to handle various decision making alternatives. The item or demand sizes are not fixed, which requires nonlinear terms to be incorporated into the model.

As it was stated in Chapter 1 and will be explained further in the following sections, by taking into account the time division structure of the resource allocation sequence and other features of the telecommunications standard, we extend the two-dimensional nature of the problem to three-dimensional by adding additional knapsack and partitioning constraints.

In the following sections, we present exact nonlinear integer programming models with different objectives. The fuzzy variants of these models that adapt most of the constraints with respect to multiple objectives are also presented in this chapter.

3.1 Sequential Rectangular Packing (SRP) Problem

We propose a resource allocation modeling framework for a sequential two-dimensional packing problem which may have direct applications in wireless telecommunications area pertaining to the IEEE 802.16 standard. All the models described in this chapter aim optimal usage of the physical layer defined by this standard, which characterizes the data packages sent from a base station to a fixed or mobile network service user station. The data transmitted for each user are modeled as rectangular blocks, dimensions of which correspond to time duration and frequencies used in the wireless data transfer process. Placement of these rectangular blocks in identical rectangle bins called frames, whose dimensions are identified by the unit transfer time (usually 5 ms.) and a fixed frequency bandwidth depending on the base station's technological specifications, is modeled as a two-dimensional packing problem.

As reviewed in the previous chapter, most studies in the area aim to maximize the packing performance of a single frame employing strip packing techniques. These models assume identical user demand levels per frame subject to quality of service (QoS) constraints such as minimum transmission rate or maximum allowable delay. Shapes of the rectangles depend on the objective of the problem, which can be minimizing power consumption or maximizing bandwidth usage or throughput.

In this study, a planning horizon is composed of a sequence of frames (as opposed to a single frame) for representing the dynamic nature of the problem. Allowing varying demand sizes for each user, the aim is to solve interdependent multiple packing problems integrated with service level constraints. We hereafter name this problem as Sequential Rectangular Packing (SRP) Problem.

We propose four mathematical models for the SRP problem, which differ according to the objectives and the constraints involved. Three of these are exact nonlinear integer models whereas the last one is a fuzzy optimization model. Before delving further into each model's specific details, we firstly identify our common assumptions as realistically as possible. Thus, we aim a thorough representation of

the up-to-date technology specifications that are valid and commonly used for all the models mentioned in this chapter. Next, we introduce the parameters of our problem such as the length of the frame sequence that corresponds to the planning or so-called scheduling horizon for base station resources, or QoS restrictions for each user. After we define the decision variables and their relations with each other, we outline our mathematical models.

3.1.1 Assumptions for the SRP Models

We henceforth label the subscriber stations (mobile or fixed) which require wireless data transfer from the base station as "users". Each user can be allocated at most one burst (rectangular area) in a frame. Hence, in our study, the issues of burst compaction or utilizing more than one burst in a frame for one user are not considered. All models allocate the base station resources in a continuous manner, within the solution process and among separate solutions. In particular, the solution of a frame sequence is used as input parameter for the following sequence's problem. Some parameters correspond to measures or demands of ongoing data transfers, allowing the introduction of new user demands and eliminating some users in previous sequences which have completed their transfers.

Regarding the service quality constraints, we consider the minimum data transfer rate requirements and maximum allowable delay period parameters. A two-dimensional unit area used is named as a "slot", consistent with the related literature. Without loss of generality, we assume an identical data transmission capacity for each slot, ignoring possible modulation and coding differences allowed by the wireless telecommunications standard. Each slot can carry the same amount (bytes) of data resulting in homogenous frames in terms of frequency subchannels. Namely, rectangles that have the same areas correspond to the same amount of data carriage capacities even when they are placed on different frequency channels.

As the number of bursts or number of user rectangles that are to be packed in a frame increases, so does the downlink map (DL-MAP) size of a frame. A dynamic

assignment of DL-MAP size, which depends on the number of rectangles packed, is not considered in the models. Alternatively, we reduce the width of the frames by a fixed size for all the problem instances. So for example, instead of a 17x30 sized frame which fits most of the problems in the literature, we use a narrower 12x30 frame to be able reserve the remaining 5x30 block for DL-MAP allocation.

Some of the SRP models aim to serve all user demands in the given sequence. For obtaining feasibility for these models, we assume that the capacity of the frames can cover total demand for the respective period. For other models aiming at profit maximization, we ignore the respective capacity constraints for feasibility, and allow for choosing the best or profitable candidate set of user demands to be packed optimally.

Without loss of generality, all parameters are assumed to be positive integers. We present our parameter notations and definitions below.

3.1.2 Indices and Parameters

Two indices are used for identifying users and frames. User index is $i \in I = \{1,...,n\}$, where n= number of users to be allocated through the frames; whereas we index the frames by $j \in J = \{1,...,m\}$, where m= number of frames in the sequence (planning horizon).

Other parameters concerning resource and service level requirements are:

d_i : Total amount (in slots) of requested data (new or remaining) for user i,

s_i : Minimum data transfer rate (slots/frame) for user i,

$\phi_i = \min\{ms_i, d_i\}$: Data amount (in slots) to be transferred / assigned throughout the frame sequence,

p_i : Profit gained by satisfying the data demand ϕ_i of user i,

λ_i : Maximum delay period (in frames) for user i in order not to cause any timeout error (data transfer interruption),

W : Frame width, H : Frame height, $A = WH$: Frame area (all frames identical in size),

$\alpha_i = \left\lceil \frac{\phi_i}{A} \right\rceil$: Minimum number of frames to which user i should be assigned,

θ_i : Latest frame to maintain or to begin the data transfer for user i ($\leq \lambda_i$ for ongoing transfers, equal to m+1 for users to whom transfers are yet to be scheduled).

It should be noted that, when a user i is selected for transfer and the demand d_i is not totally satisfied in the frame sequence of a problem instance, its demand will be updated as $d_i = d_i - \phi_i$ for the next frame sequence.

Recall that we assume all parameters d_i, s_i, ϕ_i, p_i, λ_i, W, H, α_i and $\theta_i \in Z+$ (positive integers), $\forall i \in I$. The length of the frame sequence, m, is one of the experimentation parameters listed in the next chapter. For use in the fuzzy extensions of the models, the parameters d_i and λ_i form the most appropriate choices for the application of the flexibility assumptions more naturally. Hence, the quality degrees might be evaluated according to the demand satisfaction levels and the allocation interval (or delays) values of all users, for such a purpose.

3.1.3 Decision Variables

We define the following binary and integer variables for our models. It should be noted that some of the SRP models do not use all of these variables:

$$u_i = \begin{cases} 1, if \text{ user } i \text{ is selected for data transfer in the frame sequence,} \\ 0, otherwise. \end{cases}, \forall i \in I.$$

$$z_{ij} = \begin{cases} 1, if \text{ there is a transfer for user } i \text{ in frame } j \text{ (assigned a rectangle),} \\ 0, otherwise. \end{cases}, \forall i \in I, \forall j \in J.$$

x_{ij} : x-coordinate of the left-bottom corner of the rectangle assigned to user i in frame j.

y_{ij} : y-coordinate of the left-bottom corner of the rectangle assigned to user i in frame j.

w_{ij} : Width of the rectangle allocated to user *i* in frame *j*.

h_{ij} : Height of the rectangle allocated to user *i* in frame *j*.

$a_{ij} = w_{ij}\, h_{ij}$: Area of the rectangle allocated to user *i* in frame *j*.

$r_{ij} = \begin{cases} \phi_i - a_{i1}, \text{for } j = 1, \\ r_{i(j-1)} - a_{ij}, \text{for } j > 1. \end{cases}, \forall i \in I$, total remaining demand for user *i* after frame *j*.

The artificial binary variables, which will be explained below, are:

σ_{ij}, γ_{pqjk}, and δ_{pqj}.

3.2 Sequential Rectangular Packing with Profit Maximization (SRP-I)

In the first SRP model, we aim to allocate the base station resources to the most profitable subset of users. The characterization of the profit depends on the context, in which the problem instance is defined and solved. Obviously, the users who are charged more than the others by the service provider are more likely to be favored. On the other hand, profit levels might also correspond to subscriber priorities, e.g. when allocating network resources in case of humanitarian aid situations where uninterrupted communication between government or civil agency members is crucial. As another example, the profit could be associated directly with the demand value as in a subset sum problem. Namely, the higher the data demand of a user as in video transmission requests, the higher its profit level.

In any of the above definitions of the profit parameter, it might be stated that there seems no difference between handling the problem on a single frame and through a sequence of frames. Actually, the selection of the most profitable subset is the same for both, and corresponds to a one-dimensional knapsack problem. However, when it comes to optimal or even feasible packing of the user rectangles in a frame, the number of users do matter for practical solutions. Distributing the users over a frame sequence using some additional constraints, the sizes of the single frame packing problems may reduce. This will present an opportunity for obtaining acceptable solutions in shorter times, despite the difficulty of the problem that considers the packing of all users. Therefore, instead of simple averaging of user demand that

corresponds to equal rectangle sizes in each frame, we allow different partition sizes of user demand in each frame, as long as all feasibility constraints are satisfied according to QoS parameters.

SRP-I represents a realistic modeling approach considering similar examples in the literature where an optimal selection or scheduling takes place besides the two-dimensional packing. Based on the definition of the profits assigned to each user, the most rewarding of the feasible packings are sought. In a way, we convert a modified multiple knapsack problem with identical capacities to a combination of a 1-D knapsack and a simultaneous partitioning-and-packing problem. When the profits are identical to demands (i.e., when $p_i = \phi_i$), the user selection process is analogous to a multiple subset-sum problem. Partitioning is restricted according to the demand sizes and QoS constraints, while packings only deal with area allocations.

For this model, the problems that correspond to the case $\sum_{i \in I} \phi_i \leq mA$ are not considered. In this case, the problem reduces to packing all n users in a single frame with updated ϕ_i values ϕ_i/m. According to the above, the SRP-I model is defined as follows:

$$\max Z_K = \sum_{i \in I} p_i u_i \tag{3.1}$$

s.t.

$$\sum_{j \in J} z_{ij} \geq u_i \alpha_i, \forall i \in I. \tag{3.2}$$

$$z_{ij} \leq u_i, \forall i \in I, \forall j \in J. \tag{3.3}$$

$$\sum_{j \in J} w_{ij} h_{ij} \geq u_i \phi_i, \forall i \in I. \tag{3.4}$$

$$x_{ij} + w_{ij} \leq W, y_{ij} + h_{ij} \leq H, \forall i \in I, \forall j \in J. \tag{3.5}$$

$$\left.\begin{aligned} z_{ij} - 1 \leq x_{ij} \leq (W-1) z_{ij}, \\ z_{ij} - 1 \leq y_{ij} \leq (H-1) z_{ij}, \\ z_{ij} \leq w_{ij} \leq W z_{ij}, \\ z_{ij} \leq h_{ij} \leq H z_{ij}, \end{aligned}\right\} \forall i \in I, \forall j \in J. \tag{3.6}$$

$$\left.\begin{aligned} &1-A\sigma_{ij} \le r_{ij} \le \phi_i(1-\sigma_{ij}) \\ &\sum_{k=j+1}^{j+\lambda_i} z_{ik} \ge z_{ij} - \sigma_{ij},\ j \le m-\lambda_i \end{aligned}\right\} \sigma_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J. \tag{3.7}$$

$$r_{im} \le \phi_i(1-u_i), \forall i \in I \tag{3.8}$$

$$\left.\begin{aligned} &z_{pj} + z_{qj} \le \delta_{pqj} + 1, \\ &z_{pj} + z_{qj} \ge 2\delta_{pqj}, \end{aligned}\right\} \delta_{pqj} \in \{0,1\}, \forall p,q \in I,\ p<q, \forall j \in J \tag{3.9}$$

$$\left.\begin{aligned} &x_{pj} + w_{pj} - x_{qj} \le (1-\gamma_{pqj1})W, \\ &x_{qj} + w_{qj} - x_{pj} \le (1-\gamma_{pqj2})W, \\ &\gamma_{pqj1} + \gamma_{pqj2} \le 2 - \delta_{pqj}, \end{aligned}\right\} \gamma_{pqj1,}\gamma_{pqj2} \in \{0,1\}, \forall p,q \in I,\ p<q, \forall j \in J. \tag{3.10}$$

$$\left.\begin{aligned} &y_{pj} + h_{pj} - y_{qj} \le (1-\gamma_{pqj3})H, \\ &y_{qj} + h_{qj} - y_{pj} \le (1-\gamma_{pqj4})H, \\ &\gamma_{pqj3} + \gamma_{pqj4} \le 2 - \delta_{pqj}, \end{aligned}\right\} \gamma_{pqj3}, \gamma_{pqj4} \in \{0,1\}, \forall p,q \in I,\ p<q, \forall j \in J. \tag{3.11}$$

$$\left.\begin{aligned} &\gamma_{pqj1} + \gamma_{pqj3} \le 2(1 - z_{pj} + \delta_{pqj}), \\ &\gamma_{pqj2} + \gamma_{pqj4} \le 2(1 - z_{qj} + \delta_{pqj}), \\ &\gamma_{pqj1} + \gamma_{pqj2} + \gamma_{pqj3} + \gamma_{pqj4} \ge \delta_{pqj}. \end{aligned}\right\} \forall p,q \in I,\ p<q, \forall j \in J. \tag{3.12}$$

$$m - \theta_i \le u_i(1 + m - \theta_i) - 1, \forall i \in I \tag{3.13}$$

$$\sum_{j=1}^{\theta_i} z_{ij} \ge u_i, \forall i \in I. \tag{3.14}$$

$$x_{ij},\ y_{ij},\ w_{ij},\ h_{ij},\ a_{ij} \ge 0,\ \forall i \in I,\ \forall j \in J. \tag{3.15}$$

The objective function in (3.1) is the maximization of the total profit gained from the users that are selected for data transfer. First two constraints (3.2) and (3.3) link the decision variables u_i and z_{ij}. The nonlinear constraint (3.4) enforces the satisfaction of the user demand through the sequence of frames, if it is selected. Obviously, this means that the rectangle sizes can vary even if the area values are the same, as long as the total demand is satisfied. Constraint (3.5) simply limits the boundaries of each user rectangle for fully fitting in a frame, while constraint (3.6) defines the bounds of variables x_{ij}, y_{ij}, w_{ij}, h_{ij}, a_{ij} in relation with z_{ij}. The maximum delay constraint (3.7) both bounds the values of the variable r_{ij} and forces the

allocation of a user rectangle in a frame, if the total demand is not satisfied according to the QoS parameter λ_i. The binary variable $\sigma_{ij} = 0$ if $r_{ij} > 0$ (nonzero remaining demand), and 1 otherwise. Thus, constraint (3.8) enforces that there is no remaining demand after the last frame for the selected users.

Constraints (3.9)-(3.12) deal with feasible placements of user rectangles, preventing overlaps on the two-dimensional frame area. For determining the users allocated in the same frame, the variable δ_{pqj} is used in (3.9)-(3.12). $\delta_{pqj} = 1$ if both user p and q are allocated in frame j, and 0 otherwise. Constraint (3.10) manages the horizontal positions of the user rectangles. The variable γ_{pqj1} assumes the value of 1 if the rectangle of user p is situated on the left of the one of user q in frame j, and γ_{pqj2} is similarly defined for the rectangle of user q. Constraint (3.11) is the vertical positioning version of (3.10); and in a similar fashion, γ_{pqj3} equals 1 if the rectangle of user p lies below the one of user q in frame j, and γ_{pqj4} is defined vice versa. Constraint (3.12) links all the related binary variables used in (3.9), (3.10) and (3.11) for avoiding unnecessary overlap checks.

Lastly, constraint (3.13) enforces the selection of a user if his data transfer still continues from previous problem instances/sequences by using the θ_i parameter values, while constraint (3.14) restricts the latest beginning frame of these ongoing transfers. The nonnegativity of the integer decision variables is imposed by the constraint (3.15).

As presented above, SRP-I extends the rectangular allocation aspect of the frame packing problem defined in the literature, using the QoS constraints to a 3D structure by considering the sequential time frame element as an additional dimension via the binary decision variables and partitioning constraints defined.

3.2.1 Complexity Results for SRP-I

We next show that the special case of SRP-I where all ϕ_i are integer multiples of W (or where all ϕ_i are integer multiples of H) is NP-hard in the ordinary sense, and

prove through transformation from the Binary Knapsack Problem, which is known to be weakly NP-hard (Garey & Johnson, 1979). The following theorem states this result formally:

Theorem 3.1. *The SRP-I problem with either;*

- $\phi_i = \Omega_i W$, $\forall i$, *where* Ω_i *is a nonnegative integer multiplier, or*
- $\phi_i = \Omega_i H$, $\forall i$, *where* Ω_i *is a nonnegative integer multiplier*

is NP-hard in the ordinary sense.

Proof. Note that, for any of the above special cases, once the user subset to be packed is selected, an efficient strip packing solution without any overallocation is readily available. Namely, as the demand of each user is an exact multiple of the width/length of the frame, an optimal solution will assign a rectangular area to each user exactly equal to its demand. Therefore, our concern is to find whether there exists a feasible subset of users F such that $\sum_{i \in F} p_i \geq P$, where P is a positive integer.

To prove the complexity result, we reduce the Knapsack Problem to this special case of SRP-I. In an instance of a Knapsack Problem, we are given a set $\{1,...,k\}$ and with each one of its elements i is associated two positive integers q_i and v_i, reflecting the size and the value of i, respectively. Moreover, let Q and V be two positive integers. We are asked to find a subset $U \subseteq \{1,...,k\}$ such that $\sum_{i \in U} v_i \geq V$ and $\sum_{i \in U} q_i \leq Q$.

Given this instance of Knapsack Problem, we construct the following instance of SRP-I:

- $n = k$ users
- $\phi_i = q_i$ for $i = 1,..., k$
- $p_i = v_i$ for $i = 1,..., k$
- $A = Q / m$ for each of the m frames
- $P = V$

Note that, solving this instance will solve the knapsack instance as well. The Knapsack Problem is NP-hard in the ordinary sense, so are the given special cases of SRP-I. □

Corollary 3.1. *The SRP-I problem is NP-hard in the ordinary sense.*

Proof. Without the special demand structure defined by the special cases in Theorem 3.1, an optimal solution without any overallocation is not trivial for a selected subset of users. As the general version of the problem is at least as hard as its special cases, the proof follows. □

3.2.2 Lower and Upper Bounds for SRP-I

In this section, we propose a lower bound and an upper bound for the SRP-I problem. In order to establish the proposed upper bound, we first pose the following theorem and proof.

Theorem 3.2. *Consider the following instance of the continuous knapsack problem (CKP):*

- $k = n$ *items*
- $q_i = \phi_i$ *for* $i = 1,..., k$
- $v_i = p_i$ *for* $i = 1,..., k$
- $Q = mA$.

The corresponding mathematical model for the CKP can be expressed as:

$$\max Z_{UB} = \sum_{i \in I} p_i u_i$$

$$s.t. \sum_{i \in I} \phi_i u_i \le mA,$$

$$where \ 0 \le u_i \le 1, \forall i \in I.$$

Solution of this instance will provide an upper bound for SRP-I.

Proof. An exact solution of CKP, having a capacity equal to the total area of all frames, provides a relaxation on SRP-I in the following manner: First, it relaxes the rectangular packing structure and assumes that the demand of the selected user subset can be packed using any tetris-like pattern. Second, it allows partial demand allocation for the last selected user. Hence, it provides an upper bound for SRP-I. □

Based on Theorem 3.2, an exact solution for the defined continuous knapsack problem, hence an upper bound for SRP-I, can be obtained in polynomial time by the following algorithm (Dantzig, 1957):

AlgorithmUB

S1. Index the items (corresponding to users in SRP-I) in nonincreasing order of their p_i/ϕ_i ratios,

S2. Find the first item (in order) that does not fully fit into the knapsack. Mathematically, find:

$$s = \min\left\{ i : \sum_{j=1}^{i} \phi_i > mA \right\}.$$

S3. The optimal solution is:

$$u_i = 1 \text{ for } i = 1, \ldots, s-1,$$

$$u_i = 0 \text{ for } i = s+1, \ldots, n,$$

$$u_s = \frac{mA - \sum_{i=1}^{s-1} \phi_i}{\phi_s},$$

and the upper bound value is computed as:

$$Z_{UB} = \sum_{i=1}^{s} p_i u_i \ .$$

In order to obtain a simple and tight lower bound for the SRP-I problem, the following algorithm is used.

AlgorithmLB

S0. Consider a W by H single frame having area A.

Set $\phi_i \leftarrow \phi_i / m$, $\forall i \in I$.

$D_{\min} = \min\{W, H\}$ and $D_{\max} = \max\{W, H\}$.

Compute the number of unit strips of length D_{min} required for packing each demand. A unit strip of length D_{min} is defined as a rectangle having one

dimension as D_{min} and the other as one. The number of unit strips necessary for user i is:

$$N_i = \left\lceil \frac{\phi_i}{D_{\min}} \right\rceil, \forall i.$$

S1. Index the users in nonincreasing order of their p_i/ϕ_i ratios.

S2. Find the first item (in order) that does not fully fit into the frame.

Mathematically, find:

$$s = \min\left\{ i : \sum_{j=1}^{i} N_i > D_{\max} \right\}.$$

S3. The lower bound solution is:

$$u_i = 1 \text{ for } i = 1, \ldots, s-1,$$

$$u_i = 0 \text{ for } i = s, \ldots, n,$$

and the lower bound value is computed as:

$$Z_{LB} = \sum_{i=1}^{s-1} p_i u_i .$$

We incorporate the above upper and lower bounding procedures developed in this section, namely *AlgorithmLB* and *AlgorithmUB*, into the exact solution scheme of SRP-I, as will be explained in Chapter 4.

In the next sections, the two SRP models that cover the case $\sum_{i \in I} \phi_i \le mA$, that is, the sum of all user demands being smaller than the total available area, are described. They differ from SRP-I mainly by their objective functions, and they might be used firstly as individual problems when the sum of all user demands can be satisfied, as described in the following sections. Also, they may be defined as subproblems that incorporate additional criteria over the SRP-I profit maximization, to form a multi-criteria perspective.

3.3 Sequential Rectangular Packing with Minimum Partitions (SRP-II)

In this model, no profit is defined for satisfying the data demand of a user. Therefore, we either deal with the case where all users are considered for packing with all their ϕ_i fitting in the sequence of frames, or another objective is sought for packing an already selected subset of users. SRP-II model is developed for finding frame allocations that are as little fragmented as possible. Namely, the number of user rectangles or partitions placed in all frames is minimized.

As an example, let us consider a mobile station's downlink demand of 100 slots. We can partition this demand as 60 and 40 slots over two frames, and place according to other constraints, say with rectangle dimensions 4x15 and 5x8, respectively. On the other hand, let us suppose that there is the alternative of assigning the same demand to a square with sizes 10x10. Then, assuming all remaining feasibility conditions are satisfied, the second alternative with fewer partitions (one rectangle instead of two) should be selected according to the objective of SRP-II.

