Published in the *Journal of the Operational Research Society* 51:1083-1092 (2000).
© Palgrave Macmillan, doi:10.1057/palgrave.jors.2601011

Packing Problems and Project Scheduling Models: An Integrating Perspective

Sönke Hartmann*

Christian-Albrechts-Universität zu Kiel, Lehrstuhl für Produktion und Logistik, D-24098 Kiel, Germany. E-mail: hartmann@bwl.uni-kiel.de

*supported by the *Studienstiftung des deutschen Volkes*

Abstract. This paper considers two problem classes that are important to researchers as well as practitioners, namely packing and project scheduling problems. The two problem categories are described, and a classification is given for the different kinds of packing problems and project scheduling concepts. While both problem classes are different with respect to their fields of application, similarities of their mathematical structures are examined. It is shown that all packing problems considered here are special cases of models for project scheduling. The aim is to indicate which project scheduling models can be used to capture the different types of packing problems. Finally, some implications for research on optimization algorithms for these two problem classes are discussed, and the applicability of the results of this work in practice is pointed out.

Keywords. Packing, Cutting, Project Management, Scheduling, Modeling.

1 Introduction

Packing problems consider the task of packing small items (e.g., boxes) into large objects (e.g., containers). Depending on their field of application, several types are distinguished. Important classes include the problem of packing as many items as possible into a fixed number of containers and that of packing all given items into the smallest possible number of containers. Packing problems occur with one, two, or three (relevant) spatial dimensions. Often further restrictions such as weight limitations of the containers have to be observed. With many applications in logistics, packing problems are important to practitioners as well as to researchers. For overviews of packing problems, we refer to Dyckhoff [9] and Wottawa [31].

Project scheduling is an important part of project management. A project consists of activities which must be carried out to achieve a predefined goal. When scheduling a project, certain requirements must be observed, especially limited availabilities of resources that are needed to perform the activities. Many models have been suggested that cover different requirements occuring in practice, e.g., different objectives and the possibility to capture alternative ways to accomplish an activity. Projects can be found in various areas such as construction, production planning, and research and development. A large number of software systems for project management have been developed. Moreover, project scheduling problems have attracted many researchers. For surveys, we refer to Brucker et al. [3] and Özdamar and Ulusoy [20].

Obviously, packing and project scheduling problems are completely different with respect to their applications. Nevertheless, one may be interested in the underlying structures. This paper compares the mathematical properties of packing and project scheduling problems. Our goal is to show that many packing problems are, from a mathematical point of view, special cases of project scheduling models. We discuss which structures of project scheduling models subsume which types of packing problems, and we also point out how these theoretical insights can be used in practice.

Indentifying an optimization problem as a special case of another problem is an important issue in research on combinatorial optimization. The benefit of dealing with this is twofold: First and more important, this allows to transfer solution approaches from one problem to the other: If a promising algorithm for the special case exists, the underlying concepts can probably be extended to the more general problem. On the other hand, a procedure for the general problem can be applied to the special case as well, eventually by additionally exploiting the special problem structure of the latter. In fact, we will point out how our theoretical results can be used to construct new heuristics of pactical relevance. Second, knowledge about relations between different problems is useful when establishing complexity proofs for specific problems because the core of a complexity proof is the polynomial-time transformation of one problem into another. Loosely speaking, if we know that some problem is NP-hard (and a few more technical assumptions are fulfilled), then a more general problem is NP-hard as well (cf. Garey and Johnson [15]).

With these points in mind, it is no surprise that the analysis of similarities between different problem structures plays an important role in the literature. The following list of references shows that the relationships of a broad variety of combinatorial optimization problems on the one hand and packing or project scheduling problems on the other hand have been examined. Demeulemeester and Herroelen [6] show how concepts from production planning problems such as setup times and batches can be modeled using the the standard resource-constrained project scheduling problem (RCPSP) generalized by minimal time lags, release dates, deadlines, and resource availability varying with time. Dyckhoff [9] notes that many different optimization problems such as assembly line balancing, multiprocessor scheduling, capital budgeting, coin changing, and memory allocation for data storage can be viewed as special cases of cutting and packing problems. Garey et al. [14] mention that their "resource-constrained scheduling problem" (which is an RCPSP where the durations of the activities are equal to one) subsumes the bin packing problem. Drexl and Salewski [8] express a school timetabling problem using project scheduling concepts including multiple modes, mode identity, partially renewable resources, minimal time lags, and a cost based objective. Scholl et al. [22] mention that the one-dimensional bin packing problem with the objective of minimizing the bin capacity is equivalent to minimizing the makespan of independent jobs on identical parallel processors. Based on the ideas given in Schrage [23], Sprecher [26] formally describes how the job shop scheduling problem is included in the RCPSP as a special case. Moreover, Sprecher [26] shows that the flow shop problem and the open shop problem are special cases of the RCPSP. Schrage [23] additionally states that the RCPSP includes a variant of the two-dimensional cutting stock problem. Sprecher [25] as well as de Reyck and Herroelen [5] consider the assembly line balancing problem. Sprecher [25] transforms it into an RCPSP with time-varying resource capacity while de Reyck and Herroelen [5] model it as an RCPSP with start-start precedence constraints (but with constant resource capacity).