Although we have already stated our assumption that ignores the DL-MAP overhead, the objective of minimizing the number of rectangles contributes a lot in this direction for practical applications and might be employed when the number of users is high. Hence, by calculating the actual DL-MAP sizes for our packing solutions, we are able to measure solution performances better and solve the same packing problems with more realistic updated frame sizes.

As mentioned above, since there is not a selection of a user subset with respect to profit, the decision variable u_i is not included in this model. Thus, the related constraints are modified or omitted accordingly. The SRP-II model formulation is below:

$$\min Z_P = \sum_{i \in I} \sum_{j \in J} z_{ij} \tag{3.16}$$

s.t.

$$\sum_{i\in I}\phi_i \le mA. \tag{3.17}$$

$$\sum_{j\in J} z_{ij} \ge \alpha_i, \forall i \in I. \tag{3.18}$$

$$\sum_{j\in J} w_{ij}h_{ij} \ge \phi_i, \forall i \in I. \tag{3.19}$$

$$x_{ij} + w_{ij} \le W, y_{ij} + h_{ij} \le H, \forall i \in I, \forall j \in J. \tag{3.5}$$

$$\left.\begin{aligned} &z_{ij} - 1 \le x_{ij} \le (W-1)z_{ij}, \\ &z_{ij} - 1 \le y_{ij} \le (H-1)z_{ij}, \\ &z_{ij} \le w_{ij} \le Wz_{ij}, \\ &z_{ij} \le h_{ij} \le Hz_{ij}, \end{aligned}\right\} \forall i \in I, \forall j \in J. \tag{3.6}$$

$$\left.\begin{aligned} &1 - A\sigma_{ij} \le r_{ij} \le \phi_i(1-\sigma_{ij}) \\ &\sum_{k=j+1}^{j+\lambda_i} z_{ik} \ge z_{ij} - \sigma_{ij}, j \le m - \lambda_i \end{aligned}\right\} \sigma_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J. \tag{3.7}$$

$$r_{im} \le 0, \forall i \in I \tag{3.20}$$

$$\left.\begin{aligned} &z_{pj} + z_{qj} \le \delta_{pqj} + 1, \\ &z_{pj} + z_{qj} \ge 2\delta_{pqj}, \end{aligned}\right\} \delta_{pqj} \in \{0,1\}, \forall p,q \in I, p < q, \forall j \in J \tag{3.9}$$

$$\left.\begin{aligned} &x_{pj} + w_{pj} - x_{qj} \le (1-\gamma_{pqj1})W, \\ &x_{qj} + w_{qj} - x_{pj} \le (1-\gamma_{pqj2})W, \\ &\gamma_{pqj1} + \gamma_{pqj2} \le 2 - \delta_{pqj}, \end{aligned}\right\} \gamma_{pqj1,}\gamma_{pqj2} \in \{0,1\}, \forall p,q \in I, p < q, \forall j \in J. \tag{3.10}$$

$$\left.\begin{aligned} &y_{pj} + h_{pj} - y_{qj} \le (1-\gamma_{pqj3})H, \\ &y_{qj} + h_{qj} - y_{pj} \le (1-\gamma_{pqj4})H, \\ &\gamma_{pqj3} + \gamma_{pqj4} \le 2 - \delta_{pqj}, \end{aligned}\right\} \gamma_{pqj3}, \gamma_{pqj4} \in \{0,1\}, \forall p,q \in I, p < q, \forall j \in J. \tag{3.11}$$

$$\left.\begin{aligned} &\gamma_{pqj1} + \gamma_{pqj3} \le 2(1 - z_{pj} + \delta_{pqj}), \\ &\gamma_{pqj2} + \gamma_{pqj4} \le 2(1 - z_{qj} + \delta_{pqj}), \\ &\gamma_{pqj1} + \gamma_{pqj2} + \gamma_{pqj3} + \gamma_{pqj4} \ge \delta_{pqj}. \end{aligned}\right\} \forall p,q \in I, p < q, \forall j \in J. \tag{3.12}$$

$$\sum_{j=1}^{\theta_i} z_{ij} \ge 1, \forall i \in I. \tag{3.21}$$

$$x_{ij}, y_{ij}, w_{ij}, h_{ij}, a_{ij} \ge 0, \forall i \in I, \forall j \in J. \tag{3.15}$$

Constraint (3.16) is the minimization objective for the number of total rectangles placed in all the frames. Constraint (3.17) provides the trivial check for the total user demand, so that all user areas can fit in available area at least initially. Constraints (3.18) and (3.19) are modified forms of the constraints (3.2) and (3.4) of SRP-I, excluding the decision variable u_i. Similarly, constraint (3.20) is the modified version of constraint (3.8); enforcing that there is no remaining demand after the last frame for all users.

Since all the users must be placed in this model, by utilizing constraint (3.21) instead of its counterpart (3.14) in SRP-I, the latest beginning frames of all ongoing user transfers are bounded. It should be noted that this constraint becomes redundant for users with new transfers ($\theta_i = m+1$), because constraint (3.18) already handles more. The rest of the constraints are similar to those in the SRP-I model, as well as the nonnegativity of the decision variables except r_{ij}.

Next, we show that the special case of SRP-II where all users are placed in every frame in the sequence is strongly NP-hard:

Theorem 3.3. *SRP-II problem is strongly NP-hard*.

Proof. The area packing problem APP described in Hurkens et al. (2011) is a special case of the SRP-II problem that is defined on a single frame, the proof follows. □

3.4 Sequential Rectangular Packing with Minimum Overallocation (SRP-III)

The objective of the third model, namely SRP-III, is the minimization of overallocation or wasted space, which occurs due to the discrete nature of the problem. The same user demand or area can be satisfied by different sizes of rectangles, and since all parameters and variables are integers, some solutions might include rectangle placements with excess (wasted) spaces within.

For example, let a user *S* has a demand of 51 slots. Both a rectangle placement with sizes 3x17 and one with 2x26 satisfy her demand. All other constraints fulfilled, SRP-III model simply prefers the former solution over the latter as no overallocation occurs. In the latter case, there is an overallocation of 1 slot, computed as: 2x26 – 51=1.

Similar to the main assumption stated in SRP-II, all users can be considered for packing and there is no decision regarding the selection of users to be packed. Hence, the only difference of SRP-III from SRP-II lies in the objective function, which aims the most efficient utilization of the downlink frame areas. All the constraints being identical with the SRP-II model, the SRP-III model is defined as follows:

$$\min Z_O = -\sum_{i \in I} r_{im} \tag{3.22}$$

s.t.

$$\sum_{i \in I} \phi_i \le mA. \tag{3.17}$$

$$\sum_{j \in J} z_{ij} \ge \alpha_i, \forall i \in I. \tag{3.18}$$

$$\sum_{j \in J} w_{ij} h_{ij} \ge \phi_i, \forall i \in I. \tag{3.19}$$

$$x_{ij} + w_{ij} \le W, y_{ij} + h_{ij} \le H, \forall i \in I, \forall j \in J. \tag{3.5}$$

$$\left.\begin{aligned} z_{ij} - 1 \le x_{ij} &\le (W-1) z_{ij}, \\ z_{ij} - 1 \le y_{ij} &\le (H-1) z_{ij}, \\ z_{ij} \le w_{ij} &\le W z_{ij}, \\ z_{ij} \le h_{ij} &\le H z_{ij}, \end{aligned}\right\} \forall i \in I, \forall j \in J. \tag{3.6}$$

$$\left.\begin{aligned} 1 - A\sigma_{ij} \le r_{ij} &\le \phi_i (1 - \sigma_{ij}) \\ \sum_{k=j+1}^{j+\lambda_i} z_{ik} &\ge z_{ij} - \sigma_{ij}, \; j \le m - \lambda_i \end{aligned}\right\} \sigma_{ij} \in \{0,1\}, \forall i \in I, \forall j \in J. \tag{3.7}$$

$$r_{im} \le 0, \forall i \in I \tag{3.20}$$

$$\left.\begin{aligned} z_{pj} + z_{qj} &\le \delta_{pqj} + 1, \\ z_{pj} + z_{qj} &\ge 2\delta_{pqj}, \end{aligned}\right\} \delta_{pqj} \in \{0,1\}, \forall p, q \in I, \; p < q, \forall j \in J \tag{3.9}$$

$$\left.\begin{aligned} &x_{pj} + w_{pj} - x_{qj} \le \left(1-\gamma_{pqj1}\right)W, \\ &x_{qj} + w_{qj} - x_{pj} \le \left(1-\gamma_{pqj2}\right)W, \\ &\gamma_{pqj1} + \gamma_{pqj2} \le 2-\delta_{pqj}, \end{aligned}\right\} \gamma_{pqj1,}\gamma_{pqj2} \in \{0,1\}, \forall p,q \in I,\ p<q, \forall j \in J. \tag{3.10}$$

$$\left.\begin{aligned} &y_{pj} + h_{pj} - y_{qj} \le \left(1-\gamma_{pqj3}\right)H, \\ &y_{qj} + h_{qj} - y_{pj} \le \left(1-\gamma_{pqj4}\right)H, \\ &\gamma_{pqj3} + \gamma_{pqj4} \le 2-\delta_{pqj}, \end{aligned}\right\} \gamma_{pqj3},\gamma_{pqj4} \in \{0,1\}, \forall p,q \in I,\ p<q, \forall j \in J. \tag{3.11}$$

$$\left.\begin{aligned} &\gamma_{pqj1} + \gamma_{pqj3} \le 2\left(1-z_{pj}+\delta_{pqj}\right), \\ &\gamma_{pqj2} + \gamma_{pqj4} \le 2\left(1-z_{qj}+\delta_{pqj}\right), \\ &\gamma_{pqj1} + \gamma_{pqj2} + \gamma_{pqj3} + \gamma_{pqj4} \ge \delta_{pqj}. \end{aligned}\right\} \forall p,q \in I,\ p<q, \forall j \in J. \tag{3.12}$$

$$\sum_{j=1}^{\theta_i} z_{ij} \ge 1, \forall i \in I. \tag{3.21}$$

$$x_{ij},\ y_{ij},\ w_{ij},\ h_{ij},\ a_{ij} \ge 0,\ \forall i \in I,\ \forall j \in J. \tag{3.15}$$

The objective function in (3.22) is simply the negated sum of the values of the decision variables r_{ij} after the last frame, which is equivalent to the total overallocation. All constraints are identical with SRP-II, and presented here again for the sake of completeness.

CHAPTER FOUR
COMPUTATIONAL RESULTS

For validating and testing the models developed in this thesis, two different data sets have been generated for computational experimentation. The computational results pertaining to these data sets are presented in this chapter. In Section 4.1, we present the experiment design and results for the first data set, which has a single class of demand and random profits. Section 4.2 describes the data with two classes of demands and two classes of profits, and the corresponding computational results are supplied.

4.1 Data Set 1: Data with a Single Demand Class and Random Profits

The first data set was designed by considering 24 different combinations of three parameter levels. 10 instances were generated from each combination adding up to a total of 240 instances. Differentiating only the number of users, frames and frame area sizes, overall experiment design for Data Set 1 is summarized in Table 4.1.

Table 4.1 Parameter levels for Data Set 1

Parameter name	Levels
n (# of users)	10, 20, 40, and 80
m (# of frames)	2, 4, and 8
$A = W$xH (frame area in slots)	360 (12x30) and 1200 (20x60)
d_i	~ UD (16, 128)
p_i	~ UD (48, 240)
λ_i	~ UD (1, m)
θ_i	m+1 (all transfers new)

As summarized in Table 4.1, the demand and profit values for this data set are uniformly generated independently from each other. All users are assumed to belong to a single demand class, and the profits are generated randomly for all users. Subset sum logic might be employed using the same problem inputs whenever necessary, by simply putting $p_i = d_i$.

Both problem generation and solution processes are automated by a console application developed in C# programming language on MS Visual Studio 2010 IDE (integrated development environment). The three models presented in Chapter 3 are tested and validated on GAMS 23.9 using BARON 11.5 (Tawarmalani and Sahinidis 2005) and SCIP 2.1.2 (Achterberg et al. 2008; Achterberg 2009) as different MINLP solvers. Moreover, the application design easily allows integration of different solvers and algorithms on top of the existing ones developed for this thesis. The visual packing results are obtained by developing Excel macro codes in Visual Basic language.

The PC configuration used for experimentation was above average with 2.30 GHz CORE i7 3610QM processing power, 8 GB RAM memory and run on 64-bit Windows 7 operating system. Example GAMS codes for SRP-I, SRP-II and SRP-III are provided in Appendix A.1, A.2 and A.3, respectively.

The distributions of the problem instances according to user demand density, which is the ratio of total demand to available area, and the overall model sizes in terms of decision variable and constraint counts are given in Table 4.2.

Of all the generated instances, 135 instances that correspond to the case where $\sum_{i \in I} \phi_i \leq mA$ are used only for testing and comparing the solutions of models SRP-II and SRP-III, which do not involve user profits. The rest (105 problem instances) have demand densities higher than 100%, and these are considered for testing the SRP-I model. A 20-minute runtime limit is employed for all solutions.

Table 4.2 Demand densities and model sizes for Data Set 1

n	*A (W*x*H)*	*m*	Demand density (%)	Average # of decision variables	Average # of constraints
10	12x30	2	109	617	1325
		4	48	1221	2588
		8	26	2441	5147
	20x60	2	28	611	1306
		4	14	1221	2585
		8	8	2441	5146
20	12x30	2	201	2241	4870
		4	99	4447	9600
		8	51	8881	19093
	20x60	2	61	2221	4810
		4	30	4441	9569
		8	15	8881	19094
40	12x30	2	392	8481	18540
		4	207	16921	36939
		8	101	33785	73605
	20x60	2	117	8481	18541
		4	59	16881	36741
		8	30	33761	73377
80	12x30	2	759	32961	72282
		4	394	65841	144274
		8	203	131601	288276
	20x60	2	236	32961	72281
		4	117	65841	144285
		8	61	131521	287570

An initial set of runs for the SRP-I problem were completed without employing any of the explicit bounds proposed in Section 3.2, and the results are presented in Table 4.3. In this case, no problem could be solved to optimality. Furthermore, as it can be observed in Table 4.3, feasible solutions of only 25 of the instances could be

found by either one of the solvers within the given runtime limit of 20 minutes as shown by the *# of incumbent solutions* column. In particular, SCIP solver reached feasible solutions for 23 instances in total, 21 of which has better objective values than the ones obtained by BARON, which obtained 9 feasible solutions in 105 instances.

Table 4.3 Results for incumbent solutions of SRP-I without incorporating any bounds (Data Set 1)

n	*A (W*x*H)*	*m*	# of instances	# of incumbent solutions	Average gap (%)	Average # of rectangles	Average # of users packed
10	12x30	2	6	6	7	15	8
20	12x30	2	10	4	51	12	8
		4	3	2	89	7	4
40	12x30	2	10	0	-	-	-
		4	10	1	96	2	2
		8	6	0	-	-	-
	20x60	2	10	4	68	21	13
80	12x30	2	10	0	-	-	-
		4	10	0	-	-	-
		8	10	0	-	-	-
	20x60	2	10	1	75	19	12
		4	10	7	93	7	5

The gap averaged over 10 instances is used as a performance measure, where the gap percentage is defined as the difference between the best upper bound solution and the best/incumbent lower bound solution obtained during the runtime limit (if not optimal) by the solver, as a percentage of the upper bound solution. Mathematically;

$$\% \text{ Gap} = \frac{BestUB_{Solver} - \text{Incumbent Solution}}{BestUB_{Solver}}.100. \qquad (4.1)$$

As it can be seen from Table 4.3, except the smallest problem instances with 10 users, the average gaps are quite high. Average number of rectangles and users packed are also shown in the table. The objective value of the instance with the smallest gap value of 4.3%, which was proven to be optimum by an unlimited-time run, belongs to a 10-user instance solution. The corresponding packing for this solution is illustrated in Figure 4.1. Such a visual representation of the solution is generated automatically in MS Excel as output for each problem instance by the developed codes. Some example problem input and output files are provided in Appendix B. Figure 4.1 is directly taken from such an output file.

The problem input corresponding to Figure 4.1 has 10 users, with a total demand of 735 slots and potential profit of 1550, which are to be packed over 2 frames, each with size 12x30. 621 decision variables and 1336 constraints were evaluated by the SCIP solver to reach the objective value of 1483 in 13 seconds, which couldn't be proven to be optimal in 20 minutes.

Figure 4.1 SRP-I solution for a problem with 10 users packed over 2 frames

Note that, 9 of the 10 users are selected for packing in 17 partitions (rectangles) over 2 frames, with a total overallocation of only 10 slots. The average area of the rectangles placed is 39 slots, compared to the average demand 74 per user.

In order to improve the solution performance for the SRP-I problem, the lower and upper bounds defined in Section 3.2.2 were incorporated into the model, and a second set of results were obtained for the same 105 instances by using only the SCIP solver and a 10-minute runtime limit. The results are presented in Table 4.4. The table lists the instances and corresponding experimental parameters, along with the performance measure values. The codes for *AlgorithmLB* and *AlgorithmUB* can be found in Appendix C.1 and C.2, respectively.

Table 4.4 Results for incumbent solutions of SRP-I employing lower and upper bounds (Data Set 1)

n	*A (W*x*H)*	*m*	# of instances	Average initial gap (%)	Average final gap (%)
10	12x30	2	6	7.8	4.6
20	12x30	2	10	13.8	13.1
		4	3	10.7	10.6
40	12x30	2	10	14.3	14.3
		4	10	19.4	19.4
		8	6	24.2	24.0
	20x60	2	10	12.2	12.2
80	12x30	2	10	17.3	17.3
		4	10	26.5	26.5
		8	10	38.9	38.9
	20x60	2	10	18.9	18.9
		4	10	24.7	24.7

As the lower bound obtained by *AlgorithmLB*, which was presented in Section 3.2.2, is employed in this case, all instances have an initial feasible solution.

Therefore, we report the corresponding initial gap percentage between the lower bound and the initial upper bound found using *AlgorithmUB* (see in Section 3.2.2), averaged over 10 instances. The initial gap percentage is computed as:

$$\% \text{ Initial Gap} = \frac{UB_{AlgorithmUB} - \text{LB}_{AlgorithmLB}}{UB_{AlgorithmUB}}.100$$

The average final gap percentage values attained by the solver are also shown in Table 4.4, which are computed using equation (4.1).

Although only one optimal solution was obtained in the 10-minute running limit, the average gap values listed in Table 4.4 are far superior compared to those presented in Table 4.3. The optimal solution corresponds to the same problem instance used in Figure 4.1, this time obtained in just six seconds as a result of the employment of bounds, and is shown in Figure 4.2.

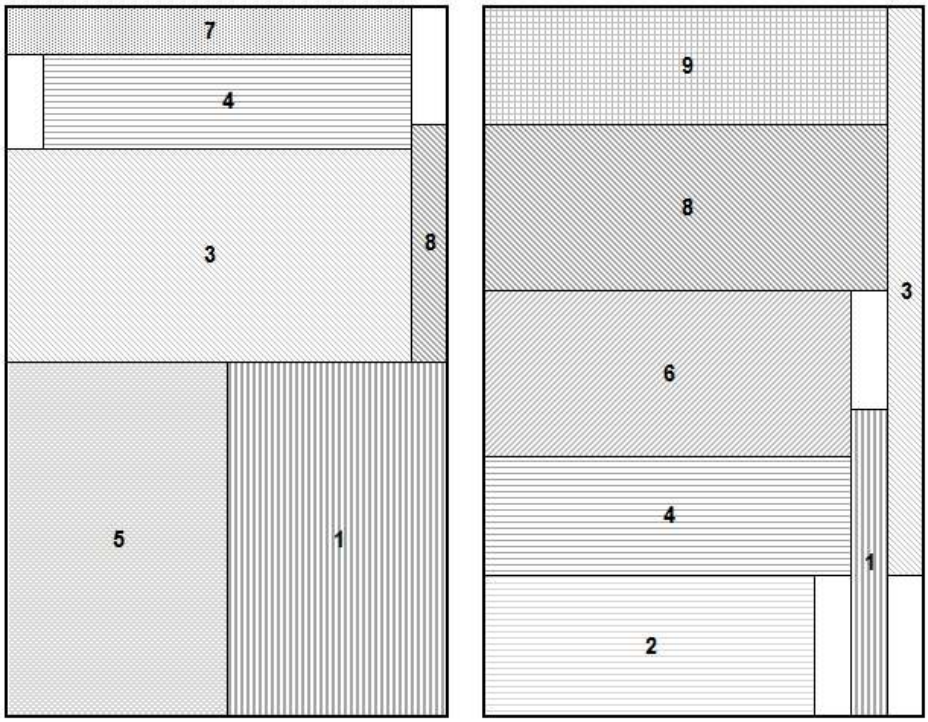


Figure 4.2 SRP-I solution for the same problem employing bounds

It should also be noted that the optimal packing of the problem shown in Figure 4.2 has a fewer number of rectangles, 13 as opposed to 17 of Figure 4.1, but with a much higher overallocation of 40 slots in total. Consequently, the average area (53 slots) of the rectangles placed is larger.

The difference between the two set of SRP-I runs suggests that, implementing bounds such as the ones described in Section 3.2.2, and probably more sophisticated ones, can improve the solution performance further either for exact or approximate solutions.

As for the SRP-II and SRP-III models, no feasible solution could be obtained for problem instances having 40 or 80 users within the time limit. The results for other problem instances with 10 and 20 users are presented in Table 4.5. All problem instances with 10 users are solved to optimality by the two models using either one of the solvers. The shortest solution times performed by any of the two solvers are taken into account while averaging.

Table 4.5 Comparison of SRP-II and SRP-III solutions for n=10, 20 (Data Set 1)

n	A (WxH)	m	# of instances	# of optimal solutions		Average # of rectangles		Average best solution time (s)		Worst solution time (s)	
				SRP-II	SRP-III	SRP-II	SRP-III	SRP-II	SRP-III	SRP-II	SRP-III
10	12x30	2	4	4	4	12	18	278.7	19.4	736.8	245.7
		4	10	10	10	13	29	15.8	6.5	833.9	37
		8	10	10	10	13	26	31	50.8	94.6	179.5
	20x60	2	10	10	10	12	19	1.1	1.6	47.1	55.6
		4	10	10	10	12	37	1.8	1.1	552.4	48.5
		8	10	10	10	13	73	3.7	0.6	1051.2	65.1
20	12x30	2	0	-	-	-	-	-	-	-	-
		4	7	0	0	-	-	-	-	-	-
		8	10	2	6 (3)	26	64	484.8	624.1	658.3	1031.7
	20x60	2	10	10	8 (2)	25	35	139.5	278.1	770.2	1182.2
		4	10	9	10	25	66	195.6	187.4	399.9	848.8
		8	10	5 (5)	10	24	150	482.1	32.1	983.4	275.4

* Numbers of incumbent solutions reached within the 20-minute limit are given in parentheses.