2 Packing Problem Types

Generally speaking, a packing problem consists of packing small objects (e.g., boxes) into one or more large objects (e.g., containers) with respect to a given objective. We make a few general assumptions on the packing problems considered here. First, we restrict ourselves to objects of rectangular shape. Second, we consider only orthogonal packing patterns, that is, the edges of the small objects to be packed must be either parallel or orthogonal to the edges of the large objects. To keep the description simple, we will refer to the small objects as boxes and to the large objects as containers.

This section briefly defines four basic types of packing problems. The classification is based on that of Wottawa [31]. Moreover, several extensions with high practical relevance are summarized.

Finally, we mention what distinguishes packing problems from a related problem class, the cutting stock problems.

2.1 Bin Packing

In the bin packing problem, a number of boxes is given. For each box, its size is given. The boxes are to be packed into containers for which the size is given as well. It is assumed that the containers are identical, that is, they are of the same size. The objective is to use the minimum number of containers to pack all boxes, such that for each container the size is observed. A packing pattern consists of an assignment of boxes to containers, along with coordinates in all relevant dimensions for each box within the assigned container.

In the one-dimensional version of the bin packing problem, we have B boxes with length l_b, $b = 1, \dots, B$, which have to be packed into a minimum number of containers all of which are L units long. We remark here that also many other interpretations (and thus applications) are possible. An example is to view L as the limited weight capacity of the containers and l_b as the weight of box b, given that space is not scarce.

The two-dimensional bin packing problem considers B boxes with length l_b and width w_b, $b = 1, \dots, B$. They have to be packed into identical containers with length L and width W. Note that in the two-dimensional case, the actual application may allow a rotation of the boxes. Formally, rotation of a box b can be thought of as swapping its length l_b and its width w_b.

Finally, in the three-dimensional bin packing problem, we have B boxes with length l_b, width w_b, and height h_b, $b = 1, \dots, B$. They have to be packed into identical containers with length L, width W, and height H. Again, rotation of the boxes may be allowed. Here, it must be specified in which dimensions the boxes are allowed to be rotated.

Giving a mathematical model for packing problems with more than one dimension is a difficult task. Often the set of feasible box positions within a container is discretized. This leads to large integer programs with a huge number of variables and constraints. We do not give the details here and refer the reader to the different modeling approaches proposed by Beasley [2] as well as Hadjiconstantinou and Christofides [16].

2.2 Knapsack Packing

The knapsack packing problem considers only one container. We have B boxes, and each box $b = 1, \dots, B$ is associated with a value γ_b. The objective is to select boxes to be packed into the container such that the sum of the values of the packed boxes is maximal. Again, of course, the size of the container must be observed. The knapsack packing problem can be given in one, two, or three dimensions. For the size of the boxes and the container, we use the same notation as in the bin packing problem. A packing pattern assigns to each box the information whether it is packed or not, together with the coordinates of the boxes to be packed.

Two special cases should be briefly mentioned: First, the objective to pack as many boxes as possible is obtained from defining $\gamma_b = 1$ for all boxes $b = 1, \dots, B$. Second, the objective to use as much space of the container as possible is achieved by letting the value of each box be equal to its volume. In the three-dimensional case, e.g., we would set $\gamma_b = l_b \cdot w_b \cdot h_b$ for all boxes $b = 1, \dots, B$.

Note that in the one-dimensional case, the knapsack packing problem coincides with the classical knapsack problem (where we would speak of weights instead of lengths). It should be emphasized, however, that there is a difference between the knapsack packing problems described above and other knapsack problems without spatial dimensions. Let us consider an extension of the classical knapsack problem, the so-called multiple knapsack problem, sometimes also called multi-dimensional knapsack problem (see Chu and Beasley [4]). Using binary decision variables x_b

(with $x_b = 1$ if box b is packed and $x_b = 0$ otherwise), this problem can be stated as follows:

$$\begin{aligned} \text{Maximize} \quad & \sum_{b=1}^{B} \gamma_b x_b & & (1) \\ \text{subject to} \quad & \sum_{b=1}^{B} a_{bi} x_b \le A_i & i = 1, \dots, n \qquad & (2) \\ & x_b \in \{0, 1\} & b = 1, \dots, B. \qquad & (3) \end{aligned}$$

Each of the n constraints (2) is a knapsack constraint. This allows to capture, for example, a length constraint (with A_1 being the length of the container and a_{b1} being the length of box b) and a weight constraint (with A_2 being the weight limit of the container and a_{b2} being the weight of box b). However, this does not allow to capture two or more spatial dimensions (length and width of the boxes and the container). Consequently, we will distinguish the knapsack packing problem (with spatial dimensions) on the one hand and the classical and multiple knapsack problems (with non-spatial dimensions) on the other hand.

Let us close this subsection with a brief remark on the size of knapsack packing problems which have been solved exactly. Currently, the largest knapsack packing problems that have been solved to optimality have 2 dimensions, a container with a size of 100×100, and up to 97 boxes (see Fekete and Schepers [12]).

2.3 Strip Packing

In the strip packing problem, we have one container of infinite length. In the two-dimensional case, the width W of the container is given; in the three-dimensional case, also its height H is given. The objective is to pack a given number of boxes such that the used length of the container is minimal. As in the bin packing problem, all boxes have to be packed. A packing pattern is an assignment of coordinates within the container to each box.