The number of optimal solutions obtained by the solver out of the given number of instances are listed in the table as well as the average number of rectangles placed, the average best solution times and the maximum solution times (in seconds) for each parameter combination, by distinguishing model parameters such as frame size (A) and frame sequence length (m). It should be noted that the number of user rectangles allocated in SRP-II solutions are significantly lower than in the SRP-III solutions of the same instances, which justifies this study's motivation for using the SRP-II objective for minimizing the number of partitions. For example, when n=10, m=4, and A=12x30, the SRP-II solution allocates 13 rectangles in average, as opposed to an average 29 rectangles placed by the SRP-III solutions.

The difference in the solution times of the 10-user instances suggests that minimizing the number of rectangles is a more complex objective than minimizing overallocation. This might also be observed by looking at the number of optimal solutions obtained within the runtime limit for the 20-user problems. On the other hand, the difference in average solution times are significant as the number of users doubles, where the duration gets much higher with the increasing number of variables and constraints.

The variances of average solution times seem higher for the problems with 10 users and small frames (A=12x30) for both models. This is because of the higher demand densities for the 2-frame problems (Table 4.2), and therefore, tighter instances. On the other hand, for the instances with larger frame area (A=20x60), the SRP-II solutions take longer as the number of frames, m, increases. For the same setting, SRP-III solutions show the opposite trend and take shorter times. This result is expected, since as observed in Table 4.5, SRP-III assigns rectangles in each frame to almost every user to minimize overallocation with no partitioning concern, which allows finding solutions in shorter times when compared to tighter settings with fewer frames.

Two example packings generated by using SRP-II and SRP-III models are presented in Figures 4.3 and 4.4, respectively.

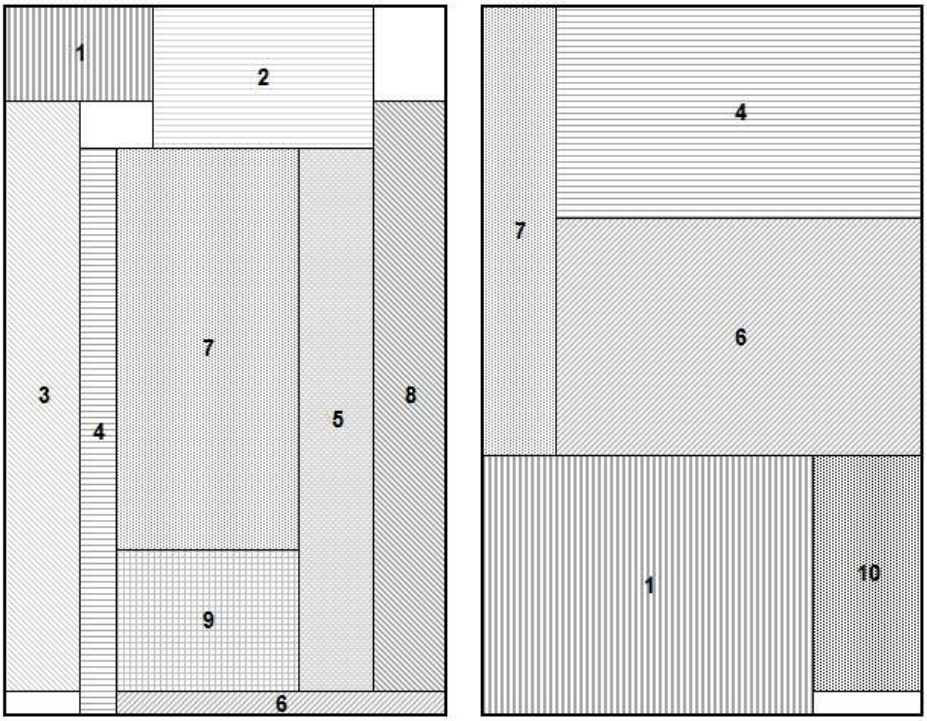


Figure 4.3 SRP-II solution for a problem with 10 users packed over 2 frames

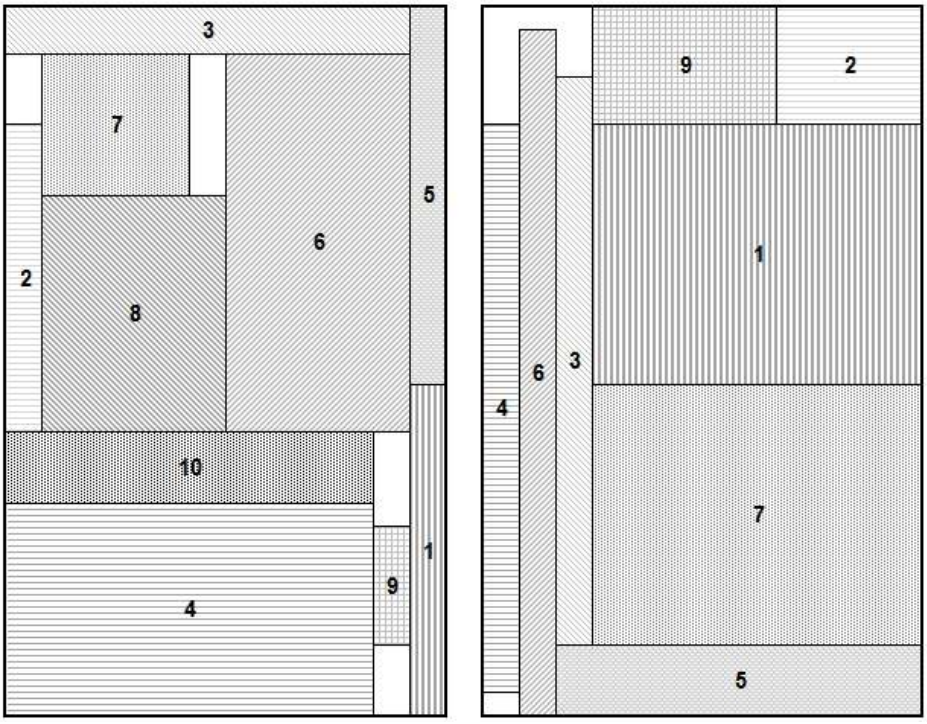


Figure 4.4 SRP-III solution for the same problem with 10 users over 2 frames

The same problem input is used for both models: 10 users, with a total demand of 694 slots, are required to be packed over 2 identical frames of size 12x30. A total of 611 decision variables and 1309 constraints are handled by the solvers. The SRP-II GAMS code for this instance is provided in Appendix A.2.

In Figure 4.3., in line with the objective of SRP-II, the total number of placed rectangles are 14 instead of 20, which would be the trivial value if the problem aimed packing all 10 users in every frame. As seen from the figure, placing only 5 user rectangles was sufficient for the second frame, and the demands of only 4 users are partitioned through both frames (users 1, 4, 6 and 7). The total overallocation of the solution is 9 slots.

In contrast, the total number of rectangles allocated in SRP-III solution is 18, while no overallocation occurred, as seen in Figure 4.4. All demand is partitioned through two frames except users 8 and 10. As expected, the average rectangle area value is larger in SRP-II solution (50 slots) than in SRP-III (39 slots), which is another hint allowing us to distinguish the differences in objectives.

As stated above, the objective of minimizing the number of partitions proves itself worthy in terms of the reduction of items to be packed. Specifically in the second frame of Figure 4.3, only 5 rectangles are packed instead of 10. Furthermore, it should be noted that the objective of minimizing overallocation might be better handled on a single frame for simplicity, as there seems no significant reduction in the number of rectangles involved (Table 4.5).

In terms of solution time, SRP-III solution for this specific problem instance was obtained in much shorter duration than for SRP-II, 19 seconds and 692 seconds, respectively.

4.2 Data Set 2: Data with Two Demand and Profit Classes

After examining the results of the runs with Data Set 1 for the exact nonlinear integer programming models developed in this thesis, it can be safely stated that the theoretical problem difficulties are reflected appropriately. For a more thorough analysis, in order to test other network scenario conditions that can simulate real life packet traffic load, the structures and parameters of similar problem instances from the literature are investigated in detail, and adapted as Data Set 2 for further computation. The experiment results for this data set are provided in this section.

For practical purposes, and for compensating our assumption that ignores the DL-MAP overhead, all problem instances with 80 users are omitted from Data Set 2. Examining the results of Section 4.1, it is obvious that the highly combinatorial nature of our problems make it nearly impossible to obtain solutions in acceptable times for large numbers of users involved (Tables 4.2, 4.3). All instances with large frame areas (A=20x60) are also omitted from the data set due to the fact that, as the demand densities are lower for these instances and the packings are distributed in a very sparse manner, the solutions become trivial and meaningless to evaluate the performance of the proposed sequential approach.

For being able to simulate real network traffic load within our experimentation boundaries, two different classes of packet traffic are considered. Explicitly, user demands are classified as *data* and *voice* traffic as in Lodi et al. (2011). Moreover, the problem instances in Data Set 2 are generated from two different distributions of these classes, namely 75% data versus 25% voice traffic, and 50% from each. Hereafter, the distribution of users with 75% data and 25% voice traffic class will be denoted as *TR1*, and the latter distribution with 50% of each class as *TR2*.

The main difference between these *TR1* and *TR2* are due to their traffic activity levels and packet sizes (Lodi et al., 2011). It is assumed that the MSs with data traffic are always active, i.e., the BS has always backlogged data waiting for transmission directed to them. Thus, to mimic this continuous flow of data, their maximum delay parameter values are assumed lower, to be exact, equal to 2 frames (10 ms) for all

computations. On the other hand, since voice applications have a discontinuous nature, which assumes lower probabilities for packets waiting for transmission from the BS side, the maximum delay values for users with voice traffic are chosen as discrete uniform in the interval [3, 8].

In addition to the classification of demand, as opposed to the independent random assignment of profit values regardless of the user demand levels in Data Set 1, the users with voice traffic are prioritized in Data Set 2. Namely, higher profit coefficients are used for voice traffic than the ones used for data traffic. The parameter settings used for Data Set 2 are listed in Table 4.6.

Using the settings in Table 4.6, 180 problem instances were generated in Data Set 2, where 82 instances that correspond to the case $\sum_{i \in I} \phi_i \leq mA$ are used for testing and comparing the solutions of SRP-II and SRP-III, and the rest (98 problem instances) with demand densities higher than 100% are considered for testing SRP-I. All instances are tested on the SCIP solver within a 10-minute running limit. The results of the new runs are analyzed in the same way presented in Section 4.1 and the same computer configuration is used for the runs.

The distributions of the problem instances according to user demand density, defined as the ratio of total demand to available area, and the overall model sizes in terms of the number of decision variables and constraints are presented in Table 4.7.

The main differences in Data Set 2 as compared to Data Set 1 are the introduction of user classes that differ by demand sizes, respective profit values, and maximum delay parameters. As presented in Table 4.7, the model sizes do not change, since they are only dependent on the values *n*, and *m*. However, the demand densities of the new problem instances are about 10% higher than the problems of Data Set 1.

Table 4.6 Parameter levels for Data Set 2

Parameter name	Levels
n (# of users)	10, 20, and 40
m (# of frames)	2, 4, and 8
$A = W$xH (frame area in slots)	360 (12x30)
User class distribution (data + voice)	75% + 25% and 50% + 50%
$d_{i,data}$ or $d_{i,voice}$	~ UD (12, 192) or UD (10, 80)
$p_{i,data}$ or $p_{i,voice}$	d_i [UD (1, 6)] or d_i [UD (6, 12)]
$\lambda_{i,data}$ or $\lambda_{i,voice}$	2 or ~ UD (3, 8)
θ_i	m+1 (all transfers new)

Table 4.7 Demand densities and model sizes for Data Set 2

n	m	Demand density (%)		Average # of decision variables	Average # of constraints
		75% data + 25% voice	50% data + 50% voice		
10	2	119	104	618	1321
	4	69	58	1221	2585
	8	27	23	2441	5155
20	2	237	203	2241	4861
	4	117	91	4452	9619
	8	63	57	8881	19118
40	2	482	425	8481	18521
	4	227	214	16921	36933
	8	126	98	33789	73679

As an initial experiment, the 98 instances are tested for the SRP-I problem without using any of the bounds proposed in Section 3.2.2. In this case, no problem could be solved to optimality by the SCIP solver within the 10-minute time limit, and only 19 feasible solutions could be obtained. Four of the feasible packings in the 10-user, 2-frame instances that belong to *TR1* and *TR2* distributions have about 2% gaps, one of which was proven to be optimal in 4 seconds by the SRP-I runs employing bounds. It should be also noted that, three of these feasible solutions are trivial packings that packed only one user, and even one rectangle. The results are presented in Table 4.8. Gap computations are identical to those in Section 4.1.

Table 4.8 Results for incumbent solutions of SRP-I without incorporating any bounds (Data Set 2)

n	Traffic distribution	*m*	# of instances	# of incumbent solutions	Average gap (%)	Average # of rectangles	Average # of users packed
10	*TR1*	2	8	8	12	14	8
	TR2	2	5	5*	6	14	8
20	*TR1*	2	10	1	48	6	3
		4	9	0	-	-	-
	TR2	2	10	2	58	8	6
		4	2	0	-	-	-
40	*TR1*	2	10	0	-	-	-
		4	10	0	-	-	-
		8	10	3	96	1	1
	TR2	2	10	0	-	-	-
		4	10	0	-	-	-
		8	4	0	-	-	-

* One of these solutions are later proven to be optimal.

Similar to the runs executed in Section 4.1, SRP-I solutions to the same instances are explored by incorporating the lower and upper bounds developed in Section 3.2.2. Since *AlgorithmLB* is now employed in the solution mechanism, all instances

presented in Table 4.8 have guaranteed initial feasible solutions. Moreover, for two of the 10-user, 2-frame instances, one of which belongs to *TR1* and the other to *TR2*, optimal packings are obtained within 10 and 4 seconds, respectively. The corresponding average initial and final gaps attained are reported in Table 4.9 in a similar fashion to Table 4.4.

Table 4.9 Results for incumbent solutions of SRP-I employing lower and upper bounds (Data Set 2)

n	Traffic distribution	*m*	# of instances	Average initial gap (%)	Average final gap (%)
10	*TR1*	2	8*	6.8	3.6
	TR2	2	5*	10.0	5.1
20	*TR1*	2	10	15.0	14.0
		4	9	13.6	13.5
	TR2	2	10	16.8	15.3
		4	2	15.4	15.0
40	*TR1*	2	10	14.9	14.9
		4	10	20.8	20.8
		8	10	23.3	23.3
	TR2	2	10	16.3	16.3
		4	10	23.3	23.3
		8	4	21.3	21.3

* Two of these instances are solved to optimality and not included in this table's calculations.

The gap values in Table 4.9 are very close to the ones summarized in Table 4.4. It seems that whatever the demand densities are, the dominant factor determining the solution time and quality is the problem size, which is defined by the number of users and frames involved. Therefore it can be stated that the SRP-I model is robust in terms of demand density parameter.

One of the two optimal solutions, which is obtained in 10 seconds by the SCIP solver, is shown in Figure 4.5. Nine of the 10 users are packed, with an overallocation of 5 slots, by filling up all the available area of the two frames. In this instance, there is only one user (user 1) with voice traffic demand (28 slots). However, having a relatively higher profit coefficient, it contributes to the 6% of the optimal total profit gained despite its relatively smaller transfer size (4% of the total demand packed). This is a direct result of the data generation scheme used for Data Set 2, which involves generating profit values dependent on demand classes. The SRP-I GAMS code for this instance is provided in Appendix A.1.

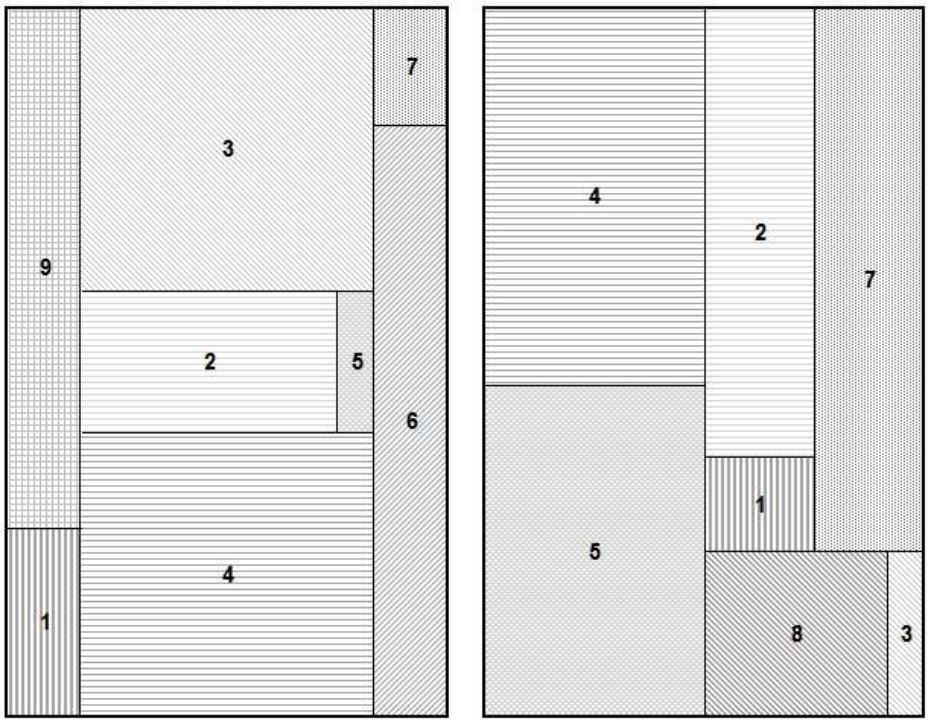


Figure 4.5 SRP-I solution for a problem with 10 users packed over 2 frames with traffic distribution *TR1* (75% of users with data + 25% voice traffic)

For the sake of completeness, SRP-II and SRP-III models for the remaining 82 instances were also executed, despite the lacking effect of different profit classes but only higher average demand sizes. In line with the results of Section 4.1, no feasible

solution could be obtained for problem instances having 40 users within the runtime limit.

The results for problem instances having 10 and 20 users are summarized in Table 4.10. Nearly all problem instances with 10 users are solved to optimality by the two models, regardless of their traffic distributions. As noted before, the determining factor affecting the solution performances of these models remains to be the user demand size.

Table 4.10 Comparison of SRP-II and SRP-III solutions for n=10, 20 (Data Set 2)

n	Traffic dist.	m	# of instances	# of optimal solutions		Average # of rectangles		Average solution time (s)		Worst solution time (s)	
				SRP-II	SRP-III	SRP-II	SRP-III	SRP-II	SRP-III	SRP-II	SRP-III
10	*TR1*	2	2	2	2	12	16	9.4	90.8	12.0	161.5
		4	10	10	10	15	38	54.1	23.7	120.5	226.3
		8	10	9 (1)	10	13	32	30.4	71.4	57.5	167.8
	TR2	2	5	5	5	11	18	15.6	21.3	35.8	51.9
		4	10	10	10	13	38	26.4	5.4	82.3	48.0
		8	10	10	10	12	31	41.0	67.9	113.0	150.9
20	*TR1*	2	0	-	-	-	-	-	-	-	-
		4	1	0	0	-	-	-	-	-	-
		8	10	0	2 (4)	-	74	-	544.9	-	572.1
	TR2	2	0	-	-	-	-	-	-	-	-
		4	8	1	4 (2)	20	62	294.0	365.0	294.0	545.8
		8	10	0	1 (5)	-	51	-	471.2	-	471.2

* Number of incumbent solutions reached within the 10-minute limit are given in parentheses.

Nevertheless, the two entries in Table 4.10 regarding the average solution times of 10-user, 4-frame instances for SRP-III model with both traffic distributions are worth mentioning. To be precise, excluding the worst, the average solution times of the remaining 9 instances become 1.2 and 0.7 seconds respectively for the *TR1* and *TR2* cases, instead of 23.7 and 5.4 seconds.

Two optimal packing examples from the mentioned setting (10 users, 4 frames) with *TR1* distribution are given in Figures 4.6 and 4.7 for comparison. It should be noted that the first instance (Figure 4.6) with 80% demand density was solved to

optimality within 226 seconds by packing 27 rectangles, while the latter (Figure 4.7) having 71% demand density was packed in 0.4 seconds in 40 rectangles. Although both optimal values are zero, it seems that SRP-III is much easier to solve when the demand density is lower. In fact, all instances of the same setting except the one shown in Figure 4.6, have demand densities of at most 77%, and have been solved to optimality in times shorter than 2 seconds. The SRP-III GAMS code for the instance in Figure 4.7 is provided in Appendix A.3.

Figure 4.6 SRP-III optimal packing by 27 rectangles of a problem with 10 users over 4 frames with traffic distribution *TR1* (75% of users with data + 25% voice traffic); with average user demand of 116 slots

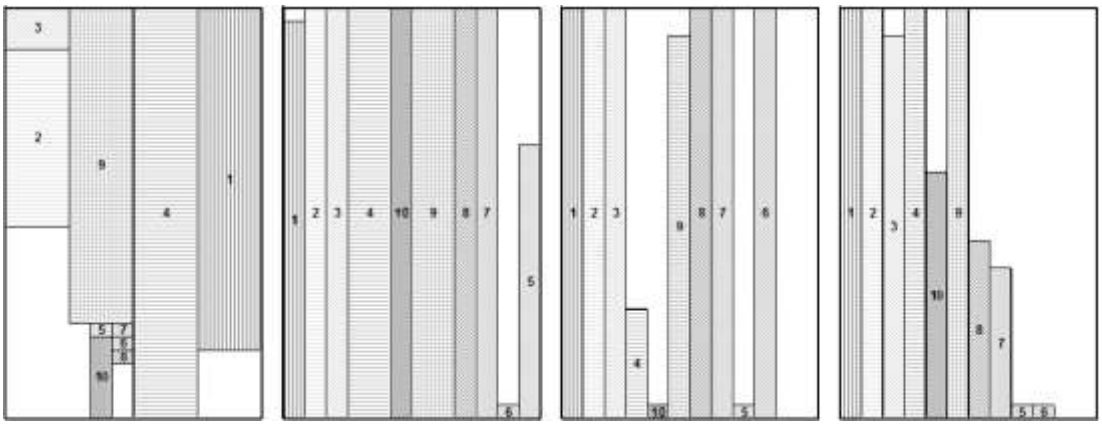


Figure 4.7 SRP-III optimal packing by 40 rectangles of a problem with 10 users over 4 frames with traffic distribution *TR1*; with average user demand of 102 slots

4.3 Discussion

The computational results presented in this chapter illustrate the complex nature of the problems defined in this thesis. A thorough experimentation was performed on two separate data sets with different structures, and the analysis suggests that, although slightly different results are obtained for the performance of the models among the two data sets, the difficulty of obtaining repeated solutions of the problem in short periods of time remains unchanged under changing conditions.

Therefore, the inherent complexity of the problem intuitively forces the utilization of approximation approaches instead of trying to attain exact solutions. Examining the results, one can argue that the solution times for even the smallest problems are very poor and unacceptable for most practical situations.