Note that this problem type is considered only for two or three dimensions because the one-dimensional case is trivial: There, the used container length would be equal to the sum of the lengths of the boxes.

2.4 Pallet Loading

The pallet loading problem originates from mass production systems, where consumer goods to be distributed are packed on pallets using an automatic pallet loading machine. According the general characterization of Wottawa [31], it is merely a knapsack packing problem with identical boxes. Therefore, the transformation of the knapsack packing problem into a project scheduling problem holds for the pallet loading problem as well. Consequently, the latter will not be treated here in more detail. We only mention that several variants of the pallet loading problem with different constraints can be found in the literature (cf., e.g., Liu and Hsiao [18] and Morabito and Morales [19]).

2.5 Extensions of Packing Problems

In many cases, the basic packing problem types described above are not sufficient to meet all requirements occuring in practice. That is, some packing pattern that fulfills all constraints of a basic problem type may not be useful in practice because other specific demands are not included in the basic problem. In what follows, we briefly summarize a few features that may extend the basic packing problem types.

If some box is heavy, it may be required that it is not packed on top of some other box. This case is relevant especially in the three-dimensional bin packing and knapsack packing problems. Similarly to the previous situation, it may not be allowed to pack other boxes on top of some box.

This may occur, e.g., because of its fragile content. Finally, each box may be associated with a certain weight, and a weight limit for the container(s) may have to be observed (note that in many practical applications containers or machines for container handling are associated with weight limits). Weight can be viewed as another problem dimension which is non-spatial.

2.6 Relationship to Cutting Stock Problems

Cutting problems make up an important class of combinatorial optimization problems that is closely related to packing problems. While packing problems consist of packing small objects (e.g., boxes) into one or more large objects (e.g., containers), cutting problems consist of cutting one or more large objects (raw material such as, e.g., tubes or wood) into small pieces. This implies that there is some kind of duality between packing and cutting problems. From a mathematical point of view, it does not matter whether the pattern computed for a given set of small and large objects is interpreted as a cutting or a packing pattern.

Consider the following example which shows that the one-dimensional bin packing problem can be interpreted as an analogous cutting problem of the same dimension: Wooden planks have to be cut into pieces of given lengths. The objective is to minimize the number of planks required.

There is, however, an important exception for which the duality between both problem classes does not hold. In many cutting problems, only so-called guillotine cuts are allowed. A restriction to guillotine cuts occurs if the material can only be cut completely from one side to the other due to technical requirements (e.g., of a specific saw). As mentioned by Exeler [11], the main difference between packing and cutting problems is the restriction to guillotine cuts in many cutting problems; there is no equivalent of guillotine cuts in packing problems. Note that the restriction to guillotine cuts can affect only two or three-dimensional cutting problems.

As a consequence, we will refer only to packing problems throughout this paper. Due to the duality mentioned above, all upcoming statements and transformation proofs are applicable to cutting problems also, as long as they are not restricted to guillotine cuts.

3 Project Scheduling Models

Many different project scheduling models have been proposed, covering a broad variety of real-world requirements. Starting with a description of the standard model for project scheduling, this section gives an overview of those formal project scheduling concepts that will be used in the transformation approaches to cover packing problems.

3.1 Classical Problem Setting: The RCPSP

The resource-constrained project scheduling problem (RCPSP) has evolved as a standard model in the literature. Although rather simple, it covers many aspects that are relevant in real-world project scheduling.

We consider a project with J activities $j = 1, \ldots, J$. The processing time (or duration) of an activity j is denoted as p_j. The planning horizon is referred to as T. We assume that the planning horizon is divided into time intervals of equal length called periods (e.g., days), and that the processing times p_j are given as discrete multiples of one period. Once started, an activity may not be interrupted, i.e., preemption is not allowed.

Due to technological requirements, there are precedence relations between some of the jobs. These precedence relations are given by sets of immediate predecessors $\mathcal{P}_j$ indicating that an activity j may not be started before each of its predecessors $i \in \mathcal{P}_j$ is completed.

Each activity requires certain amounts of resources to be performed. The resources are called renewable because their full capacity is available in every period. Examples for such resources are manpower and machines. The set of renewable resources is referred to as $\mathcal{K}^\rho$. For each resource

$k \in \mathcal{K}^\rho$ the per-period-availability is assumed to be constant and given by R_k^ρ. Activity j requires r_{jk} units of resource k in each period it is in process.

The parameters are assumed to be nonnegative and integer valued. A solution is given by a schedule which assigns a start time s_j to each activity j, such that the precedence and resource constraints are not violated. The objective is to find a schedule which allows for the earliest possible end of the project, i.e., the minimal makespan. A mathematical programming formulation of the RCPSP has been given by Pritsker et al. [21].

Let us now have a brief look at the size of project scheduling problems which have been solved exactly. From a systematically generated set of standard RCPSP test instances with 30 activities in each project, all projects have been solved to optimality (cf. Demeulemeester and Herroelen [7]). From test sets with larger projects, only some instances have currently been solved exactly.

3.2 Multiple Execution Modes

The basic RCPSP assumes that an activity can only be executed in a single way which is determined by a fixed duration and fixed resource requirements. Starting with the work of Elmaghraby [10], the activity concept as given in the standard RCPSP was extended by allowing several alternatives called modes in which an activity can be performed. Each mode reflects a feasible way to combine a duration and resource requests that allow to accomplish the underlying activity.

We denote the number of modes in which activity j can be performed as M_j. The processing time of activity j being executed in mode m is referred to as p_{jm}. The request of activity j being executed in mode m for renewable resource $k \in \mathcal{K}^\rho$ is r_{jmk}. Once started in one of its modes, an activity must be completed in that mode; mode changes and preemption are not permitted. A solution for a multi-mode project scheduling problem is given by a schedule which assigns a start time and a mode to each activity. For the sake of shortness, we do not give a mathematical formulation here and refer the reader to Talbot [29].

Again, we briefly consider the size of problems for which optimal solutions have been found. The standard test set with the largest multi-mode projects of which all projects have been solved exactly contains projects with 20 activities and 3 modes per activity (cf. Sprecher and Drexl [27]).

3.3 Nonrenewable Resources

In project scheduling models with multiple modes, often so-called nonrenewable resources are considered in addition to the renewable resources already contained in the basic RCPSP. This resource categorization has been developed by Slowinski [24] and Weglarz [30]. Whereas renewable resources are limited on a per-period basis, nonrenewable resources are limited for the entire project. An example for the latter resource category is money; hence, nonrenewable resources allow to model a budget for the project.

For the sake of simplicity, we consider only a single nonrenewable resource. Its availability is given by R^ν. Activity j in mode m requires r_{jm}^ν units of the nonrenewable resource. Note that nonrenewable resources need only be considered in multi-mode models. Within a single-mode environment, the activities' consumption of a nonrenewable resource would be known in advance, that is, it could not be influenced by scheduling decisions.

3.4 Renewable Resource Capacities Varying with Time

In the standard model, the availability of the renewable resources has been assumed to be constant over time, i.e., the capacity is the same in each period. This assumption, however, may not be useful in some cases. Consider, e.g., a changing availability of workers due to holidays. In order to cover time-dependent resource capacities (cf. Sprecher [26]), we denote the availability of renewable resource k at time t as $R_k^\rho(t)$.

3.5 Deadline

Many different kinds of temporal contraints have been considered in the project scheduling literature; for an overview see Bartusch et al. [1]. An important case is a so-called deadline which requires the project to be completed before some given time instant δ. Deadlines are usually used in connection with objectives that consider other criteria than the minimization of the makespan. Such an alternative objective is given below.

3.6 Resource-Based Objective

Motivated by real-world situations, a broad variety of objective functions for project scheduling models have been proposed (cf., e.g., Franck and Schwindt [13]). While the objective to minimize the makespan is among the most popular ones, another important one is the minimization of the nonrenewable resource consumption in a multi-mode environment.

For the purpose of this paper, it is sufficient to consider the following case: The project must be finished before a given deadline δ. We assume that one nonrenewable resource is given. The objective is to find a schedule including a mode assignment m_j for each activity j such that the resulting (mode-dependent) consumption $\sum_{j=1}^{J} r^{\nu}_{jm_j}$ of the nonrenewable resource is minimized. Interpreting the nonrenewable resource as money, such an objective leads to the minimization of the mode-dependent costs required to complete the project on time.

4 Transformation of the Bin Packing Problem

We are now ready to outline formal similarities between packing and project scheduling problems. A few similarities are rather obvious: There is a certain correspondence between boxes to be packed and activities to be scheduled. Furthermore, the constraint that activity preemption is not allowed corresponds to the natural requirement that boxes must be packed as a whole. On the other hand, considering the precedence relations between activities, there is also a project scheduling concept for which there is no similar concept in packing problems.

Let us now start our investigation with a look at the bin packing problem in one, two, and three dimensions. We will show for each dimension which project scheduling model components are needed to capture the respective bin packing variant. Recall, the bin packing problem is to pack a given number of boxes into a minimum number of identical containers. In order to allow a formal transformation into project scheduling models, we assume that the size of the boxes and containers is given in terms of discrete multiples of a basic length unit.

4.1 One Dimension

First, we consider the one-dimensional bin packing problem. As outlined by Garey et al. [14], this problem can be transformed into the RCPSP as follows: For each box b, we define an activity j with processing time $p_j = 1$ and a request for the only renewable resource of $r_{j1} = l_b$ units. We have $J = B$ and $\mathcal{K}^\rho = \{1\}$. The constant capacity of the resource is given by $R^\rho_1 = L$. There are no precedence relations between the activities. Performing an activity in the t-th period corresponds to packing the related box into the t-th container. Consequently, the minimal makespan corresponds to the minimal number of containers needed to pack all boxes.

4.2 Two Dimensions

Now we turn to the two-dimensional case. In a rather simple approach for a similar problem, Schrage [23] proposed to reflect the two spatial dimensions with the resource dimension and the time dimension of an RCPSP with one renewable resource. Each box would correspond to an activity, with a processing time equal to the width and a resource request equal to the length of the box. The problem with this idea is that a resulting schedule only contains start times (which correspond to the x-coordinates of the boxes), whereas the second dimension (the y-coordinates) needed for the packing (or cutting) pattern would not be considered. Also note that the resource constraints of the RCPSP do not treat activities as geometric rectangles, that is, activities are not necessarily assigned to the same resource units over their processing times.