The practical computational target set by Lodi et al. (2011) was ignored in this thesis for obvious reasons. First, the significant differences between the structures of the single and multiple frame problems render the computational target for the single frame problem infeasible for the multiple frame problem. Second, the main aim and motivation of this study is to introduce novel exact models for the new problem, provide an insight, lay the groundwork and spread canvas for further studies. Nevertheless, in addition to the explanation and discussion of the computational results provided in Sections 4.1 and 4.2, we present and discuss in this section a couple of alternative approaches for the proposed problem for obtaining faster approximate results.

In the next sections, we briefly deliberate two alternative modeling and solution ideas for reaching higher computational performance instead of obtaining optimal solutions. The first is based on a decomposition scheme for the problem while the second involves a two-phase heuristic approach. We believe that such approaches, and probably some more advanced ones, can result in better solution times for small problem instances while making it possible to tackle larger instances in future studies.

4.3.1 Decomposition of SRP

The modeling approach in Chapter 3 employed for SRP involves simultaneous partitioning and packing of the items along frames. An imminent idea for tackling such problems is decomposition.

The problem SRP has two natural components. The first subproblem considers the assignment of the users to the frames using maximum delay constraints, and partitioning the user demands over the assigned frames. Thus, this subproblem involves both the binary decision variables (z_{ij}) for frame selection and the demand partition variables (a_{ij}) for satisfying the feasibility conditions imposed by the QoS parameters.

The second subproblem on the other hand encompasses the packing of the selected user partitions (rectangles) on their assigned frames. If the rectangle sizes are assumed to be variable corresponding to same area values, as in SRP, it would be wise to incorporate some efficient and well-known packing methods from the literature.

According to the capabilities of the selected algorithm, if needed, some randomized size-fixing approaches can also be incorporated as alternative solution approaches. These will be mentioned in detail in the next section.

Obviously, each objective of the three SRP models requires different decision priorities for the first subproblem, which involves user-frame assignments. For example, when the maximization objective of SRP-I is considered, the users to be partitioned should be selected by an adaptation of one of the efficient knapsack solution approaches in the related literature (Kellerer et al., 2005).

The general flow of the decomposition approach can be summarized as below:

(1) Solve initially for z_{ij} and a_{ij}, and fix their values for the second subproblem,
(2) Fix the assigned rectangle sizes, w_{ij} and h_{ij}, and pack them without overlapping for each frame,

(3) Compare the objective function value with the previous ones and the relaxed bounds, update the best solution value, and terminate consequently either after a fixed number of iterations or after a predetermined threshold is reached.

As it can be seen from the above flow, the decomposition idea proposed here is a basic divide-and-conquer strategy, employing a loop for two different models to solve the SRP problem.

All the steps for this decomposition approach can be modeled on the same solvers used in the experimental computations. The main aim is to reduce the initial problem sizes, hence being able to obtain faster solutions for larger problems with more users involved. However, the quality of the solutions should be evaluated carefully with this approach, as it surely will yield suboptimal results.

4.3.2 Two-Phased Heuristic Algorithms for SRP

In this section, ideas for a two-phase heuristic approach for handling SRP are presented. In such a heuristic, the first phase can handle the user selection and partitioning as in the decomposition approach, while the second phase can consider the packing of the partitioned demand through frames. Below, we list some possible ideas that define the outline of this heuristic:

- As in the previous decomposition approach, the users to be partitioned can be selected by an adaptation of efficient knapsack solution approaches (Kellerer et al., 2005).

- Within the heuristic, two different approaches can be considered for partitioning each user's total demand, namely ϕ_i, over the frame sequence. So, regardless of the packing algorithm chosen, one can employ two approaches for this phase. Note that if a user is selected, it should be assigned at least to α_i, and at most to m frames.

- The first approach is a random partitioning, dividing ϕ_i in random t_i pieces, where $\alpha_i \leq t_i \leq m$. In this method, both the selection process of t_i and the areas of the pieces are randomly managed. The assignment of the pieces to frames should be constrained by the maximum delay parameters, λ_i.
- The second partitioning approach involves the division of demand parameter ϕ_i in equal t_i pieces, where again $\alpha_i \leq t_i \leq m$ holds.
- The output of this phase will be the initial frame assignments and areas, but the sizes of the rectangles are still to be constructed by the rectangle packing stage of the heuristic. In both approaches, the idea is simply to begin the packing process as quick and with fewer rectangles per frame as possible.

- For the packing phase of the problem, or explicitly for the DL subframe allocation subproblem, again it is possible to select among the many existing algorithms in the literature.
  - As summarized in Chapter 2, the algorithm developed by Cicconetti et al. (2010), which was represented again in full detail by Lodi et al. (2011) for optimally packing of the so-called distributed permutation zone, is one such appropriate method. The name of the algorithm is recursive tiles and stripes (*RTS*). But, since the authors also take the DL-MAP overhead issue into account, an adaptation of the algorithm is required for SRP.
  - Another algorithm that could be implemented is the eOCSA algorithm, which is developed by So-In et al. (2009b) for handling the BS resource allocation problem in two stages: First, the scheduler sorts each user terminal in a descending order based on their demands for satisfying the QoS throughput guarantee. Then, the bursts are packed from right to left and from bottom to top in the DL subframe. However, since the selection of users is already present at this phase, it would be sufficient to focus on the second stage of their algorithm.

The two-phase heuristic defined by the above outline is again based on a natural decomposition of the problem into two-phases, hence relies on a divide-and-conquer approach. It can be coded and implemented on any platform of preference, and will

produce different bounds for the problem if more than one approach is employed at different phases. However, as the outline indicates, the quality and the runtime of the solutions will be heavily dependent on the performance of the selected algorithms for the separate phases. In addition, it must be kept in mind that the solution for the decomposed problem, even when it is proven to be optimal, will still be suboptimal for SRP, as it provides a relaxation on the original problem. Therefore, the performance evaluation of such a heuristic requires extensive computational experimentation.

As the computational complexities of the SRP models presented in Chapter 3 (Theorem 3.1, Corollary 3.1 and Theorem 3.3) and the computational results of this chapter suggest, acceptable solutions to these problems in tolerable times are very difficult to attain or simply impossible. For this reason, more flexible mechanisms are required for still being able to maintain the novelty brought by SRP models, but with more practical and realistic representations.

In order to reduce the SRP model sizes defined above, we bring forth some fuzzy extensions in the next chapter by adapting consequential objectives and constraints. The foundations for these fuzzy measures and parameters are introduced in order to imitate a more realistic representation of the actual network resources for potential practical problems.

CHAPTER FIVE
A FUZZY PERSPECTIVE FOR SRP

The frame packing solution approaches and typical bin packing algorithms in literature do not allow for partial allocations of the items, as reviewed in Chapter 2. Explicitly, an item must be placed fully in a bin, or cannot be placed at all by such an approach. However, some service resource allocation problems such as wireless communication systems may possess highly variable and unpredictable characteristics. The service demand levels may not be exactly satisfied in some periods of time, or with respect to some criteria.

As mentioned earlier in Section 2.2, Kim et al. (2001) introduced the fuzzy bin packing problem defined as packing non-rigid rectangular items into an open rectangular bin, namely as a strip packing problem. They employed fuzziness in the height dimension by using triangular fuzzy numbers. Their aim was to minimize both the height of a packing and the extra cost due to the size reduction of each item. The authors presented a closed form solution by representing the total cost as the sum of the height cost and the size reduction cost given by a quadratic function. Reducing the height of an item decreased the overall height cost but increased the reduction cost due to lower quality of the item according to the results of the study.

Nasibov (2004) proposed a new approach for the bin packing problem, which was briefly summarized in Section 2.2, with the evaluation of the packing quality under fuzzy source constraints. Using the bin packing notation, the items are to be allocated in two sets of containers, one is comprised by m main containers S_i and the second having only one reserve container S_{m+1}. The author defined four fuzzy relations between the items and the containers imposing certain constraints on the placement of items, which are also mentioned briefly in the study by Eliiyi & Nasibov (2010).

The four fuzzy relations mentioned, which will be presented in this chapter in detail, reflect simply the degrees of the mutual attachment of items, the mutual compatibility of items, the mutual attachment of an item to a container, and the

mutual compatibility of an item to a container, resulting in several matrices whose entries take values from the interval [0, 1]. Containers must be filled with respect to certain conditions, e.g. sufficient degree of filling factor, so that the consistency degree of the final packing is maximized and certain classical total indicators or measures, such as volume or weight, are minimized for the items placed in the reserve container.

Along these lines, we propose a framework to extend both the frame packing models from the literature and the models developed in this thesis, by using an adaptation of the fuzzy constraint approach of Nasibov (2004). Evidently, besides the current objectives of the exact models, an overall quality concept of packing in a single or a sequence of frames should be incorporated in the new models.

We introduce fuzziness in item areas (burst sizes) to be packed within frames, in order to allow partial packing among frames. Assuming triangular fuzzy numbers for area values, minimum admissible service rates and overallocation limits might be more appropriately represented. Hence, by reducing the size of a user demand, more rectangles can be packed in that frame by increasing the capacity utilization. The remainder of the reduced user demand may be considered for packing in the next frames of the sequence, or may not be packed at all, thereby reducing the overall quality level while satisfying higher priority constraints. All these fuzzy user demand sizes are defined as positive integers as in frame packing, which may depend on user data packet classes, profit or priority levels. In addition, a continuous time basis consideration of the frame packing problem might be employed by differentiating between frames. Therefore, the frames are identical in size, but their order is important.

Before we describe the adaptation of the fuzzy attachment and compatibility relations that are defined by Nasibov (2004) in the next section, it should be noted that the integration of these relations to existing models should be employed in as a simple manner as possible. For example, an attachment value of 1 between an item (user or rectangle) and frame will specify that the item must be packed in that frame,

and a zero value will assume no such restriction. In contrast, a compatibility value of 0 between items will indicate the items shouldn't be placed together in any frame, while 1 will describe an absolute independence when placing those in the same frame. To keep the new fuzzy models simpler, some of these relations may be omitted, which are not relevant depending on the nature of the problem or when assigning the values of the corresponding relation matrices might not bring any discernible benefits.

5.1 A General Fuzzy Packing Formulation

The first important contribution of the fuzzy approaches that are presented in this section is the introduction of an overall packing quality level as a secondary objective or performance measure. In this fashion, when the constraints of the existing models are relaxed and suboptimal solutions are obtained, one could measure the acceptability of these new solutions. In order to attain such a fuzzy measure, Nasibov (2004) defined some mutual consistency degrees for containers at hand. His notation is adapted in parallel to the terminology of this thesis, and presented as below.

The items are denoted by user indices $i \in I = \{1 \ldots n\}$, and the containers by frame indices $j \in J = \{1 \ldots m\}$. There are also user classes Q_k ($k \in \{1 \ldots \kappa\}$), equivalent to the ones mentioned in Section 4.2, which group users according to their packet type, MS features, demand sizes, subscription priorities, respective profit values, and maximum delay parameters. Each of these user classes might enforce its specific restrictions (QoS parameters) or constraints in the model.

There is an ordered sequence of frames with fixed rectangular sizes of width W and height H, where m+1 is defined as the reserve frame. Every frame has an area A equal to WxH. As mentioned in the beginning of this chapter, there are four fuzzy attachment and compatibility relations within users and between users and frames defined as follows:

- $R_1 = \|R_1(p,q)\|$ is the symmetric and reflexive relation indicating the mutual attachment degree of users *p* and *q*,
- $R_2 = \|R_2(p,q)\|$ is the symmetric and reflexive relation indicating the mutual compatibility degree of users *p* and *q*,
- $R_3 = \|R_3(p,j)\|$ is the relation indicating the mutual attachment degree of frame *j* and user *p*,
- $R_4 = \|R_4(p,j)\|$ is the relation indicating the mutual compatibility degree of frame *j* and user *p*,

where $p,q \in I$, $j \in J$. For simplicity, we make use of the binary decision variables z_{ij} and δ_{pqj}, which are defined in Chapter 3, here again. Note that z_{ij} = 1 if user *i* is placed in frame *j*, and 0 otherwise. The other variable δ_{pqj} on the other hand, which was first described in Section 3.2, and used particularly in constraints (3.9)-(3.12), deals with feasible placements of user rectangles on the two-dimensional frame area. Recall that δ_{pqj} equals 1 if both user *p* and *q* are placed in same frame *j*, and 0 otherwise.

The matrices representing the relations defined above may be generated or defined with respect to service parameters, deadlines, item priorities or mechanisms that utilize user specific transfer information. Using these fuzzy relations, the consistency degrees for each frame could be defined as below.

$$K_1(j) = 1 - \max_{p,q \in I} \{R_1(p,q) \mid z_{pj} = 1, \delta_{pqj} = 0\}, \forall j \in J. \quad (5.1)$$

$$K_2(j) = \min_{p,q \in I} \{R_2(p,q) \mid z_{pj} = 1, \delta_{pqj} = 1\}, \forall j \in J. \quad (5.2)$$

$$K_3(j) = 1 - \max_{p \in I} \{R_3(p,j) \mid z_{pj} = 0\}, \forall j \in J. \quad (5.3)$$

$$K_4(j) = \min_{p \in I} \{R_4(p,j) \mid z_{pj} = 1\}, \forall j \in J. \quad (5.4)$$

$K_1(j)$ is the measure of separation of the users that are placed in frame *j* from those that are not, with respect to their mutual attachment levels. $K_3(j)$ is the separation measure defined according to attachment relations between the users that are not

allocated to frame j. Similarly, $K_2(j)$ corresponds to the mutual compatibility level of users packed in the same frame, whereas $K_4(j)$ measures the compatibility between a frame j and the users of rectangles which are placed in it. Thus, combining all these degrees in one quantity, the following equation is obtained for overall quality level:

$$\Pi = \min_{j \in J} \min\{K_1(j), K_2(j), K_3(j), K_4(j)\}. \tag{5.5}$$

Due to the fuzzy packing formulation of Nasibov (2004), for each pair of user class Q_k and set of users placed in a frame defined as U_j, the constraints might be redefined as below:

$$F_k(i \mid i \in Q_k \cap U_j, i \in I) \subseteq B_{kj}, \forall j \in J, \forall k \in 1..\kappa. \tag{5.6}$$

where F_k's are defined as arbitrary linear functions over the domain of users belonging to class Q_k placed in frame j, and B_{kj} are convex fuzzy sets. The relation $\subseteq$ in constraint (5.6) may be any of the various comparison relations used for fuzzy sets. In fact, the same constraints might also be formulated in the following manner to reflect frame-wise constraints for the users of same class:

$$\sum_{i \in Q_k \cap U_j} F_k(i) \subseteq B_{kj}, \forall j \in J, \forall k \in 1..\kappa. \tag{5.7}$$

These constraints may include some classical capacity considerations such as total volume, area, weight or cost, as well as the service-quality-related parameters such as maximum delay and minimum throughput encountered in telecommunications.

Therefore, as regards to the definitions and constraints (5.1)-(5.6), the fuzzy packing problem introduced in Nasibov (2004) can be reformulated as a bicriteria optimization problem as follows:

$$\max \Pi, \tag{5.8}$$

$$\min \sum_{i \in U_{m+1}} F(i) \tag{5.9}$$

s.t.

$$F_k(i \mid i \in Q_k \cap U_j, i \in I) \subseteq B_{kj}, \forall j \in J, \forall k \in 1..\kappa. \tag{5.6}$$

where U_{m+1} is the set of users packed in the reserve container. It should be noted that, while the original version of the fuzzy packing model defined above assigns each user (items) exclusively to one frame (containers), we assume that the same attachment and compatibility relations can be generalized and integrated to in the partitioned demand nature of our SRP models.

For simplifying the solution of the problem presented above and being able to estimate the overall quality level in advance, the maximization objective (5.8) can be dropped and a satisfaction degree parameter g can be utilized. In this manner, the fuzzy packing problem will be decomposed into a minimization problem comprising a crisp objective for the reserve container and fuzzy constraints that depend on the value of the parameter g. The decomposed version of the problem takes the following form:

$$\Pi \geq g, g \in (0,1]. \tag{5.10}$$

$$\min \sum_{i \in U_{m+1}} F(i) \tag{5.9}$$

s.t.

$$F_k(i \mid i \in Q_k \cap U_j, i \in I) \subseteq_g B_{kj}, \forall j \in J, \forall k \in 1..\kappa. \tag{5.11}$$

where relation $\subseteq_g$ in constraint (5.11) corresponds to the fulfillment of the constraint F_k with degree g. Moreover, for further improving the quality estimates, some upper bounds for the feasible values of g are determined with their respective proofs (Nasibov, 2004), and implemented in an algorithm developed for the solution of the decomposed problem. Most relevant two of these bounds are also presented here for the sake of completeness.

$$g_{UB1} = \min_{p \in I, j \in J} \max\{1 - R_3(p, j), R_4(p, j)\}. \tag{5.12}$$

$$g_{UB2} = \min_{p,q \in I} \max\{1 - \hat{R}_1(p,q), R_2(p,q)\}. \quad (5.13)$$

In the above equations, $\hat{R}_1$ is the transitive closure, which is equal to R_1^{n-1} composed by the recursive minimax product of the R_1 relation matrices.

Besides the general approach presented above, we propose more SRP specific perspectives regarding the particular problem parameters or decision variable. The first parameter to be analyzed is the demand parameter of the users, as explained in the next section.

5.2 Fuzzy Demand Definitions for SRP

As stated in the beginning of this chapter, due to highly variable and unpredictable characteristics of wireless communication systems, service demand levels may not be fully satisfied in some periods of packet transfers between base stations and user stations. Moreover, having flexibility on minimum demand satisfaction rates of some users or classes of users would allow faster and simpler solutions especially for large problems, even reaching optimality for problem cases not reported in Chapter 4.

For this purpose, the SRP models developed in this thesis and presented in Chapter 3 incorporate a ϕ_i parameter to be able to reduce d_i levels for the problem's time horizon to some extent. However, it is observed that, when large number of users are to be packed (i.e. when $n \geq 20$), more flexibility could be useful to further facilitate solution performance. Thus, there is the additional question to assess the tradeoff for distinguishing between an optimal solution in longer solution times and a suboptimal solution in acceptable times. Obviously, exchanging optimality for fast solutions amounts to incurring additional criteria, most of which depend on subjective decisions.

Consequently, our first straightforward effort focuses on converting the demand parameters ϕ_i to triangular fuzzy numbers. In this manner, both the reduction in

demand satisfaction and the overallocation through the sequence of frames could be handled simultaneously. To be more precise, we can define fuzzy demand parameters $\tilde{\phi}_i = (\phi_{i,\min}, \phi_i, \phi_{i,\max})$ where all $\phi_{i,min}$, ϕ_i , and $\phi_{i,max}$ are positive integers. As expected, ϕ_i is the actual demand level that has the highest demand satisfaction level β=1, and the others depicting minimum demand size reduction and maximum overallocation limits. The corresponding fuzzy parameter definition is summarized in Figure 5.1. It can be said that fuzzy demand parameters penalize reductions in assigned areas or overallocations by lower satisfaction levels β, while relaxing feasibility conditions. The penalties can be directly reflected also on the profit levels to have more tangible performance measures.

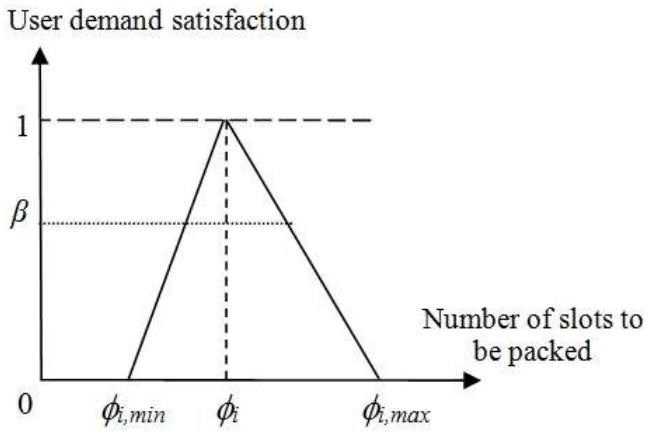


Figure 5.1 A general fuzzy demand parameter definition for SRP models

As a simple illustration on a single frame packing, let's assume that the downlink data demands of two mobile stations *p* and *q* are defined by the triangular fuzzy numbers $\tilde{\phi}_p = (54, 60, 66), \tilde{\phi}_q = (42, 47, 52)$. Three possible rectangular placements of the users in a frame *j* of size *W*=10, *H*=20 are shown in Figure 5.2. P_{1p}, P_{2p} and P_{3p} represent different rectangle orientations and shapes for user *p* with an ideal area of 60 units, whereas P_{1q}, P_{2q} and P_{3q} for user *q*. In the first case of packing, user rectangles P_{1p} and P_{1q} have sizes 5x12 and 4x12 respectively. The unit area separately denoted as *OA* corresponds to the overallocated slot for user *q* (4x12 - 47

= 1). This case is obviously a feasible packing for all the SRP models developed. Demand of user p is exactly satisfied, while overallocation of one slot for user q results in a 80% overall packing quality level β for this case (computed as 80% = (52-48) / (52-47)) by the corresponding membership function.

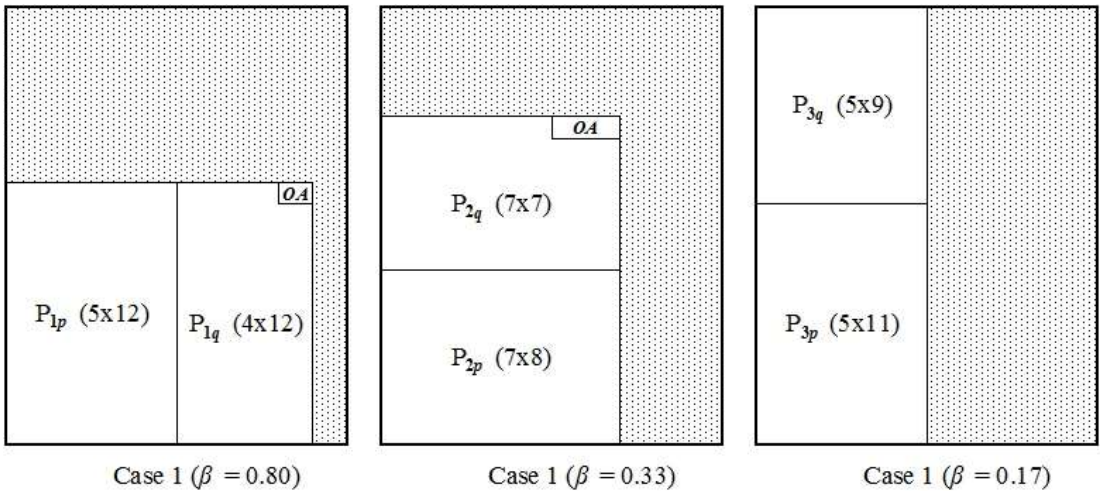


Figure 5.2 Possible placements of two users with fuzzy demands on a frame

Overall packing quality levels for Case 2 and Case 3 given in Figure 5.2 are evaluated as 33% and 17% respectively in the same manner, and considered to be infeasible for the crisp SRP models. It should be noted that, while the partial satisfaction due to fuzzy demand parameters cause lower quality levels, the remaining available unpacked (shaded in Figure 5.2) frame area of 92 slots in Case 1 increases gradually in the other cases (95 slots in Case 2, and 100 in Case 3), allowing more space for probable placements of other user demands.