In order to obtain both x and y-coordinates as a result of our model-based approach, we have to adapt the above idea as follows: In addition to the time axis corresponding to the x-coordinates, we reflect the y-coordinates by L renewable resources $k = 1, \ldots, L$ (and hence $\mathcal{K}^\rho = \{1, \ldots, L\}$) with time-varying availability

$$R_k^\rho(t) = \begin{cases} 0, & \text{if } t = n \cdot (W+1) \text{ for some integer } n \geq 1 \\ 1, & \text{otherwise.} \end{cases}$$

Each "block" of consecutive nonzero resource capacity is related to a container. The capacity of 0 is needed to separate the blocks corresponding to different containers. It forces each activity to be scheduled completely within a single block, that is, it ensures that a box is packed into a single container.

Now we introduce an activity j for each box b. Each possible y-coordinate of some box b is reflected by a mode of the related activity j for which we obtain $M_j = L - l_b + 1$ different modes. The duration of any mode $m \in \{1, \ldots, M_j\}$ is given by $p_{jm} = w_b$. The (constant) request of mode m for resource $k \in \{1, \ldots, L\}$ is

$$r_{jmk} = \begin{cases} 1, & \text{if } k \in \{m, \ldots, m + l_b - 1\} \\ 0, & \text{otherwise.} \end{cases}$$

Having obtained a schedule containing start times and modes, the x-coordinate of the left bottom corner of a box is given by the start time s while the y-coordinate is given by the mode number m reduced by 1. From the makespan Z of the schedule, we compute the number of used containers as $\left\lceil \frac{Z}{W+1} \right\rceil$. Note that (x, y)-coordinates of the packing pattern defined above is based on our transformation idea that implicitly puts the containers one after another along the x-axis. This is to say that the x-coordinate includes both the number of the container and the actual x-coordinate in that container. Separating these two different informations, we obtain the container number of a box as $C = \left\lceil \frac{x+1}{W+1} \right\rceil$. Subsequently, the x-coordinate within that container is given by $x' = x - (C-1) \cdot (W+1)$.

Summing up, we have used the RCPSP with multiple modes and time-varying resource capacity to comprise the two-dimensional bin packing problem. That is, each project schedule (with start times and modes for the activities) leads to a well-defined packing pattern (with x and y-coordinates as well as container numbers for the boxes).

The transformation idea is sketched out in Figure 1, where we have some box b with $l_b = w_b = 2$ and a container size of $L = 4$ and $W = 8$, resulting in three modes for the activity related to box b. Observe that the y-axis depicts 4 resources with a time-dependent capacity of 1 or 0 (and not one resource with capacity 4 or 0). Considering the example position of mode $m = 2$, we see that it starts at time $s = 3$ and requires only resources $k = 2$ and $k = 3$. This translates to box coordinates $x = 3$ $(= s)$ and $y = 1$ $(= m - 1)$ in the first container. As already mentioned above, the "gap" between the two blocks with non-zero capacities is used to separate the related containers.

So far, we assumed a fixed orientation for the boxes when they are packed. This assumption is often desired (consider, e.g., corresponding cutting problems with patterned raw material). In many cases, however, it is natural to allow the boxes to be rotated. We reflect rotation as follows: As already mentioned, a rotated box b is obtained from swapping length l_b and width w_b. Following the mode construction above, this simply leads to additional modes that have to be considered. This idea is displayed in Figure 2.

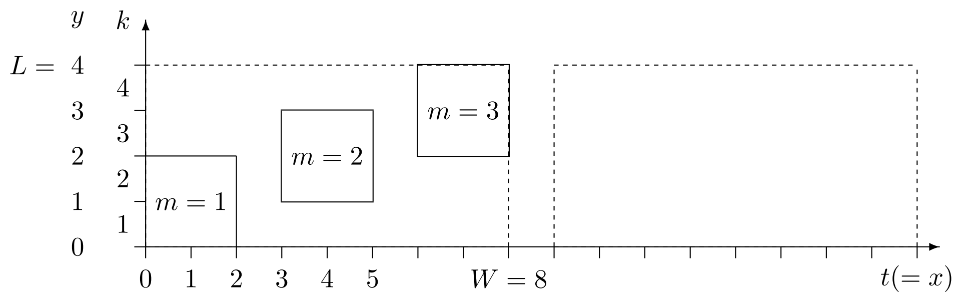

Figure 1: Transforming the two-dimensional bin packing problem

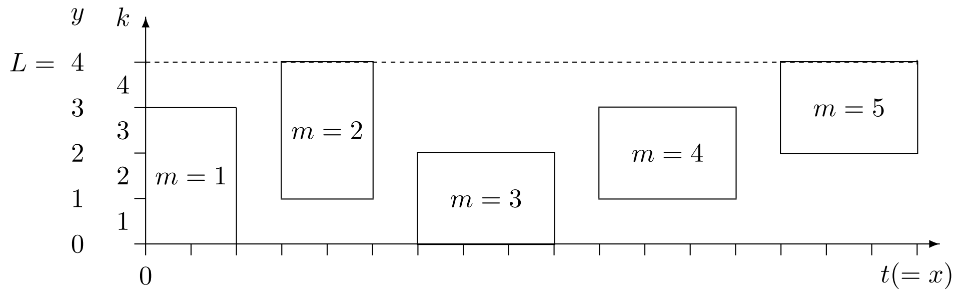

Figure 2: Two-dimensional bin packing problem — rotating boxes

4.3 Three Dimensions

Finally, we examine the three-dimensional bin packing problem. Extending the approach for the two-dimensional case, we reflect the x-coordinate with the time-axis while both the y and z-coordinates are expressed using renewable resources. The project scheduling model under consideration is again the RCPSP with multiple modes and renewable resource capacity varying with time.

We define $L \cdot H$ renewable resources. For the sake of comprehensibility, a resource will be referred to as a pair (y, z) with $y \in \{0, \dots, L-1\}$ and $z \in \{0, \dots, H-1\}$. Each such resource (y, z) is assigned a time-dependent capacity

$$R^{\rho}_{(y,z)}(t) = \begin{cases} 0, & \text{if } t = n \cdot (W+1) \text{ for some integer } n \geq 1 \\ 1, & \text{otherwise.} \end{cases}$$

Next, we define an activity j for each box b. The modes of activity j are denoted as (y_j, z_j), reflecting the possible coordinates of the related box b in the length- and height-dimensions (the width-dimension is again captured by the activity's start time). Hence, we have $y_j \in \{0, \dots, L-l_b\}$ and $z_j \in \{0, \dots, H-h_b\}$, and we obtain $M_j = (L-l_b+1) \cdot (H-h_b+1)$ different modes for activity j. The processing time of activity j in any mode (y_j, z_j) is given by the width w_b of the related box b, that is, $p_{j,(y_j,z_j)} = w_b$. Finally, we have to establish the resource requests which again correspond to the space filled by the box packed at the position related to the mode. Activity j being performed in mode (y_j, z_j) requires renewable resource (y, z) as given by

$$r_{j,(y_j,z_j),(y,z)} = \begin{cases} 1, & \text{if } y \in \{y_j, \dots, y_j + l_b - 1\} \text{ and } z \in \{z_j, \dots, z_j + h_b - 1\} \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, if we want to allow the box to be rotated in some or all planes, we introduce additional modes in analogy to the two-dimensional case.

Given a schedule containing a start time and a mode for each activity j, the x-coordinate of the left bottom front corner of the related box is given by the start time s_j while the y and z-coordinates are given by the mode (y_j, z_j). The number of containers needed is again given as in the two-dimensional case.

5 Transformation of the Knapsack Packing Problem

We express the knapsack packing problem in terms of project scheduling concepts analogously to the respective bin packing problem of the same dimension. We only need a few modifications: In addition to the modes of an activity defined for the bin packing problem, we need one more mode reflecting that the related box is not packed. This mode has a processing time of 0 periods and does not require the renewable resources corresponding to the container space.

Next, we introduce a nonrenewable resource. The request of each of the new modes (which indicate that the related box is not packed) for this nonrenewable resource is equal to the value of that box. For the remaining modes (which imply that the related box is selected to be packed), we define a nonrenewable resource request of 0 units. As we have only one container, we redefine the renewable resource capacities as being constant and equal to 1 if we are dealing with two or three dimensions (in the one-dimensional case, we keep the constant capacity of L units). The width of the container is secured by imposing a deadline on the project. In the one-dimensional case, we set $\delta = 1$, otherwise we define $\delta = W$.

Now we use the objective to minimize the consumption of the nonrenewable resource. Note that this minimizes the sum of the values of the boxes not being packed, which is equivalent to the original value maximization of the knapsack packing problem.

Summing up the transformations given above, the knapsack packing problem (with up to three dimensions) has been shown to be a special case of the RCPSP with multiple modes, a deadline, and a nonrenewable resource based objective (and with constant renewable resource availabilities).

Note that this transformation is not the only possible approach. Alternatively, we can capture the width of the container by allowing time-varying renewable resources. In the one-dimensional case, the resource availability would be 0 after time instant 1, otherwise it would be 0 after time W. Hence, there would be no resources after the time corresponding to the container width, and we could omit the deadline.

Clearly, the knapsack packing problem can be extended by allowing more than one container (but still a limited number of containers). This can be modeled by using the time-varying renewable resource capacities as for the bin packing problem, but with zero capacities for periods later than those corresponding to the given containers.

When introducing the knapsack packing problem, we also mentioned other knapsack problems with non-spatial dimensions. In particular, we discussed the classical knapsack problem and the multiple knapsack problem. As the classical knapsack problem is equivalent to the one-dimensional knapsack packing problem, it can be transformed into a project scheduling problem as outlined above. Let us now consider the multiple knapsack problem. We use the transformation of the one-dimensional knapsack packing problem as starting point. Again, we have a nonrenewable resource based objective, a deadline, and box-related activities with two modes and related nonrenewable resource consumption and processing time. Instead of using a single renewable resource, we now introduce one renewable resource for each of the multiple knapsack constraints. The i-th knapsack constraint is reflected by the i-th renewable resource with capacity A_i. The activity corresponding to box b requires a_{bi} units of the i-th renewable resource. Hence, also the classical knapsack problem and the multiple knapsack problem can be transformed into multi-mode project scheduling problems.