Regarding the adaptation of the fuzzy demand concept to SRP problems, the necessary modifications are rather straightforward for the SRP-II and SRP-III models where there is no profit involved. It might be more appropriate for the adaptation of the SRP-I model to incur individual profit penalties defined by β_i, instead of an overall quality level affecting the value of the objective function (3.1). Moreover, for the profit maximization case, there is no need to impose an extra limit for

overallocations. Thus, a more appropriate definition of the fuzzy demand parameter for the SRP-I problem should be similar to the one shown in Figure 5.3.

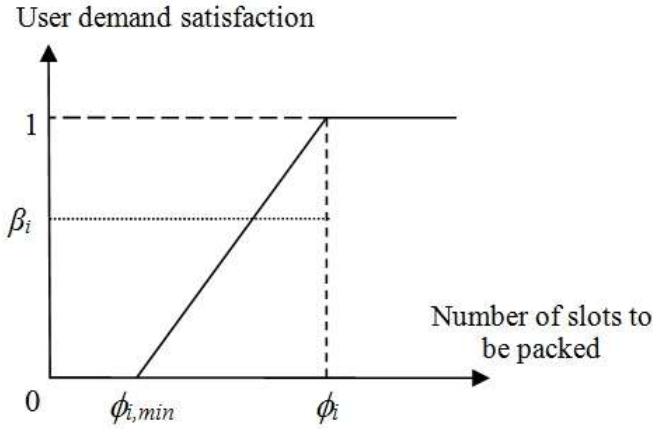


Figure 5.3 Fuzzy demand parameter definition for the SRP-I model

Therefore, in the adapted version of SRP-I for the cases of fuzzy demand values, the modified versions of the objective function (3.1) and constraint (3.4) will take the following form.

$$\max Z_{\tilde{K}} = \sum_{i \in I} \beta_i p_i u_i \tag{5.14}$$

$$\sum_{j \in J} w_{ij} h_{ij} \geq u_i (\phi_{i,\min} + \beta_i (\phi_i - \phi_{i,\min})), \forall i \in I. \tag{5.15}$$

In the above equations, $\beta_i \in [t_i,1]$, and t_i's (>0) are included in order to generalize minimum demand satisfaction levels for each user. The rest of the constraints are the same as defined in Section 3.2. Consequently, depending on the choice of utilization of β_i, either as a decision variable or parameter, fuzzy SRP-I constitutes a convenient alternative for producing acceptable solutions.

Following the same ideas, the SRP-II and SRP-III models can be adapted for the fuzzy case (Figure 5.1) with an additional overall quality maximization objective and by modifying constraint (3.19) used in both models as follows.

$$\max \beta \tag{5.16}$$

$$\phi_{i,\max} - \beta(\phi_{i,\max} - \phi_i) \geq \sum_{j \in J} w_{ij} h_{ij} \geq \phi_{i,\min} + \beta(\phi_i - \phi_{i,\min}), \forall i \in I. \tag{5.17}$$

Similar to the option mentioned for fuzzy SRP-I, objective (5.16) might be dropped and the overall packing quality level β can be taken as a problem parameter for fuzzy SRP-II and SRP-III. The bidirectional bounds imposed by constraint (5.17) can be considered as a pseudo-bicriteria extension for SRP-II, as it also handles the overallocation levels explicitly. Therefore, instead of solving a fuzzy version of the SRP-III model separately, it might suffice to incorporate the overallocation in fuzzy SRP-II model merely by using constraint (5.17). In any case, it is worth investigating and experimenting on mechanisms for assigning precise values for parameters $\phi_{i,min}$ and $\phi_{i,max}$ in order to reach significant performance gains.

5.3 Alternative Objectives for SRP with Fuzzy Information

Apart from the objectives of the SRP models proposed in Chapter 3, further intuitive objectives come to mind. Three of these different approaches deal with decisions regarding the last frame of the sequence. We propose these alternative objective functions for SRP with fuzzy information in this section.

First and the simplest of these objective functions is the minimization of the total area packed in the last frame of the sequence. Obviously, this objective replaces those of SRP-II and SRP-III models, where the case $\sum_{i \in I} \phi_i \leq mA$ holds. Moreover, if the demand density of the problem instance is very low, say less than (m-2)/m, then the length of the frame sequence m should be updated accordingly in order to render the objective function meaningful. Minimization of the total area packed in the last frame of the sequence is evidently analogous to the reserve container objective (5.9) of the fuzzy packing problem presented in Section 5.1, and may be expressed as:

$$\min Z_{LA} = \sum_{i \in I} a_{im} \,, \tag{5.18}$$

where a_{im} is the nonlinear decision variable for the area of the rectangle assigned to user i in the last frame m. Note that, this new objective might be accompanied either by the packing quality objective (5.8) and fuzzy attachment and compatibility constraints (5.6) given in Section 5.1, or by the fuzzy demand linked objective (5.16) and constraint (5.17) given in Section 5.2.

The arguments above are also valid for the second alternative objective function proposed, which basically replaces (5.18) with the following.

$$\min Z_{LP} = \sum_{i \in I} z_{im} \,. \tag{5.19}$$

The objective in (5.19) minimizes the number of rectangles placed in the last frame, where z_{im} is the binary decision variable describing whether user i is assigned a rectangle in the last frame m.

The third objective pertaining to the last frame is essentially a makespan objective directly relevant to the subframe packing problem for WiMAX, which minimizes the latest completion time of all user transfers. Utilizing the time-axis positioning decision variable x_{im} and the rectangle width variable w_{im} for each user in the last frame, this objective can be expressed as:

$$\min Z_{LF} = \max_{i \in I} \{x_{im} + w_{im}\} \,. \tag{5.20}$$

Resuming the main scheme of this study and recalling the computational results summarized in the previous chapter, the most significant challenge in bin packing problems as well as in all frame packing models in the literature, is dealing with the large number of items/users involved. For this reason, a sequential packing approach has been proposed in this thesis to reduce the problem sizes and obtain faster solutions for the new integrated problem. Especially, constraints (3.9) to (3.12),

which deal with the feasible placements of user rectangles on the two-dimensional frame area, contribute a lot to the already highly-combinatorial nature of the problem. Moreover, the computational experimentation for the SRP-II problem suggests promising results that can be implemented for similar problems.

The last objective proposed in this section incurs the maximum number of users involved in every frame of the sequence, named as the fragmentation level. For example, let's assume that in a sequence of two frames there are 9 users packed in the first frame and 7 in the second. The resulting fragmentation level of this packing is equal to 9. Consequently, the objective is the minimization of this fragmentation level, which can be expressed as:

$$\min Z_F = \max_{j \in J} \{ \sum_{i \in I} z_{ij} \}. \tag{5.21}$$

Although very similar to objective (3.16) of the SRP-II model, the main idea of the objective in (5.21) is to maintain an overall balance regarding the user rectangle population of each frame. In a way, DL-MAP overhead of the downlink frame is tried to be minimized via distributing users as evenly as possible with this objective.

A crude lower bound for Z_F in (5.21) can be computed as $\underline{Z}_F = \left\lceil \frac{n}{m} \right\rceil$ for problem instances satisfying $\sum_{i \in I} \phi_i \leq mA$. It should also be noted that this lower bound might require to be updated if objective (5.21) is implemented together with the fuzzy attachment relation R_3 presented in Section 5.1.

In the next section, an example packing is used for generating some of the fuzzy relational matrices presented in Section 5.1.

5.4 An Example Packing Using Fuzzy Relation Matrices

In order to assemble an intelligible and meaningful numerical example, only some of the fuzzy relations defined in Section 5.1 are considered for simplicity. The

sample problem is the instance whose input file is given in Figure B.2 of Appendix B.2, with 10 users over 4 frames of size 12x30, and having a demand density of 80%. There are only two explicit classes of users with respect to packet traffic (data and voice) type, mainly distinguished by their demand sizes and profit values, and defined as Q_1 and Q_2, respectively. As for the fuzzy relations to be included in the example for demonstration, the compatibility relations R_2 and R_4 within users and between users and frames seem appropriate for this setting. In view of the fact that the proposed approaches allow partitioning through different frames, attachment relations might be too restrictive and irrelevant as objectives.

We start by defining relation R_4 first, which is relatively straightforward. Assume that users from class Q_2 are prioritized; the transfers of users with voice traffic must be finished earlier than the last frame. There is no such strict restriction for data traffic users of class Q_1. However, for demonstration purposes, let's assume that a compatibility value of 0.5 is assigned between any user and the last frame, if the demand size of that user is at least half of a frame area. This compatibility scheme will direct the packings of such users to earlier frames. Accordingly, fuzzy relation R_4 for the specific problem instance can be represented by the matrix in Table 5.1.

Table 5.1 Fuzzy relation matrix reflecting the mutual compatibility degrees between users and frames (relation R_4)

	Users									
Frames	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1
4	0.5	0	0	1	0.5	1	1	0	0.5	1

According to the assumptions listed above, one could easily recognize from the contents of the table that the users with voice traffic are 2, 3 and 8, while users 1, 5

and 9 with data traffic have demands at least of 180 slots, i.e. more than half the frame size.

For the fuzzy relation R_2, a similar but more complicated scheme is required. Recall that, in the definition of R_4, the assignment of the users with large demand sizes to the last frame is tried to be avoided. This strategy is adapted between such users for compatibility relation R_2 as follows: Some space should be allowed on the last frame for some of the remaining demand partitions. For this purpose, the users are ordered in their nonincreasing sizes of demand as a first step. Then, we simply enforce m-1 users with the largest demand sizes to be assigned to separate frames, setting their mutual compatibility degrees to 0. In the same manner, other mutual compatibility degrees are defined according to the ratio of mutual demand sums of users to a single frame area, and the resulting matrix is presented in Table 5.2.

Table 5.2 Fuzzy relation matrix reflecting the mutual compatibility degrees between users (relation R_2)

Users	1	2	3	4	5	6	7	8	9	10
1	1	1	0.9	1	0	0.6	0.8	1	0	0.5
2	1	1	1	1	1	1	1	1	1	1
3	0.9	1	1	1	0.9	1	1	1	0.9	0.9
4	1	1	1	1	1	1	1	1	1	1
5	0	1	0.9	1	1	0.6	0.8	1	0	0.5
6	0.6	1	1	1	0.6	1	0.9	1	0.7	0.7
7	0.8	1	1	1	0.8	0.9	1	1	0.8	0.9
8	1	1	1	1	1	1	1	1	1	1
9	0	1	0.9	1	0	0.7	0.8	1	1	0.5
10	0.5	1	0.9	1	0.5	0.7	0.9	1	0.5	1

Including both of the mutual fuzzy relation degrees defined above, the fuzzy packing problem has sufficient input accompanied by the two-dimensional placement constraints. Moreover, as relations R_1 and R_3 are not used, upper bounds g_{UB1} and g_{UB2} defined by (5.12) and (5.13) are inactive. The resulting packing problem can be solved iteratively utilizing the same solvers used for the computations in Chapter 4.

For the sake of simplicity, a simple heuristic approach is used for illustration purposes to obtain the feasible packing shown in Figure 5.4.

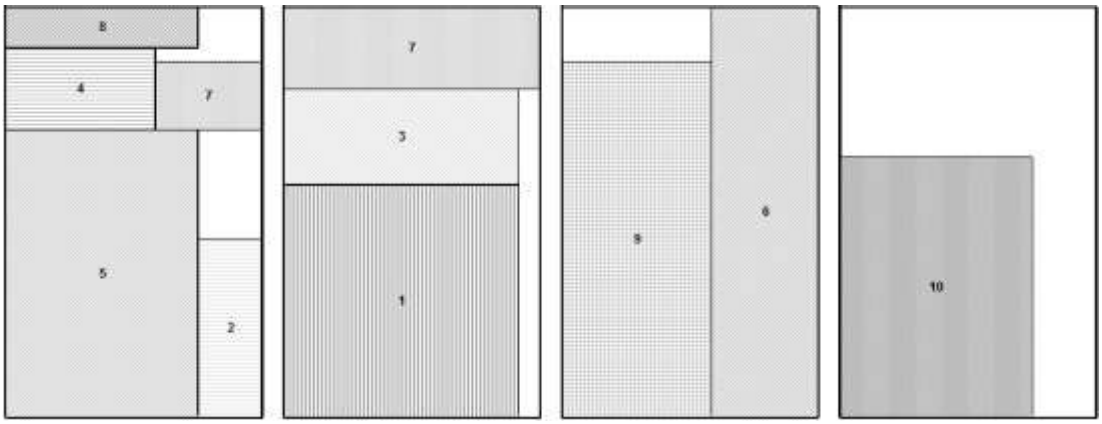


Figure 5.4 A feasible fuzzy SRP solution

The solution presented above has an overall packing quality degree $\pi = 0.7$, which is defined by equation (5.5). 10 users are distributed over 4 frames in 11 rectangles with a total overallocation of 4 slots. $K_4(j)$ values are 1 for all j, hence the resulting π value is due to the degrees of relation R_2 (Table 5.2). More precisely, $K_2(j)$ values for the sequence of frames are respectively 0.8, 0.8, 0.7 and 1.

Basically, the users having the largest demand (users 5, 1, and 9) are distributed to the first three frames. The other users follow in nonincreasing order of their demand, maintaining the largest possible compatibility degrees as possible besides minimum partitioning of users. Once all users are placed in this manner, user 7 is assigned to the last frame in a single partition. However, considering the final frame or the reserve container objective in (5.9), the rectangle of user 7 is moved in 2 partitions to frames 1 and 2, improving the objective and reducing overallocation. The placement of user 10 in the last frame, besides being the most appropriate choice regarding the overall packing quality, is also the best alternative regarding alternative objectives (5.18) and (5.19).

As opposed to classical single-objective models, the proposed fuzzy extensions developed in this chapter aim at satisfying QoS-like service demand constraints for a

specific planning horizon using fuzzy constraints and quantities, while compromising optimality to some extent to obtain efficient frame packings in acceptable computation times.

CHAPTER SIX
CONCLUSION

The span of the bin packing solutions to a variety of problems attracts the challenge of many researchers. Merging all aspects of optimization theory and practice, there are still a lot of topics yet to be investigated. In some application areas, it is observed that some basic assumptions of the classical packing problems do not apply, such as definite dimension sizes or rigidity of item shapes. Therefore, the primary focus of this thesis is chosen as novel packing modeling approaches that might also include fuzzy information. The major contribution of the thesis lies mainly in presenting novel general and representative mathematical models that include important features of the wireless standard, and providing several practical and applicable insights for solving upcoming problems of the area efficiently.

As a first stage of the theoretical study, the novel application area concerning the wireless data package transmissions is examined in Chapter 2, and a complete analysis of literature is performed. Formalizing a more general and flexible modeling perspective in Chapter 3, three exact nonlinear integer programming models with different objective functions are presented, which simultaneously handle multiple packing problems with sequential time considerations. The complexities of SRP-I and SRP-II problems are investigated, as well as an efficient bounding mechanism that is incorporated into the SRP-I model.

While developing the models, it is intended that, with slight modifications, it could easily be possible to solve different problem instances on more specialized optimization packages. In the future, additional assumptions and constraints such as burst compaction might be included in the proposed models. Also, instead of a single objective such as the minimization of the number of partitions or the maximization of utilization as presented in Chapter 3, a quality definition can also be applied using fuzzy constraints, integrating different objectives such as the QoS priorities.

As the experimental computations reported in Chapter 4 suggest, packing alternatives for the same problem instances vary significantly with respect to different objectives. Due to the large number of variables and constraints involved, small sized problem instances could not be solved in acceptable times. As it was explained in Chapter 4, no practical computational runtime target is prioritized in this thesis. Instead, by pointing out the significant differences from the problem structures in the literature and by using different objectives, new exact models are introduced.

Although the exact models developed seem sufficient for effectively capturing several aspects and characteristics of real-life problems, clearly that does not correspond to an efficient solution process using the solvers at hand. The computational results indicate that, even for the more general SRP-I profit maximization model, it is very hard to reach optimality within the determined time limit. Therefore, the solutions of the proposed model are tried to be improved by incorporating tighter bounds than the ones used by the solvers, resulting in significant improvement.

Regarding the other models, SRP-II tends to get more difficult as the length of the frame sequence increases, whereas SRP-III model finds more opportunity to partition demand in a more fragmented way to minimize wasted space. Through extensive computation and comparisons, these two models might be adapted with the same objectives to different application contexts.

The complexity of the considered problems naturally forces the utilization of approximate approaches instead of exact solutions in practical situations. Therefore, ideas for alternative modeling and solution methods are also proposed for reaching higher computational performance instead of obtaining optimal solutions. The performance of the proposed approaches may be worth investigating, as well as some other possible heuristic, metaheuristic or hybrid metaheuristic procedures.

Throughout Chapter 5 of the thesis, a general fuzzy packing modeling framework has been presented with necessary adaptations regarding the extensions of the SRP model structures. Also, some implementable alternative objectives are proposed for reinforcing the fuzzy versions of SRP in practical situations. Hence, besides maximization of the overall quality level, several objectives such as area utilization, width of the packing and overallocation are also investigated. The proposed fuzzy perspective is applied to an SRP problem instance that has previously been used in Chapter 4, defining fuzzy compatibility degrees specific to problem data. It should be noted that, besides the use of fuzzy demand parameters, the fuzzification of maximum delay parameters is also possible especially for long sequences of frames (e.g. m>4). In such a case, the more consecutive the partitioning of the users, the better may be the overall packing quality.

Utilizing α-cuts for user demand sizes and user class distinctions, different packing combinations might be generated and evaluated via different heuristics from literature. The fuzzy relations between the structure of problem instances and the packing method that is to be implemented for solving that instance might be defined even from a hyperheuristic angle. Furthermore, a flexible heuristic selection mechanism on a higher decision level may be designed depending on area and priority parameters.

For adapting typical bin packing heuristics into the fuzzy approaches developed in this thesis, it is always possible to employ one of the ranking and defuzzification techniques proposed in the literature (e.g. Dubois & Prade, 1983; Fortemps & Roubens, 1996).

Some extensions of the developed SRP models may be applied to assignment-like problems with quadratic measures. The models can also be adapted for problems involving time-related placement of items. For example, the adaptation of SRP models to storage area allocation problems, such as container handling and berth operations in port logistics might be worthy of further examination. Some special cases of our models might be adapted for the relevant problems in that area.

Moreover, new models for the existing scheduling problems could be developed by utilizing special features of the two-dimensional packing models presented in this thesis.

The specific and overall computational complexity aspects might further be explored, and more detailed algorithm performance analyses can be made for future solution efforts involving real-life implementation issues. By following the most recent studies in the literature, new algorithms might be included to form a benchmarking environment with a complete comparison base, possibly employing the developed fuzzy approaches, as well.

REFERENCES

Aho, A.V., Hopcroft, J.E., & Ullman, J. (1983). *Data structures and algorithms*. Boston: Addison-Wesley.

Azar, Y., & Regev, O. (2001). On-line bin-stretching. *Theoretical Computer Science, 268* (1), 17-41.

Bacioccola, A., Cicconetti, C., Lenzini, L., Mingozzi, E., & Erta, A. (2007). A downlink data region allocation algorithm for IEEE 802.16e OFDMA. In *Proceedings of 6th International Conference on Information, Communications & Signal Processing (ICICS 2007)* (1-5). Singapore: IEEE Press.

Baker, B.S., & Coffman, E.G., Jr. (1981). A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM Journal on Algebraic and Discrete Methods, 2* (2), 147-152.

Bansal, N., Correa, J.R., Kenyon, C., & Sviridenko, M. (2006). Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research, 31* (1), 31-49.

Batu, T., Rubinfeld, R., & White, P. (2005). Fast approximate PCPs for multidimensional bin-packing problems. *Information and Computation, 96* (1), 42-56.

Ben-Shimol, Y., Kitroser, I., & Dinitz, Y. (2006). Two-dimensional mapping for wireless OFDMA systems. *IEEE Transactions on Broadcasting, 52* (3), 388-396.

Berger, B., & Leighton, T. (1998). Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology, 5* (1), 27-40.

Birgin, E.G., & Sobral, F.N.C. (2008). Minimizing the object dimensions in circle and sphere packing problems. *Computers & Operations Research, 35* (7), 2357-2375.

Bischoff, E.E. (2006). Three-dimensional packing of items with limited load bearing strength. *European Journal of Operational Research, 168* (3), 952-966.

Boschetti, M.A. (2004). New lower bounds for the three-dimensional finite bin packing problem. *Discrete Applied Mathematics, 140* (1-3), 241-258.

Caprara, A., & Monaci, M. (2004). On the two-dimensional knapsack problem. *Operations Research Letters, 132* (1), 5-14.

Castro Silva, J.L., Soma N.Y., & Maculan, N. (2003). A greedy search for the three-dimensional bin packing problem: the packing static stability case. *International Transactions in Operational Research, 10* (2), 141-153.

Chekuri, C., & Khanna, S. (2004). On multi-dimensional packing problems. *SIAM Journal on Computing, 33* (4), 837-851.

Cicconetti, C., Lenzini, L., Lodi, A., Martello, S., Mingozzi, E.C., & Monaci, M. (2010). Efficient two-dimensional data allocation in IEEE 802.16 OFDMA. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM 2010)* (2160-2168). New Jersey: IEEE Press.

Coffman, E.G., Jr., Garey, M.R., & Johnson, D.S. (1978). An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing, 7* (1), 1-17.

Coffman, E.G., Jr., Lueker, G.S., & Rinnooy Kan, A.H.G. (1988). Asymptotic methods in the probabilistic analysis of sequencing and packing heuristics. *Management Science, 34* (3), 266-290.

Coffman, E.G., Jr., Garey, M.R., & Johnson, D.S. (1996). Approximation algorithms for bin packing: a survey. In D. Hochbaum, (Ed.). *Approximation Algorithms for NP-Hard Problems* (46-93). Boston: PWS Publishing.

Coffman, E.G., Jr., & Lueker, G.S. (2006). Approximation algorithms for extensible bin packing. *Journal of Scheduling, 9* (1), 63-69.

Coffman, E.G., Jr., Csirik, J., Rónyai, L., & Zsbán, A. (2008). Random-order bin packing. *Discrete Applied Mathematics, 156* (14), 2810-2816.

Cohen, R., & Katzir, L. (2008). Computational analysis and efficient algorithms for micro and macro OFDMA scheduling. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM 2008)* (511-519). New Jersey: IEEE Press.

Crainic, T.G., Perboli, G., & Tadei, R. (2009). TS2PACK: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research, 195* (3), 744-760.

Dantzig, G. (1957). Discrete-variable extremum problems. *Operations Research, 5* (2), 266-288.

Davies, A.P., & Bischoff, E.E. (1999). Weight distribution considerations in container loading. *European Journal of Operational Research, 114* (3), 509-527.

Dexter, F., Macario, A., & Traub, R. (1999). Which algorithm for scheduling add-on elective cases maximizes operating room utilization?: Use of bin packing algorithms and fuzzy constraints in operating room management. *Anesthesiology, 91* (5), 1491-1500.

Dickinson, J.K., & Knopf, G.K. (2000). A moment based metric for 2-D and 3-D packing. *European Journal of Operational Research, 122* (1), 133-144.

Dubois, D., & Prade, H. (1983). Ranking fuzzy numbers in the setting of possibility theory. *Information Sciences, 30* (3), 183-224.

Eilon, S., & Christofides, N. (1971). The loading problem. *Management Science Theory Series, 17* (5), 259-268.

Eliiyi, U., & Eliiyi, D.T. (2009). Applications of bin packing models through the supply chain. *International Journal of Business and Management, 1* (1), 11-19.

Eliiyi, U., & Nasibov, E. (2010). A fuzzy perspective for two-dimensional packing of variable-sized items. In *24th Mini EURO Conference Selected Papers (MEC-EurOPT-2010)*. (177-182). Vilnius: VGTU Technika.

Epstein, L. (2003). Bin stretching revisited. *Journal Acta Informatica*, *39* (2), 97-117.

Faina, L. (2000). A global optimization algorithm for the three-dimensional packing problem. *European Journal of Operational Research*, *126* (2), 340-354.

Faroe, O., Pisinger, D., & Zachariasen, M. (2003). Guided local search for the three-dimensional bin-packing problem. *INFORMS Journal on Computing*, *15* (3), 267-283.

Fasano, G. (2004). A MIP approach for some practical packing problems: Balancing constraints and tetris-like items. *4OR: A Quarterly Journal of Operations Research*, *2* (2), 161-174.

Fasano, G. (2008). MIP-based heuristic for non-standard 3D-packing problems. *4OR: A Quarterly Journal of Operations Research*, *6* (3), 291-310.

Federgruen, A., & van Ryzin, G. (1997). Probabilistic analysis of a generalized bin packing problem and applications. *Operations Research*, *45* (4), 596-609.

Fekete, S.P., & Schepers, J. (2001). New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, *91* (1), 11-31.

Fekete, S.P., & Schepers, J. (2004a). A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, *29* (2), 353-368.

Fekete, S.P., & Schepers, J. (2004b). A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, *60* (2), 311-329.

Fekete, S.P., Schepers, J., & van der Veen, J.C. (2007). An exact algorithm for higher-dimensional orthogonal packing. *Operations Research, 55* (3), 569-587.

Fischetti, M., Martello, S., & Toth, P. (1989). The fixed job schedule problem with working-time constraints. *Operations Research, 37* (3), 395-403.

Fortemps, P., & Roubens, M. (1996). Ranking and defuzzification methods based on area compensation. *Fuzzy Sets and Systems, 82* (3), 319-330.

Garey, M.R., Graham, R.L., & Ullman, J.D. (1972). Worst-case analysis of memory allocation algorithms. In *Proceedings of the Fourth Annual ACM symposium on Theory of Computing (STOC '72)* (143-150). New York: ACM Press.

Garey, M.R., & Johnson, D.S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: W.H. Freeman.

Garey, M.R., Graham, R.L., & Johnson, D.S. (1978). Performance guarantees for scheduling algorithms. *Operations Research, 26* (1), 3-21.

Hemminki, J., Leipala, T., & Nevalainen, O. (1998). On-line packing with boxes of different sizes. *International Journal of Production Research, 36* (8), 2225-2245.

Hifi, M. (2002). Approximate algorithms for the container loading problem. *International Transactions in Operational Research, 9* (6), 747-774.

Hurkens, C., Lodi, A., Martello, S., Monaci, M., & Woeginger, G. (2011). Complexity and approximation of an area packing problem. *Optimization Letters, 6* (1), 1-9.

IEEE 802.16-2009 - IEEE standard for local and metropolitan area networks, part 16: Air interface for broadband wireless access systems (revision of IEEE 802.16-2004), (2009). Retrieved March 8, 2010, from http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5062485&queryText%3DIEEE+Std+802.16%E2%84%A2-2009

IEEE Std 802.16m - IEEE standard for local and metropolitan area networks - part 16: Air interface for broadband wireless access systems, amendment 3: Advanced air interface to IEEE 802.16-2009, (2011). Retrieved January 24, 2012, from http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5765736&queryText%3DIEEE+Std+802.16%E2%84%A2-2009

Israeli, A., Rawitz, D., & Sharon, O. (2008). On the complexity of sequential rectangle placement in IEEE 802.16/WiMAX systems. *Information and Computation, 206* (11), 1334-1345.

Johnson, D.S. (1973). *Near-optimal bin packing algorithms*. Ph.D. Thesis, Massachusetts Institute of Technology, Department of Mathematics, Cambridge, Massachusetts, 1973.

Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Jr., & Graham, R.L. (1974). Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing, 3* (4), 299-325.

Kellerer, H., Pferschy, U., & Pisinger, D. (2005). *Knapsack problems*. Berlin: Springer Verlag.

Kenyon, C. (1996). Best-fit bin-packing with random order. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms* (359-364). Atlanta: SIAM Press.

Kenyon, C., & Rémila, E. (2000). A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research, 25* (4), 645-656.

Kim, J.K., Lee-Kwang, H., & Yoo, S.W. (2001). Fuzzy bin packing problem. *Fuzzy Sets and Systems, 120* (3), 429-434.

Kohayakawa, Y., Miyazawa, F.K., Raghavan, P., & Wakabayashi, Y. (2004). Multidimensional cube packing. *Algorithmica, 40* (3), 173-187.

Korte, B., & Vygen, J. (2008). *Combinatorial optimization, theory and algorithms*. Berlin: Springer-Verlag.

Lim, A., Rodrigues, B., & Wang, Y. (2003). A multi-faced buildup algorithm for three-dimensional packing problems. *Omega, 31* (6), 471-481.

Lim, A., Rodrigues, B., & Yang, Y. (2005). 3-D container packing heuristics. *Applied Intelligence, 22* (2), 125-134.

Lins, L., Lins, S., & Morabito, R. (2002). An *n*-tet graph approach for non-guillotine packings of *n*-dimensional boxes into an *n*-container. *European Journal of Operational Research, 141* (2), 421-439.

Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research, 141* (2), 241-252.

Lodi, A., Martello, S., & Vigo, D. (2004). TSpack: A unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research, 131* (1-4), 203-213.

Lodi, A., Martello, S., Monaci, M., Cicconetti, C., Lenzini, L., Mingozzi, E.C., Eklund, C., & Moilanen, J. (2011). Efficient two-dimensional packing algorithms for mobile WiMAX. *Management Science, 57* (12), 2130-2144.

Maarouf, W.F., Barbar, A.Z., & Owayjan, M.J. (2008). A new heuristic algorithm for the 3D bin packing problem. In *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering* (342-345). Netherlands: Springer.

Martello, S., & Toth, P. (1990). *Knapsack problems: Algorithms and computer implementations*. New York: John Wiley.

Martello, S., & Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management Science, 44* (3), 388-399

Martello, S., Pisinger, D., & Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research, 48* (2), 256-267.

Miyazawa, F.K., & Wakabayashi, Y. (1997). An algorithm for the three-dimensional packing problem with asymptotic performance analysis. *Algorithmica*, *18* (1), 122-144.

Miyazawa, F.K., & Wakabayashi, Y. (2000). Approximation algorithms for the orthogonal z-oriented three-dimensional packing problem. *SIAM Journal on Computing*, *29* (3), 1008-1029

Miyazawa, F.K., & Wakabayashi, Y. (2003). Cube packing. *Theoretical Computer Science*, *297* (1-3), 355-366.

Miyazawa, F.K., & Wakabayashi, Y. (2007). Two- and three-dimensional parametric packing. *Computers & Operations Research*, *34* (9), 2589-2603.

Miyazawa, F.K., & Wakabayashi, Y. (2009). Three-dimensional packings with rotations. *Computers & Operations Research*, *36* (10), 2801-2815.

Mok, P.Y., Kwong, C.K., & Wong, W.K. (2007). Optimisation of fault-tolerant fabric-cutting schedules using genetic algorithms and fuzzy set theory. *European Journal of Operational Research*, *177* (3), 1876-1893.

Nasibov, E.N. (2004). An algorithm for constructing an admissible solution to the bin packing problem with fuzzy constraints. *Journal of Computer and Systems Sciences International*, *43* (2), 205-212.

Nasibov, E.N. (2007). A problem of task allocation with fuzzy information and two-stage solution algorithm. *Automatic Control and Computer Sciences*, *41* (4), 196-202.

Necker, M.C., Köhn, M., Reifert, A., Scharf, J., & Sommer, J. (2008). Optimized frame packing for OFDMA systems. In *Proceedings of the 67th IEEE Vehicular Technology Conference (VTC2008 - Spring)* (1483-1488). Singapore: IEEE Press.

Ohseki, T., Morita, M., & Inoue, T. (2007). Burst construction and packet mapping scheme for OFDMA downlinks in IEEE 802.16 systems. In *Proceedings of IEEE*

Global Telecommunications Conference (4307-4311). Washington, DC: IEEE Press.

Padberg, M. (2000). Packing small boxes into a big box. *Mathematical Methods of Operations Research, 52* (1), 1-21.

Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research, 141* (2), 382-392.

Pisinger, D., & Segurdi, M. (2002). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization, 2* (2), 154-167.

Runarsson, T.P., Jonsson, M.T., & Jensson, P. (1996). Dynamic dual bin packing using fuzzy objectives. In *Evolutionary Computation, Proceedings of IEEE International Conference* (219-222). Nagoya: IEEE Press.

Scheithauer, G. (1999). LP-based bounds for the container and multi-container loading problem. *International Transactions in Operational Research, 6* (2), 199-213.

So-In, C., Jain, R., & Tamimi, A.K. (2009a). Scheduling in IEEE 802.16e mobile WiMAX networks: Key issues and a survey. *IEEE Journal on Selected Areas in Communications, 27* (2), 156-171.

So-In, C., Jain, R., & Tamimi, A.K. (2009b). OCSA: An algorithm for burst mapping in IEEE 802.16e mobile WiMAX networks. In *Proceedings of 15th Asia Pacific Conference on Communications (APCC 2009)* (52-58). Shanghai: IEEE Press.

So-In, C., Jain, R., & Tamimi, A.K. (2009c). eOCSA: An algorithm for burst mapping with strict QoS requirements in IEEE 802.16e mobile WiMAX networks. In *Proceedings of 2nd Wireless Days (2009 IFIP)* (1-5). Paris: IEEE Press.

Terno, J., Scheithauer, G., Sommerweiss, U., & Riehme, J. (2000). An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research*, *123* (2), 372-381.

Ullman, J.D. (1971). *The performance of a memory allocation algorithm.* Technical Report No. 100, Princeton University, New Jersey: Princeton Press.

Wang, T., Feng, H., & Hu, B. (2008). Two-dimensional resource allocation for OFDMA system. In *Proceedings of the IEEE ICC Workshops'08* (1-5). New Jersey: IEEE Press.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183* (3), 1109-1130.

Wongthavarawat, K., & Ganz, A. (2003). Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems. *International Journal of Communication Systems*, *16* (1), 81-96.

Wu, Y-L, Huang, W., Lau, S-C., Wong, C.K., & Yang, G.H. (2002). An effective quasi-human based heuristic for solving the rectangle packing problem. *European Journal of Operational Research*, *141* (2), 341-358.

APPENDICES

APPENDIX A
SAMPLE GAMS CODES

A.1 Sample GAMS Code for an instance of SRP-I

GAMS code of the SRP-I model for a problem instance is provided in this appendix. Note that the below code belongs to the problem instance whose output was depicted in Figure 4.5.

```
$title Sequential Rectangular Packing (SRP) - 5-Maximizing Profit
with Bounds
Sets  i users  / 1*10 /
       j frames  / 1*2 /
       dim dimensions used in overlapping constraints / 1*4 /;
Scalars W frame width / 12 /
       H frame height / 30 /
       nUsers / 10 /
       nFrames / 2 /
       availableProfit / 3102 /
       knapsackRelaxationBound / 3065 /
       lowerBound / 2893 /;
Parameters  d(i)  total amount of data request of user i
       /1   28,2  99,3  103,4 190,5 90,6  49,7  78,8  35,9  43,10 42/
        pr(i)  profit gained from user i
       /1   168,2 396,3 515,4 950,5 90,6  294,7 312,8 35,9  258,10
      84/
        s(i)  minimum data transfer rate of user i per frame (QoS
parameter)
       /1   28,2  99,3  103,4 190,5 90,6  49,7  78,8  35,9  43,10 42/
        lambda(i) maximum delay period for user i (QoS parameter)
       /1   7,2   2,3   2,4   2,5   2,6   2,7   2,8   2,9   2,10  2/
        theta(i)     latest frame to maintain or to begin the data
transfer for user i
       /1   3,2   3,3   3,4   3,5   3,6   3,7   3,8   3,9   3,10  3/;

scalar  A frame area;
A =  W*H;

alias(i,p,q);
alias(j,k);

parameters phi(i) minimum amount of data to be transferred to user i
in the problem frame sequence
           alpha(i);
* actual demand parameter phi
phi(i) = min (nFrames * s(i), d(i));
* minimum number of frames to be assigned to user i
alpha(i)= ceil( phi(i) / A);

binary variables
u_i(i) showing whether user i is selected for data transfer in this
frame sequence
z_ij(i,j) showing whether user i is assigned a rectangle in frame j
or not
delta(p,q,j) for choosing the users allocated in the same frame
gamma(p,q,j,dim) used in location overlapping constraints
sigma(i,j) for connecting r_ij and lambda_i (which are used for
maximum delay constraints)
```

```
;

integer variables

x_ij(i,j) x-coordinate of the left-bottom corner of the rectangle
assigned to user i in frame j
y_ij(i,j) y-coordinate of the left-bottom corner of the rectangle
assigned to user i in frame j
w_ij(i,j) width of the rectangle allocated to user i in frame j
h_ij(i,j) height of the rectangle allocated to user i in frame j
a_ij(i,j) area of the rectangle allocated to user i in frame j
r_ij(i,j) total remaining demand for user i after frame j;

free variables

total_profit profit sum of all selected users;

equations

profitSum objective function definition - maximizing total profit
areas(i,j) nonlinear area equalities for each frame
demands(i) demand constraints for each user through the frame
sequence
selectedUser(i,j) if user selected then pack all her demand

firstFrame(i,j) total remaining demand for each user i after frame 1
otherFrames(i,j) total remaining demand for each user i after next
frames

* sigma variable definitions
rij_sigmaLower(i,j) defining sigma variables for r_ij with LB
rij_sigmaUpper(i,j) defining sigma variables for r_ij with UB
rij_zij_lambda_relation(i,j) relating rij and zij variables with
maximmum delay parameters
*for avoiding maximum delay violation after no demand remaining
rij_lastFrame(i,j) r_ij upper bound only if user i is selected

xWidth(i,j) rectangle position (also dimension) constraints for each
user & frame on the horizontal axis
yHeight(i,j) rectangle position (also dimension) constraints for
each user & frame on the vertical axis
assignedFrames(i) using parameters alpha's as lower bounds for
assigned frames for user i
thetaBounds(i) for forcing the transfer beginning frame for user i
thetaBoundsForUserSelection(i) for forcing the selection of a user i
if ongoing transfer

* linking constraints for variables z_ij between x_ij, y_ij, w_ij
and h_ij
z0xBinding(i,j) x_ij lower bounds
z1xBinding(i,j) x_ij upper bounds
z0yBinding(i,j) y_ij lower bounds
z1yBinding(i,j) y_ij upper bounds
z0wBinding(i,j) w_ij lower bounds
z1wBinding(i,j) w_ij upper bounds
z0hBinding(i,j) h_ij lower bounds
z1hBinding(i,j) h_ij upper bounds

* Location overlapping constraints for rectangles in each frame,
```

```
* (a) for choosing the users allocated in the same frame
differentFrame(p,q,j) lower bound: at least one user is not
allocated on frame j
sameFrame(p,q,j) upper bound: both users p and q may be allocated on
frame j

* (b) relative positions on the horizontal axis
gammaLeft(p,q,j) user p on the left of user q if gamma_1 = 1
gammaRight(p,q,j) user p on the right of user q if gamma_2 = 1
gammaLeftRight(p,q,j) p cannot be both on the left and right of user
q if both users are on the same frame

* (c) positions on the vertical axis
gammaBelow(p,q,j) user p below the user q if gamma_3 = 1
gammaAbove(p,q,j) user p above the user q if gamma_4 = 1
gammaBelowAbove(p,q,j) p cannot be both below and above the user q
if both users are on the same frame

* linking the logical constraints (a), (b) and (c) above
gammaLeftBelow(p,q,j)
gammaRightAbove(p,q,j)
allGammas(p,q,j)
;

* CONSTRAINT DEFINITIONS
profitSum.. total_profit =e= sum ((i),u_i(i)*pr(i)) ;

areas(i,j).. a_ij(i,j) =e= w_ij(i,j)*h_ij(i,j);

* demand constraints for each user through the frame sequence
demands(i).. sum(j, a_ij(i,j)) =g= u_i(i)*phi(i);
* if user selected then pack all her demand
selectedUser(i,j).. z_ij(i,j) =l= u_i(i);

*feasibility check with respect to total frame area
* forcing the beginning of ongoing transfers that remain from
previous sequences
thetaBounds(i).. sum (j$(ord(j) <= theta(i)),z_ij(i,j)) =g= u_i(i);
* force the selection of a user i if her transfer is continuing
(<=m)
thetaBoundsForUserSelection(i).. nFrames - theta(i) =l=
u_i(i)*(1+nFrames-theta(i))-1;

* rectangle position (also dimension) constraints for each user &
frame
xWidth(i,j)..  x_ij(i,j) + w_ij(i,j) =l= W;
yHeight(i,j).. y_ij(i,j) + h_ij(i,j) =l= H;

* using parameters alpha i's as lower bounds
assignedFrames(i).. sum(j,z_ij(i,j)) =g= u_i(i)*alpha(i);

* linking constraints for variables z_ij between x_ij, y_ij, w_ij
and h_ij
z0xBinding(i,j).. z_ij(i,j) - 1 =l= x_ij(i,j);
z1xBinding(i,j).. x_ij(i,j) =l= (W-1)* z_ij(i,j);
z0yBinding(i,j).. z_ij(i,j) - 1 =l= y_ij(i,j);
z1yBinding(i,j).. y_ij(i,j) =l= (H-1)* z_ij(i,j);
z0wBinding(i,j).. z_ij(i,j) =l= w_ij(i,j);
z1wBinding(i,j).. w_ij(i,j) =l= W* z_ij(i,j);
```

```
z0hBinding(i,j).. z_ij(i,j) =l= h_ij(i,j);
z1hBinding(i,j).. h_ij(i,j) =l= H* z_ij(i,j);

* Location overlapping constraints for rectangles in each frame,
* (a) for choosing the users allocated in the same frame
differentFrame(p,q,j)$(ord(p)<ord(q)).. z_ij(p,j) + z_ij(q,j) =l=
delta(p,q,j) + 1;
sameFrame(p,q,j)$(ord(p)<ord(q)).. z_ij(p,j) + z_ij(q,j) =g=
2*delta(p,q,j);

* (b) relative positions on the horizontal axis
gammaLeft(p,q,j)$(ord(p)<ord(q)).. x_ij(p,j) + w_ij(p,j) - x_ij(q,j)
=l= (1-gamma(p,q,j,"1"))*W;
gammaRight(p,q,j)$(ord(p)<ord(q)).. x_ij(q,j) + w_ij(q,j) -
x_ij(p,j) =l= (1-gamma(p,q,j,"2"))*W;
gammaLeftRight(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"2") =l= 2 - delta(p,q,j);

* (c) positions on the vertical axis
gammaBelow(p,q,j)$(ord(p)<ord(q)).. y_ij(p,j) + h_ij(p,j) -
y_ij(q,j) =l= (1-gamma(p,q,j,"3"))*H;
gammaAbove(p,q,j)$(ord(p)<ord(q)).. y_ij(q,j) + h_ij(q,j) -
y_ij(p,j) =l= (1-gamma(p,q,j,"4"))*H;
gammaBelowAbove(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"3") +
gamma(p,q,j,"4") =l= 2 - delta(p,q,j);

* linking the logical constraints (a), (b) and (c) above
gammaLeftBelow(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"3") =l= 2 * (1 - z_ij(p,j) + delta(p,q,j));
gammaRightAbove(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"2") +
gamma(p,q,j,"4") =l= 2 * (1 - z_ij(q,j) + delta(p,q,j));
allGammas(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"2") + gamma(p,q,j,"3") + gamma(p,q,j,"4") =g=
delta(p,q,j);

* PhD Progress 5 2011-12 Ch.3 - corrections for total remaining
demand
firstFrame(i,j)$(ord(j) = 1).. r_ij(i,j) =e= phi(i)-a_ij(i,j);
otherFrames(i,j)$(ord(j) > 1)..  r_ij(i,j) =e= r_ij(i,j-1)-
a_ij(i,j);

* maximum delay constraints (10.07.2012 ideas)
rij_sigmaLower(i,j).. 1 - A*sigma(i,j) =l= r_ij(i,j);
rij_sigmaUpper(i,j).. r_ij(i,j) =l= phi(i)*(1-sigma(i,j));
rij_zij_lambda_relation(i,j)$(ord(j) <= nFrames-lambda(i)).. sum(k $
(ord(k) > ord(j) and ord(k) <= ord(j)+lambda(i)), z_ij(i,k)) =g=
z_ij(i,j)-sigma(i,j);
*for avoiding maximum delay violation after no demand remaining
rij_lastFrame(i,j)$(ord(j)=card(j)).. r_ij(i,j) =l= phi(i)*(1-
u_i(i));

* Bounds
* bounding r_ij
r_ij.UP(i,j) = phi(i);
r_ij.LO(i,j) = 1-A;
total_profit.LO = lowerBound;
total_profit.UP = knapsackRelaxationBound;

MODEL SeqRectPack /ALL/;
```

```
SeqRectPack.optfile=1;

* instruct SCIP to display less frequently
$onecho > scip.opt
display/freq = 500
$offecho

OPTION optca=1e-5, optcr=1e-5, minlp=scip, reslim=600,
iterlim=2000000;

scalar starttime; starttime = jnow

SOLVE SeqRectPack USING MINLP MAXIMIZING total_profit;
execerror=0;

file textOutput /tr1n10m2PR10.gout2/;
textOutput.pc=6;

put textOutput, SeqRectPack.modelstat:0:0, SeqRectPack.solvestat:0:0
/;
put SeqRectPack.objVal:0:0, SeqRectPack.resUsd:0:4,
SeqRectPack.nodUsd:0:0, SeqRectPack.objEst:0:0  /;
put SeqRectPack.numVar:0:0, SeqRectPack.numEqu:0:0,
SeqRectPack.numNZ:0:0, SeqRectPack.numNLNZ:0:0 /;
put card(i):0:0, card(j):0:0, W:0:0, H:0:0, sum(i, phi(i)):0:0 /;
put sum(i, u_i.l(i)):0:0, sum(i, phi(i) * u_i.l(i)):0:0,
availableProfit:0:0, lowerBound:0:0, knapsackRelaxationBound:0:0  /;

loop(i $ u_i.l(i), put ord(i):0:0, sum(j, z_ij.l(i,j)):0:0,
phi(i):0:0, sum(j, a_ij.l(i,j)):0:0, pr(i):0:0  /;
         loop(j $ z_ij.l(i,j),
put ord(j):4:0, x_ij.l(i,j):0:0, y_ij.l(i,j):0:0, w_ij.l(i,j):0:0,
h_ij.l(i,j):0:0;
put /;);
);
```