6 Transformation of the Strip Packing Problem

The last packing problem type to be considered is the strip packing problem. Again, the boxes correspond to activities that are defined as for the bin packing problem. One coordinate is related to the start times while the remaining one or two coordinates are reflected by modes (recall that the one-dimensional case of the strip packing problem is trivial). In contrast to the bin packing problem, the renewable resources have a constant availability of one unit per period. Now the makespan corresponds to the used container length. Consequently, the strip packing problem can be modeled by the standard RCPSP with multiple modes (and with constant resource capacities).

7 Transformation of Additional Features

Having discussed the basic packing problem types and dimensions, we now consider extensions of the basic problems which frequently occur in practice.

We start with the case of a heavy box that may not be packed on top of some other box. This requirement is reflected by considering only those modes of the activity related to the heavy box that are associated with a position on the bottom of the container.

Next, we deal with the case of a box on which no other box is allowed to be packed. This constraint can be modeled by adjusting the resource requests of the activity corresponding to that box. This activity would require not only the resources that correspond to the space filled by this box but also the resources reflecting the space above it. Consequently, no resources are left to pack another box on top.

Finally, we address the case of weights associated with the boxes. We consider only the basic knapsack packing problem with one container for which we have to observe a weight limit. Such a weight limit can easily be incorporated by introducing a nonrenewable resource with a capacity corresponding to the weight limit. The request of an activity for that resource is defined as follows: The mode corresponding to the related box not being packed is associated with a request of zero units. All other modes obtain a request corresponding to the weight of the box.

8 Application of the Transformation Approaches

As already mentioned in the introduction, the transformation of problem structures outlined in this paper can lay the foundation for a transfer of algorithmic ideas from one problem class to the other. Our purpose in this section is to demonstrate that such a transfer can be beneficial. In what follows, we discuss the transfer of ideas from project scheduling in order to construct a new heuristic for packing problems. Such heuristics can be applied in software packages in logistics.

First, we briefly summarize a genetic algorithm for multi-mode project scheduling that has recently been proposed by Hartmann [17]. Here, a solution is represented by a list of activities along with a mode for each activity. For such a genotype, a schedule is obtained from successively scheduling the activities in the order prescribed by the list. In each step, the next activity is taken from the list and scheduled as early as possible in the mode given by the genotype. For details on the crossover, mutation, and selection operators, we refer to the work of Hartmann [17]. According to the computational results, this genetic algorithm is currently the most promising heuristic for multi-mode project scheduling.

Our goal is now to develop the framework for a heuristic for the two-dimensional strip packing problem. We use the transformation of the strip packing problem into a multi-mode project scheduling problem and apply the genetic algorithm summarized above. Of course, the activity list corresponds to the order in which the boxes are packed. Scheduling an activity at the earliest possible start time corresponds to shifting the related box as much to the left as possible (in the x-dimension). The mode of an activity represents the coordinate in the y-dimension. The renewable resources control the limited width of the container whereas the (virtually unlimited) time axis reflects the unlimited length. Summing up, the transformation allows us to apply the genetic algorithm for project scheduling. It leads to solutions that correspond to packing patterns. Within the genetic algorithm, promising partial packing patterns can be inherited by means of crossover, mutation, and selection.

Let us now consider two definitions introduced by Sprecher et al. [28] which were originally developed to speed up an exact algorithm for multi-mode project scheduling. Within a given schedule, a multi-mode left shift is defined as a reduction of an activity's finish time, for which a mode change of this activity is allowed. Thereby, the modes and finish times of the other activities may not be changed, and the schedule must remain feasible. A mode reduction on an activity is defined as a reduction of its mode number without changing its finish time and without violating the constraints or changing the modes and finish times of the other activities.

We will now make use of these definitions in order to enhance the genetic algorithm. After computing the earliest start time for an activity, we can apply multi-mode left shifts and mode reductions to it. Translating this into the terminology of packing problems, we can modify the genetic algorithm as follows: The genotype still determines the order in which the boxes are packed. Now we try to shift the current box to the left, but not with respect to a fixed y-coordinate. Instead, we consider any y-coordinate that would allow us to shift the box further to the left. Of course, this corresponds to a multi-mode left shift of an activity. If no more left shifting is possible, we could select the smallest y-coordinate because this would lead to a larger connected space that is not used by the boxes packed so far. This is the idea of the mode reduction.

Note that one would not even have to compute all the modes (i.e., y-coordinates) for an activity because of the regular structure of the modes due to the basic length unit. The actual mode could be computed when shifting the current activity. This observation implies that the seeming drawback, the explicit computation of all possible coordinates in advance, can easily be avoided. This makes the implementation of the packing heuristic more efficient. To put it in a nutshell, project scheduling concepts have led us to a genetic algorithm for strip packing. Note that the concepts outlined here can similarly be used for the bin packing problem.

9 Summary and Conclusions

We have examined structural similarities between packing problems on the one hand and project scheduling problems on the other hand. It was shown that the main packing problem types, namely the bin packing problem, the knapsack packing problem, and the strip packing problem, can be viewed as special cases of different project scheduling models. This proposition was shown to hold for several extensions of packing problems as well, including rotation of boxes, weight limit of a container, and restrictions on box stacking. The main idea behind the transformations is that boxes to be packed correspond to activities to be scheduled, and that the container space in packing can be viewed as a resource with limited capacity. Table 1 summarizes the transformations of packing problems into project scheduling problems. For each of the considered packing problem types and each related (non-trivial) dimension, the table states which of the variants and extensions of the basic RCPSP are needed to capture the constraints and objective.