A.2 Sample GAMS Code for an instance of SRP-II

GAMS code of the SRP-II model for a problem instance is provided in this appendix. Note that the below code belongs to the problem instance whose output was depicted in Figure 4.3.

```
$title Sequential Rectangular Packing (SRP) - 2-Minimizing Partition
Sets    i users  / 1*10 /
       j frames  / 1*2 /
       dim dimensions used in overlapping constraints / 1*4 /;
Scalars W frame width / 12 /
       H frame height / 30 /
       nUsers / 10 /
       nFrames / 2 /
;
Parameters  d(i)  total amount of data request of user i
       /1   113,2 33,3  46,4  114,5 46,6  109,7 123,8 50,9  30,10 30/
        s(i)  minimum data transfer rate of user i per frame (QoS
parameter)
       /1   113,2 33,3  46,4  114,5 46,6  109,7 123,8 50,9  30,10 30/
        lambda(i) maximum delay period for user i (QoS parameter)
       /1   1,2   1,3   1,4   1,5   2,6   2,7   1,8   1,9   1,10  1/
        theta(i)     latest frame to maintain or to begin the data
transfer for user i
       /1   2,2   2,3   2,4   2,5   2,6   2,7   2,8   2,9   2,10  2/;

scalar  A frame area;
A =  W*H;

alias(i,p,q);
alias(j,k);

parameters phi(i) minimum amount of data to be transferred to user i
in the problem frame sequence
           alpha(i);
* actual demand parameter phi
phi(i) = min (nFrames * s(i), d(i));
* minimum number of frames to be assigned to user i
alpha(i)= ceil( phi(i) / A);

binary variables

z_ij(i,j) showing whether user i is assigned a rectangle in frame j
or not
delta(p,q,j) for choosing the users allocated in the same frame
gamma(p,q,j,dim) used in location overlapping constraints
sigma(i,j) for connecting r_ij and lambda_i (which are used for
maximum delay constraints)
;

integer variables

x_ij(i,j) x-coordinate of the left-bottom corner of the rectangle
assigned to user i in frame j
```

```
y_ij(i,j) y-coordinate of the left-bottom corner of the rectangle
assigned to user i in frame j
w_ij(i,j) width of the rectangle allocated to user i in frame j
h_ij(i,j) height of the rectangle allocated to user i in frame j
a_ij(i,j) area of the rectangle allocated to user i in frame j
r_ij(i,j) total remaining demand for user i after frame j;

free variables

z_srp objective function variable for the sequential rectangular
packing problem
;

equations

obj objective function definition - for minimizing partial packing
areas(i,j) nonlinear area equalities for each frame
demands(i) demand constraints for each user through the frame
sequence
simpleFeas feasibility check with respect to total frame area

firstFrame(i,j) total remaining demand for each user i after frame 1
otherFrames(i,j) total remaining demand for each user i after next
frames

* sigma variable definitions
rij_sigmaLower(i,j) defining sigma variables for r_ij with LB
rij_sigmaUpper(i,j) defining sigma variables for r_ij with UB
rij_zij_lambda_relation(i,j) relating rij and zij variables with
maximmum delay parameters

xWidth(i,j) rectangle position (also dimension) constraints for each
user & frame on the horizontal axis
yHeight(i,j) rectangle position (also dimension) constraints for
each user & frame on the vertical axis
assignedFrames(i) using parameters alpha's as lower bounds for
assigned frames for user i
thetaBounds(i) for forcing the transfer beginning frame for user i

* linking constraints for variables z_ij between x_ij, y_ij, w_ij
and h_ij
z0xBinding(i,j) x_ij lower bounds
z1xBinding(i,j) x_ij upper bounds
z0yBinding(i,j) y_ij lower bounds
z1yBinding(i,j) y_ij upper bounds
z0wBinding(i,j) w_ij lower bounds
z1wBinding(i,j) w_ij upper bounds
z0hBinding(i,j) h_ij lower bounds
z1hBinding(i,j) h_ij upper bounds

* Location overlapping constraints for rectangles in each frame,
* (a) for choosing the users allocated in the same frame
differentFrame(p,q,j) lower bound: at least one user is not
allocated on frame j
sameFrame(p,q,j) upper bound: both users p and q may be allocated on
frame j

* (b) relative positions on the horizontal axis
gammaLeft(p,q,j) user p on the left of user q if gamma_1 = 1
```

```
gammaRight(p,q,j) user p on the right of user q if gamma_2 = 1
gammaLeftRight(p,q,j) p cannot be both on the left and right of user
q if both users are on the same frame

* (c) positions on the vertical axis
gammaBelow(p,q,j) user p below the user q if gamma_3 = 1
gammaAbove(p,q,j) user p above the user q if gamma_4 = 1
gammaBelowAbove(p,q,j) p cannot be both below and above the user q
if both users are on the same frame

* linking the logical constraints (a), (b) and (c) above
gammaLeftBelow(p,q,j)
gammaRightAbove(p,q,j)
allGammas(p,q,j)
;

* CONSTRAINT DEFINITIONS
obj.. z_srp =e= sum ((i,j),z_ij(i,j)) ;

areas(i,j).. a_ij(i,j) =e= w_ij(i,j)*h_ij(i,j);

* demand constraints for each user through the frame sequence
demands(i).. sum(j, a_ij(i,j)) =g= phi(i);
*feasibility check with respect to total frame area
simpleFeas.. sum(i,phi(i)) =l= nFrames*A;
* forcing the beginning of ongoing transfers from previous sequences
thetaBounds(i).. sum (j$(ord(j) <= theta(i)),z_ij(i,j)) =g= 1;

* rectangle position (also dimension) constraints for each user &
frame
xWidth(i,j)..  x_ij(i,j) + w_ij(i,j) =l= W;
yHeight(i,j).. y_ij(i,j) + h_ij(i,j) =l= H;

* using parameters alpha i's as lower bounds
assignedFrames(i).. sum(j,z_ij(i,j)) =g= alpha(i);

* linking constraints for variables z_ij between x_ij, y_ij, w_ij
and h_ij
z0xBinding(i,j).. z_ij(i,j) - 1 =l= x_ij(i,j);
z1xBinding(i,j).. x_ij(i,j) =l= (W-1)* z_ij(i,j);
z0yBinding(i,j).. z_ij(i,j) - 1 =l= y_ij(i,j);
z1yBinding(i,j).. y_ij(i,j) =l= (H-1)* z_ij(i,j);
z0wBinding(i,j).. z_ij(i,j) =l= w_ij(i,j);
z1wBinding(i,j).. w_ij(i,j) =l= W* z_ij(i,j);
z0hBinding(i,j).. z_ij(i,j) =l= h_ij(i,j);
z1hBinding(i,j).. h_ij(i,j) =l= H* z_ij(i,j);

* Location overlapping constraints for rectangles in each frame,
* (a) for choosing the users allocated in the same frame
differentFrame(p,q,j)$(ord(p)<ord(q)).. z_ij(p,j) + z_ij(q,j) =l=
delta(p,q,j) + 1;
sameFrame(p,q,j)$(ord(p)<ord(q)).. z_ij(p,j) + z_ij(q,j) =g=
2*delta(p,q,j);

* (b) relative positions on the horizontal axis
gammaLeft(p,q,j)$(ord(p)<ord(q)).. x_ij(p,j) + w_ij(p,j) - x_ij(q,j)
=l= (1-gamma(p,q,j,"1"))*W;
gammaRight(p,q,j)$(ord(p)<ord(q)).. x_ij(q,j) + w_ij(q,j) -
x_ij(p,j) =l= (1-gamma(p,q,j,"2"))*W;
```

```
gammaLeftRight(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"2") =l= 2 - delta(p,q,j);

* (c) positions on the vertical axis
gammaBelow(p,q,j)$(ord(p)<ord(q)).. y_ij(p,j) + h_ij(p,j) -
y_ij(q,j) =l= (1-gamma(p,q,j,"3"))*H;
gammaAbove(p,q,j)$(ord(p)<ord(q)).. y_ij(q,j) + h_ij(q,j) -
y_ij(p,j) =l= (1-gamma(p,q,j,"4"))*H;
gammaBelowAbove(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"3") +
gamma(p,q,j,"4") =l= 2 - delta(p,q,j);

* linking the logical constraints (a), (b) and (c) above
gammaLeftBelow(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"3") =l= 2 * (1 - z_ij(p,j) + delta(p,q,j));
gammaRightAbove(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"2") +
gamma(p,q,j,"4") =l= 2 * (1 - z_ij(q,j) + delta(p,q,j));
allGammas(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"2") + gamma(p,q,j,"3") + gamma(p,q,j,"4") =g=
delta(p,q,j);

* PhD Progress 5 2011-12 Ch.3 - corrections for total remaining
demand
* 7.7.2012 ideas & corrections UE
firstFrame(i,j)$(ord(j) = 1).. r_ij(i,j) =e= phi(i)-a_ij(i,j);
otherFrames(i,j)$(ord(j) > 1)..  r_ij(i,j) =e= r_ij(i,j-1)-
a_ij(i,j);

* maximum delay constraints (10.07.2012 ideas)
rij_sigmaLower(i,j).. 1 - A*sigma(i,j) =l= r_ij(i,j);
rij_sigmaUpper(i,j).. r_ij(i,j) =l= phi(i)*(1-sigma(i,j));
rij_zij_lambda_relation(i,j)$(ord(j) <= nFrames-lambda(i)).. sum(k $
(ord(k) > ord(j) and ord(k) <= ord(j)+lambda(i)), z_ij(i,k)) =g=
z_ij(i,j)-sigma(i,j);

* Variable Bounds
* bounding r_ij
r_ij.UP(i,j) = phi(i);
r_ij.LO(i,j) = 1-A;
*z_ij.UP(i,j)$(ord(j) > 1) = 1$(r_ij.L(i,j-1)>0);

* Last frame remaining
r_ij.UP(i,j)$(ord(j) = card(j)) = 0;

MODEL SeqRectPack /ALL/;

OPTION optca=1e-5, optcr=1e-5, minlp=scip, reslim=600,
iterlim=2000000;

scalar starttime; starttime = jnow

SOLVE SeqRectPack USING MINLP MINIMIZING z_srp;
execerror=0;

file textOutput /n10m2a1PR02.gout/;
textOutput.pc=6;

put textOutput, SeqRectPack.modelstat:0:0, SeqRectPack.solvestat:0:0
/;
```

```
put SeqRectPack.objVal:0:0, SeqRectPack.resUsd:0:4,
SeqRectPack.nodUsd:0:0 /;
put SeqRectPack.numVar:0:0, SeqRectPack.numEqu:0:0,
SeqRectPack.numNZ:0:0, SeqRectPack.numNLNZ:0:0 /;
put card(i):0:0, card(j):0:0, W:0:0, H:0:0, sum(i, phi(i)):0:0 /;

loop(i, put ord(i):0:0, sum(j, z_ij.l(i,j)):0:0, phi(i):0:0, sum(j,
a_ij.l(i,j)):0:0 /;
         loop(j $ z_ij.l(i,j),
put ord(j):4:0, x_ij.l(i,j):0:0, y_ij.l(i,j):0:0, w_ij.l(i,j):0:0,
h_ij.l(i,j):0:0;
put /;);
);
```

A.3 Sample GAMS Code for an instance of SRP-III

GAMS code of the SRP-III model for a problem instance is provided in this appendix. Note that the below code belongs to the problem instance whose output was depicted in Figure 4.3.

```
$title Sequential Rectangular Packing (SRP) - 3-Minimizing
Overallocation
Sets    i users  / 1*10 /
         j frames  / 1*4 /
         dim dimensions used in overlapping constraints / 1*4 /;
Scalars W frame width / 12 /
         H frame height / 30 /
         nUsers / 10 /
         nFrames / 4 /
;
Parameters  d(i)  total amount of data request of user i
         /1     164,2   129,3   97,4    188,5   23,6    33,7    72,8
74,9    187,10  55/
          s(i)  minimum data transfer rate of user i per frame (QoS
parameter)
         /1     164,2   129,3   97,4    188,5   23,6    33,7    72,8
74,9    187,10  55/
          lambda(i) maximum delay period for user i (QoS parameter)
         /1     2,2     2,3     2,4     2,5     6,6     2,7     3,8
5,9     2,10    2/
          theta(i)     latest frame to maintain or to begin the data
transfer for user i
         /1     5,2     5,3     5,4     5,5     5,6     5,7     5,8
5,9     5,10    5/;

scalar  A frame area;
A =  W*H;

alias(i,p,q);
alias(j,k);

parameters phi(i) minimum amount of data to be transferred to user i
in the problem frame sequence
           alpha(i);
* actual demand parameter phi
phi(i) = min (nFrames * s(i), d(i));
* minimum number of frames to be assigned to user i
alpha(i)= ceil( phi(i) / A);

*scalar starttime; starttime = jnow

binary variables

z_ij(i,j) showing whether user i is assigned a rectangle in frame j
or not
delta(p,q,j) for choosing the users allocated in the same frame
gamma(p,q,j,dim) used in location overlapping constraints
```

```
sigma(i,j) for connecting r_ij and lambda_i (which are used for
maximum delay constraints)
;

integer variables

x_ij(i,j) x-coordinate of the left-bottom corner of the rectangle
assigned to user i in frame j
y_ij(i,j) y-coordinate of the left-bottom corner of the rectangle
assigned to user i in frame j
w_ij(i,j) width of the rectangle allocated to user i in frame j
h_ij(i,j) height of the rectangle allocated to user i in frame j
a_ij(i,j) area of the rectangle allocated to user i in frame j
r_ij(i,j) total remaining demand for user i after frame j;

free variables

waste total surplus area after the assigned sequence of frames
;

equations

* the objective value for total wasted (surplus) area used
wastedArea total area assigned after the last frame
areas(i,j) nonlinear area equalities for each frame
demands(i) demand constraints for each user through the frame
sequence
simpleFeas feasibility check with respect to total frame area

firstFrame(i,j) total remaining demand for each user i after frame 1
otherFrames(i,j) total remaining demand for each user i after next
frames

* sigma variable definitions
rij_sigmaLower(i,j) defining sigma variables for r_ij with LB
rij_sigmaUpper(i,j) defining sigma variables for r_ij with UB
rij_zij_lambda_relation(i,j) relating rij and zij variables with
maximmum delay parameters
*for avoiding maximum delay violation after no demand remaining
*rij_zij_trivial(i,j) for preventing unnecessary assignment after
all demand is satisfied

xWidth(i,j) rectangle position (also dimension) constraints for each
user & frame on the horizontal axis
yHeight(i,j) rectangle position (also dimension) constraints for
each user & frame on the vertical axis
assignedFrames(i) using parameters alpha's as lower bounds for
assigned frames for user i
thetaBounds(i) for forcing the transfer beginning frame for user i

* linking constraints for variables z_ij between x_ij, y_ij, w_ij
and h_ij
z0xBinding(i,j) x_ij lower bounds
z1xBinding(i,j) x_ij upper bounds
z0yBinding(i,j) y_ij lower bounds
z1yBinding(i,j) y_ij upper bounds
z0wBinding(i,j) w_ij lower bounds
z1wBinding(i,j) w_ij upper bounds
z0hBinding(i,j) h_ij lower bounds
```

```
z1hBinding(i,j) h_ij upper bounds

* Location overlapping constraints for rectangles in each frame,
* (a) for choosing the users allocated in the same frame
differentFrame(p,q,j) lower bound: at least one user is not
allocated on frame j
sameFrame(p,q,j) upper bound: both users p and q may be allocated on
frame j

* (b) relative positions on the horizontal axis
gammaLeft(p,q,j) user p on the left of user q if gamma_1 = 1
gammaRight(p,q,j) user p on the right of user q if gamma_2 = 1
gammaLeftRight(p,q,j) p cannot be both on the left and right of user
q if both users are on the same frame

* (c) positions on the vertical axis
gammaBelow(p,q,j) user p below the user q if gamma_3 = 1
gammaAbove(p,q,j) user p above the user q if gamma_4 = 1
gammaBelowAbove(p,q,j) p cannot be both below and above the user q
if both users are on the same frame

* linking the logical constraints (a), (b) and (c) above
gammaLeftBelow(p,q,j)
gammaRightAbove(p,q,j)
allGammas(p,q,j)
;

* CONSTRAINT DEFINITIONS
* the objective function value for total wasted (surplus) area
assigned to users
wastedArea.. waste =e= -sum((i,j)$(ord(j) = card(j)),r_ij(i,j));
areas(i,j).. a_ij(i,j) =e= w_ij(i,j)*h_ij(i,j);

* demand constraints for each user through the frame sequence
demands(i).. sum(j, a_ij(i,j)) =g= phi(i);
*feasibility check with respect to total frame area
simpleFeas.. sum(i,phi(i)) =l= nFrames*A;
* forcing the beginning of ongoing transfers from previous sequences
thetaBounds(i).. sum (j$(ord(j) <= theta(i)),z_ij(i,j)) =g= 1;

* rectangle position (also dimension) constraints for each user &
frame
xWidth(i,j)..  x_ij(i,j) + w_ij(i,j) =l= W;
yHeight(i,j).. y_ij(i,j) + h_ij(i,j) =l= H;

* using parameters alpha i's as lower bounds
assignedFrames(i).. sum(j,z_ij(i,j)) =g= alpha(i);

* linking constraints for variables z_ij between x_ij, y_ij, w_ij
and h_ij
z0xBinding(i,j).. z_ij(i,j) - 1 =l= x_ij(i,j);
z1xBinding(i,j).. x_ij(i,j) =l= (W-1)* z_ij(i,j);
z0yBinding(i,j).. z_ij(i,j) - 1 =l= y_ij(i,j);
z1yBinding(i,j).. y_ij(i,j) =l= (H-1)* z_ij(i,j);
z0wBinding(i,j).. z_ij(i,j) =l= w_ij(i,j);
z1wBinding(i,j).. w_ij(i,j) =l= W* z_ij(i,j);
z0hBinding(i,j).. z_ij(i,j) =l= h_ij(i,j);
z1hBinding(i,j).. h_ij(i,j) =l= H* z_ij(i,j);
```

```
* Location overlapping constraints for rectangles in each frame,
* (a) for choosing the users allocated in the same frame
differentFrame(p,q,j)$(ord(p)<ord(q)).. z_ij(p,j) + z_ij(q,j) =l=
delta(p,q,j) + 1;
sameFrame(p,q,j)$(ord(p)<ord(q)).. z_ij(p,j) + z_ij(q,j) =g=
2*delta(p,q,j);

* (b) relative positions on the horizontal axis
gammaLeft(p,q,j)$(ord(p)<ord(q)).. x_ij(p,j) + w_ij(p,j) - x_ij(q,j)
=l= (1-gamma(p,q,j,"1"))*W;
gammaRight(p,q,j)$(ord(p)<ord(q)).. x_ij(q,j) + w_ij(q,j) -
x_ij(p,j) =l= (1-gamma(p,q,j,"2"))*W;
gammaLeftRight(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"2") =l= 2 - delta(p,q,j);

* (c) positions on the vertical axis
gammaBelow(p,q,j)$(ord(p)<ord(q)).. y_ij(p,j) + h_ij(p,j) -
y_ij(q,j) =l= (1-gamma(p,q,j,"3"))*H;
gammaAbove(p,q,j)$(ord(p)<ord(q)).. y_ij(q,j) + h_ij(q,j) -
y_ij(p,j) =l= (1-gamma(p,q,j,"4"))*H;
gammaBelowAbove(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"3") +
gamma(p,q,j,"4") =l= 2 - delta(p,q,j);

* linking the logical constraints (a), (b) and (c) above
gammaLeftBelow(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"3") =l= 2 * (1 - z_ij(p,j) + delta(p,q,j));
gammaRightAbove(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"2") +
gamma(p,q,j,"4") =l= 2 * (1 - z_ij(q,j) + delta(p,q,j));
allGammas(p,q,j)$(ord(p)<ord(q)).. gamma(p,q,j,"1") +
gamma(p,q,j,"2") + gamma(p,q,j,"3") + gamma(p,q,j,"4") =g=
delta(p,q,j);

* PhD Progress 5 2011-12 Ch.3 - corrections for total remaining
demand
* 7.7.2012 ideas & corrections UE
firstFrame(i,j)$(ord(j) = 1).. r_ij(i,j) =e= phi(i)-a_ij(i,j);
otherFrames(i,j)$(ord(j) > 1)..  r_ij(i,j) =e= r_ij(i,j-1)-
a_ij(i,j);

* maximum delay constraints (10.07.2012 ideas)
rij_sigmaLower(i,j).. 1 - A*sigma(i,j) =l= r_ij(i,j);
rij_sigmaUpper(i,j).. r_ij(i,j) =l= phi(i)*(1-sigma(i,j));
rij_zij_lambda_relation(i,j)$(ord(j) <= nFrames-lambda(i)).. sum(k $
(ord(k) > ord(j) and ord(k) <= ord(j)+lambda(i)), z_ij(i,k)) =g=
z_ij(i,j)-sigma(i,j);
*for avoiding maximum delay violation after no demand remaining
*rij_zij_trivial(i,j)$(ord(j) > 1).. z_ij(i,j) =l= (1-sigma(i,j-1));

* Variable Bounds
* bounding r_ij
r_ij.UP(i,j) = phi(i);
r_ij.LO(i,j) = 1-A;
*z_ij.UP(i,j)$(ord(j) > 1) = 1$(r_ij.L(i,j-1)>0);

* Last frame remaining
r_ij.UP(i,j)$(ord(j) = card(j)) = 0;

MODEL SeqRectPack /ALL/;
*SeqRectPack.workspace=100;
```

```
*SeqRectPack.optfile=1;

* instruct BARON to give higher branching priorities
*$onecho > baron.opt
*NLPSol 2
*a_ij.prior 40
*z_ij.prior 10
*$offecho

OPTION optca=1e-5, optcr=1e-5, minlp=scip, reslim=600,
iterlim=2000000;

scalar starttime; starttime = jnow

SOLVE SeqRectPack USING MINLP MINIMIZING waste;
execerror=0;

file textOutput /Figure4.7-input-tr1n10m4PR08.gout/;
textOutput.pc=6;

put textOutput, SeqRectPack.modelstat:0:0, SeqRectPack.solvestat:0:0
/;
put SeqRectPack.objVal:0:0, SeqRectPack.resUsd:0:4,
SeqRectPack.nodUsd:0:0 /;
put SeqRectPack.numVar:0:0, SeqRectPack.numEqu:0:0,
SeqRectPack.numNZ:0:0, SeqRectPack.numNLNZ:0:0 /;
put card(i):0:0, card(j):0:0, W:0:0, H:0:0, sum(i, phi(i)):0:0 /;

loop(i, put ord(i):0:0, sum(j, z_ij.l(i,j)):0:0, phi(i):0:0, sum(j,
a_ij.l(i,j)):0:0 /;
         loop(j $ z_ij.l(i,j),
put ord(j):4:0, x_ij.l(i,j):0:0, y_ij.l(i,j):0:0, w_ij.l(i,j):0:0,
h_ij.l(i,j):0:0;
put /;);
);
```