As already mentioned in the introduction, the main motivation for dealing with the relationships between between two different problem classes is to provide a basis for the transformation of algorithms originally designed for one problem class to the other class. Having shown that packing problems are a special case of project scheduling problems, we can apply project scheduling algorithms to packing problems. The proofs outlined here can serve as guidelines for the transfer of algorithmic ideas. In this paper, we have sketched out how to apply a recently proposed genetic algorithm for project scheduling to the two-dimensional strip packing problem after its transformation. Thus, the examination of relationships between problem classes not only provides deeper theoretical insights but can also lead to algorithms that are applicable in practice. In fact, packing problems play an important role in logistics. Heuristics like the one discussed here are the key feature of software for optimized container loading. Moreover, packing problems are an important

Problem	dim.	modes	nonrenewable resource	deadline	capacities of renewable res.	objective
bin packing	1	single	no	no	constant	makespan
	2	multiple	no	no	time-varying	makespan
	3	multiple	no	no	time-varying	makespan
knapsack packing	1	multiple	yes	yes	constant	resource
	2	multiple	yes	yes	constant	resource
	3	multiple	yes	yes	constant	resource
classical knapsack	1	multiple	yes	yes	constant	resource
multiple knapsack	> 1	multiple	yes	yes	constant	resource
strip packing	2	multiple	no	no	constant	makespan
	3	multiple	no	no	constant	makespan

Table 1: Packing problems as special cases of project scheduling models

part of many software systems for tour planning.

References

[1] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16:201–240, 1988.

[2] J. E. Beasley. An exact two-dimensional non-guillotine cutting stock tree search procedure. *Operations Research*, 33:49–64, 1985.

[3] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112:3–41, 1999.

[4] P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. Technical report, The Management School, Imperial College, London, England, 1997.

[5] B. de Reyck and W. S. Herroelen. Assembly line balancing by resource-constrained project scheduling – A critical appraisal. Technical Report 9505, Katholieke Universiteit Leuven, Belgium, 1995.

[6] E. L. Demeulemeester and W. S. Herroelen. Modelling setup times, process batches and transfer batches using activity network logic. *European Journal of Operational Research*, 89:355–365, 1996.

[7] E. L. Demeulemeester and W. S. Herroelen. New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43:1485–1492, 1997.

[8] A. Drexl and F. Salewski. Distribution requirements and compactness constraints in school timetabling. *European Journal of Operational Research*, 102:193–214, 1997.

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

[10] S. E. Elmaghraby. *Activity networks: Project planning and control by network models.* Wiley, New York, 1977.

[11] H. Exeler. *Das homogene Packproblem in der betriebswirtschaftlichen Logistik.* Physica, Heidelberg, Germany, 1988.

[12] S. P. Fekete and J. Schepers. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In *Algorithms – ESA '97, 5th Annual European Symposium*, volume 1284 of *Lecture Notes in Computer Science*, pages 144–156. Springer, Berlin, Germany, 1997.

[13] B. Franck and C. Schwindt. Different resource-constrained project scheduling models with minimal and maximal time-lags. Technical Report WIOR-450, Universität Karlsruhe, Germany, 1995.

[14] M. R. Garey, R. L. Graham, D. S. Johnson, and A. C.-C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory*, 21:257–298, 1976.

[15] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* Freeman, San Francisco, California, 1979.

[16] E. Hadjiconstantinou and N. Christofides. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, 83:39–56, 1995.

[17] S. Hartmann. Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102:111–135, 2001.

[18] F.-H. F. Liu and C.-J. Hsiao. A three-dimensional pallet loading method for single-size boxes. *Journal of the Operational Research Society*, 48:726–735, 1997.

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

[20] L. Özdamar and G. Ulusoy. A survey on the resource-constrained project scheduling problem. *IIE Transactions*, 27:574–586, 1995.

[21] A. A. B. Pritsker, L. J. Watters, and P. M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16:93–107, 1969.

[22] A. Scholl, R. Klein, and C. Jürgens. BISON: A fast hybrid procedure for exactly solving the onedimansional bin packing problem. *Computers & Operations Research*, 24:627–645, 1997.

[23] L. Schrage. Solving resource-constrained network problems by implicit enumeration – Nonpreemptive case. *Operations Research*, 18:263–278, 1970.

[24] R. Slowinski. Two approaches to problems of resource allocation among project activities: A comparative study. *Journal of the Operational Research Society*, 31:711–723, 1980.

[25] A. Sprecher. A competitive exact algorithm for assembly line balancing. *International Journal of Production Research*. Forthcoming.

[26] A. Sprecher. *Resource-constrained project scheduling: Exact methods for the multi-mode case.* Number 409 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, Germany, 1994.

[27] A. Sprecher and A. Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107:431–450, 1998.

[28] A. Sprecher, S. Hartmann, and A. Drexl. An exact algorithm for project scheduling with multiple modes. *OR Spektrum*, 19:195–203, 1997.

[29] F. B. Talbot. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, 28:1197–1210, 1982.

[30] J. Weglarz. On certain models of resource allocation problems. *Kybernetics*, 9:61–66, 1981.

[31] M. Wottawa. PACKLIB: Ein ASCII-Datenformat für Packungsprobleme. Technical Report 96-216, Universität Köln, Germany, 1996.