APPENDIX B
EXAMPLE INPUT AND OUTPUT FILES

B.1 Configuration File for Problem Generation and Solutions

The following XML .config file developed in Visual Studio 2010 IDE is used for the generation of the input files for the problem instances used in computational experimentation, and solution of these problem instances by any available mixed integer nonlinear programming solver with GAMS. Namely, the same executable can be used without any additional modification or recompilation for different purposes and problem types, merely by changing the respective parameters presented below.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <!-- m: # of frames; n: # of users -->
    <!-- settings for problem input file generation mechanism -->
    <!-- the semicolons ";" seperate level values -->
    <!-- the hyphens "-" seperate lower & upper bound values for the
corresponding levels -->
    <add key="userLevel" value="10;20;40"/>
    <add key="frameSeqLengthLevel" value="2;4;8"/>
    <!-- the x seperates width & height sizes for frames -->
    <add key="frameSize" value="12x30"/>
    <!-- if demand level string is empty or not found transferLevel
s_i will be used for d_i= m * s_i -->
    <add key="demandLevel" value="12-192;10-80"/>    <!-- "4-32"-->
    <!-- if above demand values are used, then number of transfer
levels will be the same as d_i levels and are used in parallel -->
    <add key="transferLevel" value="6-24;5-10"/>
    <!-- frameDependentDemandRatioLevel lower & upper bounds are
percentage values for the respective frame area -->
    <!-- if demand level string is empty and
frameDependentDemandRatioLevel are nonempty, the percentage bounds
below will be used -->
    <!-- again as in demand levels above, number of transfer levels
should be the same as number of frameDependentDemandRatioLevels and
are used in parallel-->
    <add key="frameDependentDemandRatioLevel" value=""/>    <!-- "5-
10;5-20"-->
    <add key="profitLevel" value=""/>
    <!-- if profit level string is empty or not found
profitCoefficientLevel will be used -->
    <!-- if profit coefficient pc is used, then p_i= pc * d_i -->
    <add key="profitCoefficientLevel" value="4"/>
    <!-- lambda levels (as frame sequence bounds in percentages)
might be defined here as well -->
    <!-- lambda may be greater than m -->
    <!-- If simulation, these will be actual lambda value intervals
and always be used explicitly -->
    <!-- if lambda level string is empty or not found, lambda
(maximum delay) will be uniformly selected between [1,m] -->
    <add key="lambdaLevel" value="2-2;3-8"/>
```

```
    <!-- probabilities for being an ongoing transfer; if ongoing a
frame index less than or equal to lambda, otherwise frame sequence
length -->
    <!-- if theta level string is empty or not found, all demands
will be for "new" transfers, theta = m+1 -->

    <add key="thetaLevel" value=""/>
    <!-- number of instances to be generated for each setting
combination -->
    <add key="instancePerSetting" value="10"/>
    <!-- file extension that will be used for SRP problem files -->
    <add key="genFileExtension" value="txt"/>

    <!-- settings for GAMS model input and solutions -->
    <add key="gamsExecutable" value="C:\Program
Files\GAMS\23.9\gams.exe"/>
    <add key="gamsModelType" value="5-Maximizing Profit with
Bounds"/>
    <!-- 1-Feasibility; 2-Minimizing Partition; 3-Minimizing
Overallocation; 4-Maximizing Profit; 5-Maximizing Profit with Bounds
-->
    <add key="gamsModelBaseText" value="SRP-I-profit-limit10min-
scip-bothBounds.gtxt"/>
    <add key="gamsOutputFormatText" value="SRP-I-bounded-
output.gtxt"/>
    <add key="gamsOutputExtension" value="gout1"/>

    <!-- more general settings -->
    <add key="inputFileExtension" value="txt2"/>
    <!-- Use 1 for problem generation, 2 for GAMS solution, 3 for
simulation probem generation, etc.-->
    <add key="programUsage" value="3"/>

    <!-- Simulator settings -->
    <!-- If simulation problem generation settings are used, then
all demand, transfer, profit, profit coefficient & lambda related
settings -->
    <!-- should be parallel to the number of traffic classes given
below. Moreover, for profit coefficient the key below should be used
-->
    <!-- Naturally, there should be at least two different classes
for a simulation data generation. -->
    <add key="trafficClass" value="Data;Voice"/>
    <!-- the distribution of classes in percentages, thus each level
should sum up to 100. -->
    <!-- For example if there are 3 classes, then 50-20-30 is valid
-->
    <add key="classDistribution" value="75-25;50-50"/>
    <!-- if no explicit profit level given, the discrete uniform
from the below intervals will be used -->
    <add key="profitCoefficientIntervalLevel" value="1-6;6-12"/>

  </appSettings>
</configuration>
```

B.2 Example Input Files

In this appendix, example input files corresponding to the problem instances in Figures 4.5, 4.6 and 4.7 are provided. Figure B.1 exhibits the input file for the instance in Figure 4.5, involving a problem with 10 users packed over 2 frames with traffic distribution *TR1* (75% of users with data + 25% voice traffic).

```
10      2       12      30
1       28      168     7       28      3       2
2       99      396     2       99      3       1
3       103     515     2       103     3       1
4       190     950     2       190     3       1
5       90      90      2       90      3       1
6       49      294     2       49      3       1
7       78      312     2       78      3       1
8       35      35      2       35      3       1
9       43      258     2       43      3       1
10      42      84      2       42      3       1
0       757     720
2
Data    ->      %75
Voice   ->      %25
```

Figure B.1 Input file of the problem instance in Figure 4.5

In the input files, the first line indicates that there are 10 users to be packed over 2 frames, with width and height dimensions equal to 12 and 30, respectively. The succeeding 10 lines include user data for each user: User index (1) followed by the demand size (28), profit value (168), maximum delay value in frames (7), minimum transfer rate (28), θ_i value, i.e. the latest frame to maintain or to begin the data transfer for user i (3: new transfer) and lastly the user class (2: user with voice traffic).

The first line after the user parameters lines describes the problem instance, as to whether the case $\sum_{i \in I} \phi_i \leq mA$ holds or not. The value of zero means that the inequality does not hold, and the instance can be used for SRP-I solutions. This indicator is followed by the sum of user demands (757) to be packed, and the total area of all frames (720 = 2x12x30). The single entry in the next line (2) denotes the number of user traffic classes involved (Data Set 2), followed by their names and

respective distributions in percentages in the last two lines. The same logic applies for all the sample instance input files.

Figure B.2 exhibits the input file for the instance in Figure 4.6, which involves a problem instance with 10 users over 4 frames with traffic distribution *TR1* (75% of users with data + 25% voice traffic) and an average user demand of 116 slots.

```
10      4       12      30
1       187     748     2       187     5       1
2       39      468     7       39      5       2
3       76      684     6       76      5       2
4       41      205     2       41      5       1
5       189     567     2       189     5       1
6       149     447     2       149     5       1
7       97      485     2       97      5       1
8       26      286     7       26      5       2
9       182     364     2       182     5       1
10      171     684     2       171     5       1
1       1157    1440
2
Data    ->      %75
Voice   ->      %25
```

Figure B.2 Input file of the problem instance in Figure 4.6

As a last example and for comparison, Figure B.3 below exhibits the input file for the instance in Figure 4.7: A problem with 10 users over 4 frames with traffic distribution TR1; with average user demand of 102 slots.

```
10      4       12      30
1       164     820     2       164     5       1
2       129     387     2       129     5       1
3       97      582     2       97      5       1
4       188     940     2       188     5       1
5       23      276     6       23      5       2
6       33      165     2       33      5       1
7       72      432     3       72      5       2
8       74      814     5       74      5       2
9       187     1122    2       187     5       1
10      55      55      2       55      5       1
1       1022    1440
2
Data    ->      %75
Voice   ->      %25
```

Figure B.3 Input file of the problem instance in Figure 4.7

B.3 Example Output Files

In this appendix, example output files corresponding to the problem instances in Figures 4.5, 4.6 and 4.7 are provided. Figure B.4 exhibits the output file for the instance in Figure 4.5.

```
1       1
3018    10.2800 4926    3018
621     1331    4441    40
10      2       12      30      757
9       715     3102    2893    3065
1       2       28      28      168
1       0       0       2       8
2       6       7       3       4
2       2       99      99      396
1       2       12      7       6
2       6       11      3       19
3       2       103     103     515
1       2       18      8       12
2       11      0       1       7
4       2       190     192     950
1       2       0       8       12
2       0       14      6       16
5       2       90      90      90
1       9       12      1       6
2       0       0       6       14
6       1       49      50      294
1       10      0       2       25
7       2       78      79      312
1       10      25      2       5
2       9       7       3       23
8       1       35      35      35
2       6       0       5       7
9       1       43      44      258
1       0       8       2       22
```

Figure B.4 Output file of the problem instance in Figure 4.5

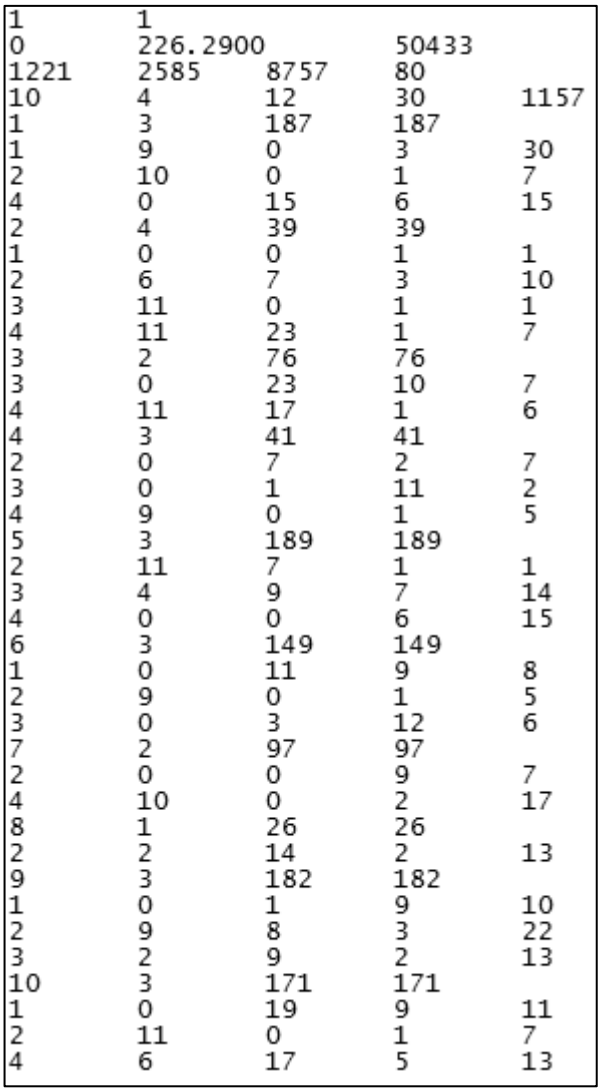
The output file given above belongs to an SRP-I solution employing lower and upper bounds. The first line shows the GAMS status messages, describing whether the solution is optimal or not, and lists its termination condition (normal, time limit, user interrupt, etc.). The next line values are respectively the objective function value, solution time in seconds, number of branch & bound nodes searched, and the last upper bound used by the solver.

The third line is regarding the problem size, and lists respectively the number of decision variables, number of constraints, number of nonzero entries in the coefficient matrix and number of nonlinear nonzeroes belonging to the model. The

fourth line describes the instance briefly: Number of users, frames, frame width and height, and lastly the total demand size in slots of all users.

The fifth line is SRP-I specific, and lists the number of users packed, total demand size of these users, total available profit of all users, and lastly the initial lower (2893) and upper bounds (3065) used. The following lines are concerning the users packed in the SRP-I solution, and ordered according to their indices. So, the next line first gives the user index (1), the number of frames (2) it is packed in, the demand of the user in slots (28), the actual area allocated for the user (28), and the profit gained. The next two lines are the respective frame indices, positions and sizes of the rectangles assigned to these frames. Hence, for user 1, the first of its two rectangles is placed in frame 1, its bottom-left x-y coordinate is (0,0) with width and height equal to 2 and 8, respectively. The remaining lines follow in the same manner.

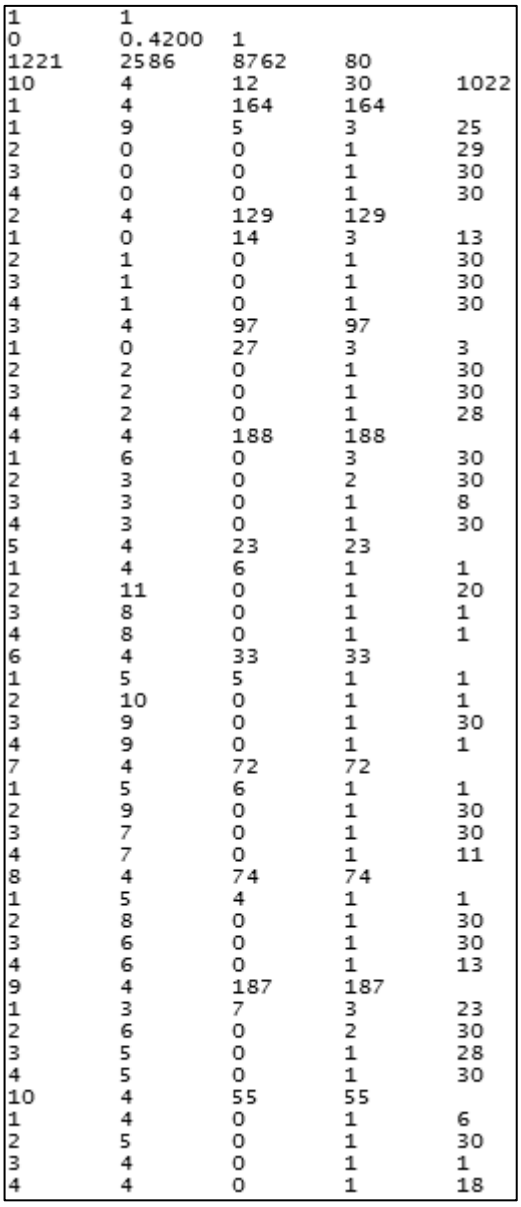
Figure B.5 exhibits the output file for the instance in Figure 4.6. This is not a solution of an SRP-I instance, thus the fifth line explained above is not present in this output. This is because, if there is a feasible or optimal solution, all users are already packed. Moreover, the user profit values are not listed in the corresponding user lines for SRP-II and SRP-III problem solutions.



```
1       1
0       226.2900        50433
1221    2585    8757    80
10      4       12      30      1157
1       3       187     187
1       9       0       3       30
2       10      0       1       7
4       0       15      6       15
2       4       39      39
1       0       0       1       1
2       6       7       3       10
3       11      0       1       1
4       11      23      1       7
3       2       76      76
3       0       23      10      7
4       11      17      1       6
4       3       41      41
2       0       7       2       7
3       0       1       11      2
4       9       0       1       5
5       3       189     189
2       11      7       1       1
3       4       9       7       14
4       0       0       6       15
6       3       149     149
1       0       11      9       8
2       9       0       1       5
3       0       3       12      6
7       2       97      97
2       0       0       9       7
4       10      0       2       17
8       1       26      26
2       2       14      2       13
9       3       182     182
1       0       1       9       10
2       9       8       3       22
3       2       9       2       13
10      3       171     171
1       0       19      9       11
2       11      0       1       7
4       6       17      5       13
```

Figure B.5 Output file of the problem instance in Figure 4.6

As a last example, Figure B.6 below exhibits the output file for the instance in Figure 4.7.



1	1			
0	0.4200	1		
1221	2586	8762	80	
10	4	12	30	1022
1	4	164	164	
1	9	5	3	25
2	0	0	1	29
3	0	0	1	30
4	0	0	1	30
2	4	129	129	
1	0	14	3	13
2	1	0	1	30
3	1	0	1	30
4	1	0	1	30
3	4	97	97	
1	0	27	3	3
2	2	0	1	30
3	2	0	1	30
4	2	0	1	28
4	4	188	188	
1	6	0	3	30
2	3	0	2	30
3	3	0	1	8
4	3	0	1	30
5	4	23	23	
1	4	6	1	1
2	11	0	1	20
3	8	0	1	1
4	8	0	1	1
6	4	33	33	
1	5	5	1	1
2	10	0	1	1
3	9	0	1	30
4	9	0	1	1
7	4	72	72	
1	5	6	1	1
2	9	0	1	30
3	7	0	1	30
4	7	0	1	11
8	4	74	74	
1	5	4	1	1
2	8	0	1	30
3	6	0	1	30
4	6	0	1	13
9	4	187	187	
1	3	7	3	23
2	6	0	2	30
3	5	0	1	28
4	5	0	1	30
10	4	55	55	
1	4	0	1	6
2	5	0	1	30
3	4	0	1	1
4	4	0	1	18

Figure B.6 Output file of the problem instance in Figure 4.7

Finally, we include in Figure B.7 below a screen shot of the MS Excel output for the instance in Figure 4.7. This output is also generated for each instance automatically by the Excel macro procedures developed in Visual Basic language.

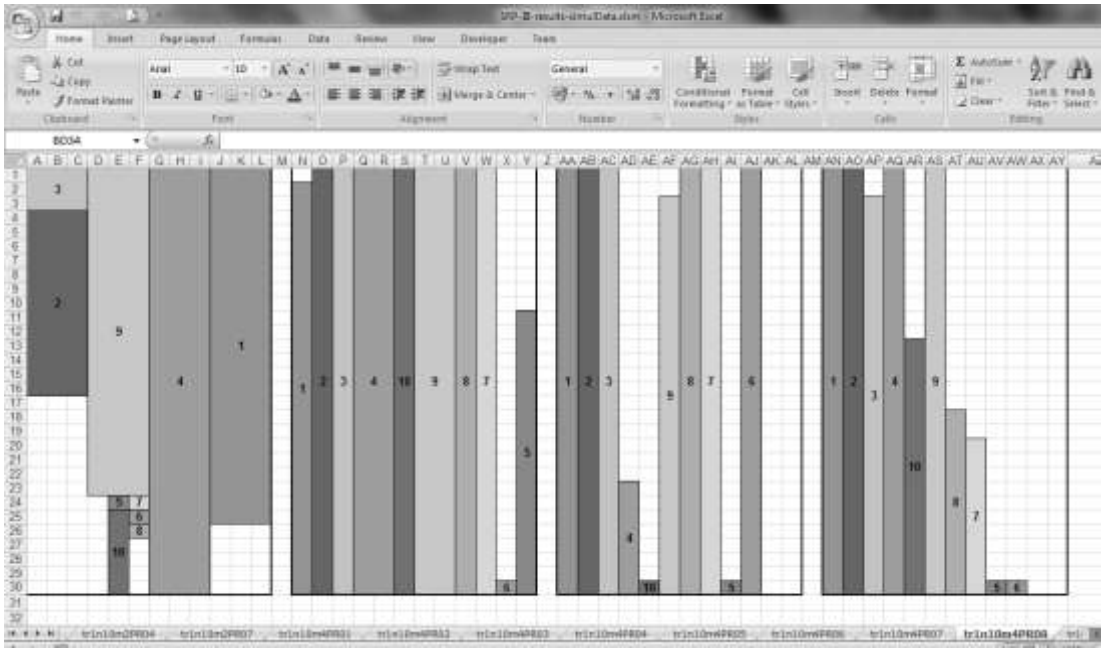


Figure B.7 MS Excel output screenshot for the problem instance in Figure 4.7

APPENDIX C

C# CODES FOR LOWER AND UPPER BOUNDS OF SRP-I

C.1 Code segment for *AlgorithmUB*

The code segment below, which is a part of the SRP-I code library, is used for obtaining the upper bound for the SRP-I problem.

```
private int KnapsackRelaxationUpperBound()
        {
            int totalArea= frames * frameWidth * frameHeight;
            if (totalDemand<totalArea) return availableProfit;
            // sorted in nondecreasing order (profit/demand)
            int[] sortedUsersProfitPerDemand =
Sorter.OriginalIndexSorted(profit_per_demand);
            int profitBound = 0;
            int user=users-1; // zero-indexed, last has the highest
profit/demand ratio

            while ((totalArea > 0) && (user >= 0))
            {
                int newPack = phi_local[sortedUsersProfitPerDemand[user]];
                if (totalArea - phi_local[sortedUsersProfitPerDemand[user]] <
0)
                {
                    double partialProfit=
(double)profit[sortedUsersProfitPerDemand[user]]* totalArea/newPack;
                    profitBound += (int)Math.Ceiling(partialProfit);
                    return profitBound;
                } else
                   {
                    profitBound += profit[sortedUsersProfitPerDemand[user]];
                    totalArea -= newPack;
                   }
                user--;
            }

            return profitBound;
        }
```

C.2 Code segment for *AlgorithmLB*

The code segment below, which is a part of the SRP-I code library, is used for obtaining the lower bound for the SRP-I problem.

```
private int SRP_I_LowerBound()
        {
            int frameArea= frameWidth * frameHeight;
            // sorted in nondecreasing order (profit/demand)
            int[] sortedUsersProfitPerDemand =
Sorter.OriginalIndexSorted(profit_per_demand);
            int profitBound = 0;
            int user=users-1; // zero-indexed, last has the highest
profit/demand ratio
            // choosing the shortest edge of the frame for less
overallocation, thus tighter bound
            int shortEdge=frameWidth;
            int longEdge=frameHeight;
            if (frameWidth > frameHeight) {
                shortEdge=frameHeight;
                longEdge=frameWidth;
            }
            // filling the frame by short edges, and ignoring unused spaces if
there is overalocation;
            // so if short edge=5 and demandPerFrame is 8, 8/5 ~ 2 short edges
used, and remainingStrip becomes (remainingStrip - 2)
            int remainingStrip=longEdge;

            while ((remainingStrip > 0) && (user >= 0)) {
                int newPack = phi_per_frame[sortedUsersProfitPerDemand[user]];
                if (newPack> remainingStrip * shortEdge)  // try the next user
                {
                    user--;
                    if (user < 0) break;

                } else
                   {
                    int
usedStrip=(int)Math.Ceiling((double)phi_per_frame[sortedUsersProfitPerDemand[u
ser]]/shortEdge);
                    profitBound += profit[sortedUsersProfitPerDemand[user]];
                    remainingStrip -= usedStrip;
                   }
                user--;
            }

            return profitBound;
        }
```